“FAKE NEWS DETECTION USING MACHINE LEARNING”

A PROJECT REPORT

Submitted by

BHAVIK KORADIYA (170140111036)

AALOK SHAH (170140111092)

SHAIKH REHANAKHATUN (170140111096)

SAURABH SINGH (170140111111)

In the fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS & COMMUNICATION

GOVERNMENT ENGINEERING COLLEGE, BHARUCH

Gujarat Technological University, Ahmedabad

ACADEMIC YEAR (2020-21)

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude towards Prof. Kunjal Tandel, our project guide and Head of the Department of Electronics & Communication Engineering who has been the driving force behind this project and whose guidance and co-operation has been a source of inspiration for us.

We are very much thankful to the professors, colleagues and authors of various publications to which we have been referring to. We express our sincere appreciation and thanks to all those encouragements that was provided on numerous occasions by our whole division. Finally, we thank our parents for their immense support.

BHAVIK KORADIYA (170140111036)
AALOK SHAH (170140111092)
SHAIKH REHANAKHATUN (170140111096)
SAURABH SINGH (170140111111)

ABSTRACT

Currently the covid-19 pandemic hassled to an increase in the popularity and spread of fake news. Fake news has quickly become a society problem, being used to propagate false or rumor information in order to change everyone's behaviour. It has been realized that propagation of fake news has had a non-negligible influence among the people all over the world.

As demonstrated by the widespread effects of the large onset of fake news, humans are inconsistent if not outright poor detectors of fake news. With this, efforts have been made to automate the process of fake news detection. The most popular of such attempts include blacklists of sources and authors that are unreliable. While these tools are useful, in order to create a more complete end to end solution, we need to account for more difficult cases where reliable sources and authors release fake news.

As such, the goal of this project was to create a tool for detecting the language patterns that characterize fake and real news through the use of machine learning and natural language processing techniques. The results of this project demonstrate the ability for machine learning to be useful in this task. We have built a model that catches many intuitive indications of real and fake news as well as an application that aids in the visualization of the classification decision.

GOVERNMENT ENGINEERING COLLEGE,
BHARUCH

ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Date: / /2021

This is to certify that the dissertation entitled "**FAKE NEWS DETECTION USING MACHINE LEARNING**" has been carried out by **BHAVIK KORADIA** under my guidance in fulfilment of the degree of Bachelor of Engineering in Electronics & Communication (8th Semester) of Gujarat Technological University, Ahmedabad during the academic year 2020-21.

Internal Guide **Head of the Department**

GOVERNMENT ENGINEERING COLLEGE,
BHARUCH

ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Date: / /2021

This is to certify that the dissertation entitled "**FAKE NEWS DETECTION USING MACHINE LEARNING**" has been carried out by **AALOK SHAH** under my guidance in fulfilment of the degree of Bachelor of Engineering in Electronics & Communication (8th Semester) of Gujarat Technological University, Ahmedabad during the academic year 2020-21.

Internal Guide **Head of the Department**

GOVERNMENT ENGINEERING COLLEGE,
BHARUCH

ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Date: / /2021

This is to certify that the dissertation entitled "**FAKE NEWS DETECTION USING MACHINE LEARNING**" has been carried out by **SHAIKH REHANAKHATUN** under my guidance in fulfilment of the degree of Bachelor of Engineering in Electronics & Communication (8th Semester) of Gujarat Technological University, Ahmedabad during the academic year 2020-21.

Internal Guide **Head of the Department**

GOVERNMENT ENGINEERING COLLEGE,
BHARUCH

ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

Date: / /2021

This is to certify that the dissertation entitled "**FAKE NEWS DETECTION USING MACHINE LEARNING**" has been carried out by **SAURABH SINGH** under my guidance in fulfilment of the degree of Bachelor of Engineering in Electronics & Communication (8th Semester) of Gujarat Technological University, Ahmedabad during the academic year 2020-21.

Internal Guide **Head of the Department**

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 OVERVIEW

The rise of fake news during the 2016 U.S. Presidential Election highlighted not only the dangers of the effects of fake news but also the challenges presented when attempting to separate fake news from real news. Fake news may be a relatively new term but it is not necessarily a new phenomenon. Fake news has technically been around at least since the appearance and popularity of one-sided, partisan newspapers in the 19th century. However, advances in technology and the spread of news through different types of media have increased the spread of fake news today. As such, the effects of fake news have increased exponentially in the recent past and something must be done to prevent this from continuing in the future.

What is fake news?

Fake news's simple meaning is to incorporate information that leads people to the wrong path. Nowadays fake news spreading like water and people share this information without verifying it. This is often done to further or impose certain ideas and is often achieved with political agendas.

For media outlets, the ability to attract viewers to their websites is necessary to generate online advertising revenue. So it is necessary to detect fake news.

1.2 MOTIVATION

We have identified the three most prevalent motivations for writing fake news and chosen only one as the target for this project as a means to narrow the search in a meaningful way. The first motivation for writing fake news, which dates back to the 19th century one-sided party newspapers, is to influence public opinion. The second, which requires more recent advances in technology, is the use of fake headlines as click-bait to raise money. The third motivation for writing fake news, which is equally prominent yet arguably less dangerous, is satirical writing. While all three subsets of fake news, namely, click-bait, influential, and satire, share the common thread of being fictitious, their widespread effects are vastly different. As such, this report will focus primarily on fake news as defined by politifact.com, fabricated content that intentionally masquerades as news coverage of actual events.‖ This definition excludes satire, which is intended to be humorous 8 and not deceptive to readers. Most satirical articles come from sources like The Onion, which specifically distinguish themselves as satire. Satire can already be classified, by machine learning techniques according to.

Therefore, our goal is to move beyond these achievements and use machine learning to classify, at least as well as humans, more difficult discrepancies between real and fake news. The dangerous effects of fake news, as previously defined, are made clear by events such as in which a man attacked a pizzeria due to a widespread fake news article. This story along with analysis from provide, evidence that humans are not very good at detecting fake news, possibly not better than chance. As such, the question remains whether or not machines can do a better job?

1.3 SIGNIFICANCE OF PROJECT

Fake News has been gatherings lot of attention worldwide recently. The effects can be political, economic, organizational, or even personal. Many of the social sites consist of fake news which confuses the society. These false news sometimes put the society in a great danger. The current project involves utilizing machine learning and natural language processing techniques to create a model that can expose whether the news/social media posts are fake or not. With this, the implementation of machine learning project based on classification and detection is proposed Using of bag-of-words, n-grams, count vectorizer and training of data on various classifiers to investigate which of them works well for this specific dataset of labelled news statements. This project eliminates the issues caused due to spreading of Fake News.

1.4 SCOPE OF STUDY

The fake news challenge is perilous and is spreading rapidly like a wildfire as it becomes easier for information to reach the mass in various flavors. With the help of artificial intelligence, we can control and limit the spread of such misinformation more quickly and efficiently as compared to manual efforts. The work in this project proposes a stacked model which fine tunes the informational insight gained from the data at each step and then tries to make a prediction. Although, many attempts have been made to solve the problem of fake news, any significant success is yet to be seen. With huge amounts of data collected from social media websites like Facebook, Twitter, etc., the best models improve every day. With the use of deep neural networks, the future work in this field seems a lot more promising. The limitations that come packaged with this problem is that, the data is erratic and this means that any type of prediction model can have anomalies and can make mistakes. For future improvements, concepts like POS tagging, word2vec and topic modelling can be utilized. These will give the model a lot more depth in terms of feature extraction and fine-tuned classification.

CHAPTER 2: USED TOOLS/ SOFTWARES/ LIBRARIES

2.1 PYTHON

Overview:

- Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation.
- It was designed with motive to enhance code readability, and its syntax allows programmers to express their concepts in fewer lines of code.
- Python is a programming language that lets you work quickly and integrate systems more efficiently.
- It supports multiple programming paradigms. It also performs automatic memory management.
- Python is best choice for implementing Artificial Intelligence and Machine Learning based Algorithms because of its richness of libraries and community support.

Specification / Features:

- Extensive support libraries (NumPy for numerical calculations, Pandas for data analytics, TensorFlow, Keras for AI and ML etc.)
- Open source and community development
- Easy to learn
- User-friendly data structures
  - High-level language
  - Dynamically typed language (No need to mention data type based on value assigned, it takes data type)
  - Object-oriented language
  - Portable and Interactive
  - Portable across Operating systems

2.2 JUPYTER NOTEBOOK

Overview:

- Jupyter notebook is a client-server application. The application starts the server on local machine and opens the notebook interface in web browser where it can be edited and run from. The notebook is saved as ipynb file and can be exported as html, pdf and LaTeX files.
- Jupyter Notebook was created to make it easier to show one's programming work, and to let others join in. Jupyter Notebook allows you to combine code, comments, multimedia, and visualizations in an interactive document — called a notebook, naturally — that can be shared, re-used, and re-worked.
- And because Jupyter Notebook runs via a web browser, the notebook itself could be hosted on your local machine or on a remote server.

Specification / Features:

- Data visualizations.
- Code sharing.
- Live interactions with code. Jupyter Notebook code isn't static; it can be edited and re-run incrementally in real time, with feedback provided directly in the browser. Notebooks can also embed user controls (e.g., sliders or text input fields) that can be used as input sources for code.
- Documenting code samples

2.3 NUMPY

Overview:

- NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided.

Specification / Features:

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and randomnumber generation.
- NumPy is a Python library and is written partially in Python, but most of the parts that requirefast computation are written in C or C++.
- NumPy is basic and common library(dependency) for various OpenCV, AI/ML libraries etc.

2.4 PANDAS

Overview:

- Pandas is an open-source Python Library providing high-performance data manipulation andanalysis tool using its powerful data structures.
- Using Pandas, we can accomplish five typical steps in the processing and analysis of data,regardless of the origin of data — load, prepare, manipulate, model, and analyze.
- Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Specification / Features:

- Fast and efficient Data Frame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and sub setting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

2.5 NLTK

Overview:

- The NLTK module is a massive tool kit, aimed at helping you with the entire Natural Language Processing (NLP) methodology. NLTK will aid you with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, and then even with helping your machine to understand what the text is all about.

Specification/ Features:

- NLTK scores very high when it comes to the ease of use and explanation of the concept.
- The learning curve of Python is very fast and NLTK is written in Python so NLTK is alsohaving very good learning kit.
- NLTK has incorporated most of the tasks like tokenization, stemming, Lemmatization,Punctuation, Character Count, and Word count.
- It is very elegant and easy to work with.

2.6 SCIKIT - LEARN

Overview:

- Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Specification/ Features:

- Rather than focusing on loading, manipulating and summarizing data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −
  - **Supervised Learning algorithms** − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
  - **Unsupervised Learning algorithms** − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
  - **Clustering** − This model is used for grouping unlabeled data.
  - **Cross Validation** − It is used to check the accuracy of supervised models on unseen data.
  - **Dimensionality Reduction** − It is used for reducing the number of attributes in data which can be further used for summarization, visualization and feature selection.
  - **Ensemble methods** − As name suggest, it is used for combining the predictions of multiple supervised models.
  - **Feature extraction** − It is used to extract the features from data to define the attributes in image and text data.
  - **Feature selection** − It is used to identify useful attributes to create supervised models.
  - **Open Source** − It is open source library and also commercially usable under BSD license.

2.7 GIT

Git is the most commonly used version control system. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

So regardless of whether you write code that only you will see, or work as part of a team, Git will be useful for you.

Git is software that runs locally. Your files and their history are stored on your computer. You can also use online hosts (such as GitHub or Bitbucket) to store a copy of the files and their revision history. Having a centrally located place where you can upload your changes and download changes from others, enable you to collaborate more easily with other developers. Git can automatically merge the changes, so two people can even work on different parts of the same file and later merge those changes without loosing each other's work!

CHAPTER 3: PROCESS / METHODOLOGY

3.1 NLP – NATURAL LANGUAGE PROCESSING

Overview:

- Language is a method of communication with the help of which we can speak, read and write. For example, we think, we make decisions, plans and more in natural language; precisely, in words. However, the big question that confronts us in this AI era is that can we communicate in a similar manner with computers. In other words, can human beings communicate with computers in their natural language? It is a challenge for us to develop NLP applications because computers need structured data, but human speech is unstructured and often ambiguous in nature.
- In this sense, we can say that Natural Language Processing (NLP) is the sub-field of Computer Science especially Artificial Intelligence (AI) that is concerned about enabling computers to understand and process human language. Technically, the main task of NLP would be to program computers for analysing and processing huge amount of natural language data.

NLP Phases:

1) Morphological Processing

It is the first phase of NLP. The purpose of this phase is to break chunks of language input intosets of tokens corresponding to paragraphs, sentences and words. For example, a word like–uneasy can be broken into two sub-word tokens as –un-easy.

2) Syntax Analysis

It is the second phase of NLP. The purpose of this phase is two folds: to check that a sentence is well formed or not and to break it up into a structure that shows the syntactic relationships between the different words. For example, the sentence like –The school goes to the boy would be rejected by syntax analyzer or parser.

3) Semantic Analysis

It is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text. The text is checked for meaningfulness. For example, semantic analyzer would reject a sentence like –Hot ice-cream.

4) Pragmatic Analysis

It is the fourth phase of NLP. Pragmatic analysis simply fits the actual objects/events, which exist in a given context with object references obtained during the last phase (semantic analysis). For example, the sentence –Put the banana in the basket on the shelf‖ can have two semantic interpretations and pragmatic analyzer will choose between these two possibilities.

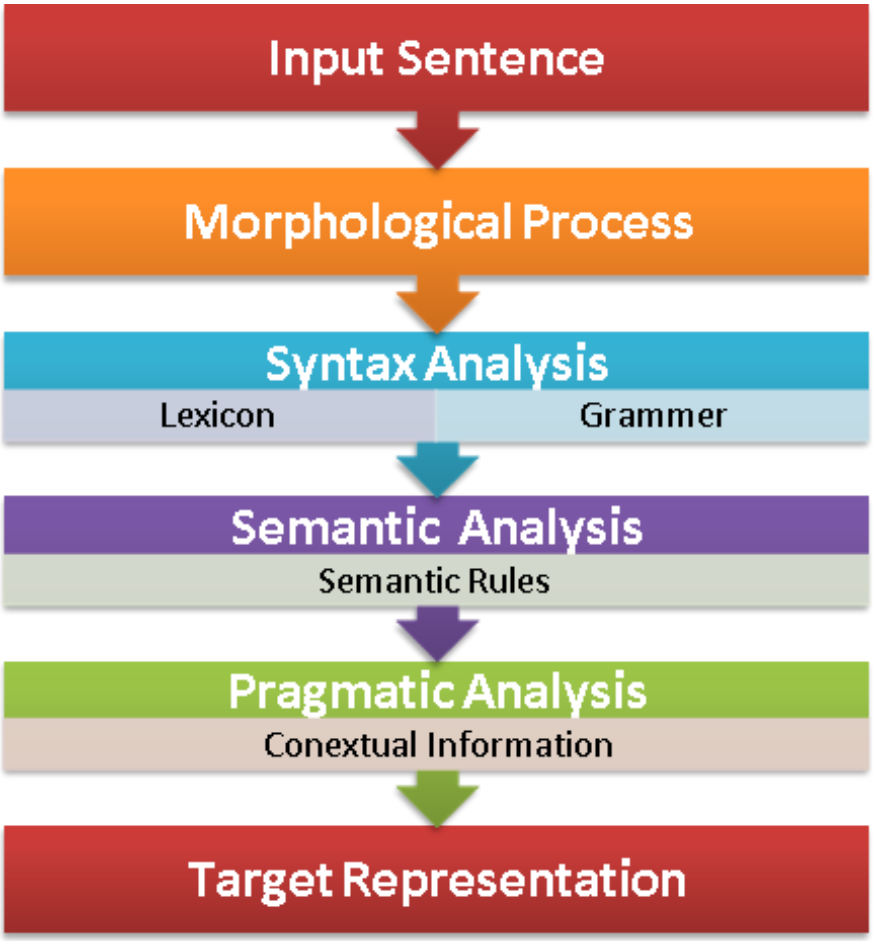


Figure 1: NLP Phases Flow Chart

Applications:

1) Text Classification and Categorization

Text classification is an essential part in many applications, such as web searching, information filtering, language identification, readability assessment, and sentiment analysis. Neural networks are actively used for these tasks.

2) Paraphrase Detection

Paraphrase detection determines whether two sentences have the same meaning. This task is especially important for question answering systems since there are many ways to ask the same question.

3) Language Generation and Multi-document Summarization

Natural language generation has many applications such as automated writing of reports, generating texts based on analysis of retail sales data, summarizing electronic medical records, producing textual weather forecasts from weather data, and even producing jokes.

4) Machine Translation

Machine translation software is used around the world despite its limitations. In some domains, the quality of translation is not good. To improve the results researchers, try different techniques and models, including the neural network approach.

5) Speech Recognition

Speech recognition has many applications, such as home automation, mobile telephony, virtual assistance, hands-free computing, video games, and so on. Neutral networks are widely used in this area.

6) Character Recognition

Character Recognition systems also have numerous applications like receipt character recognition, invoice character recognition, check character recognition, legal billing document character recognition, and so on.

3.2 ARTIFICIAL NEURAL NETWORK

Overview:

- The term –neural comes from –neuron, which is the term used for a single nerve cell. That's right – a neural network essentially means a network of neurons that perform simple and actions in our daily lives.
- Pattern recognition, object detection, and intelligence are a major aspect of the problems we faceevery day. While they are performed with so much ease that we don't even realize, the truth is that these reactions are difficult to automate.
- A multi-layer, fully-connected neural network containing an input layer, hidden layers, and an output layer is called an artificial neural network or ANN.

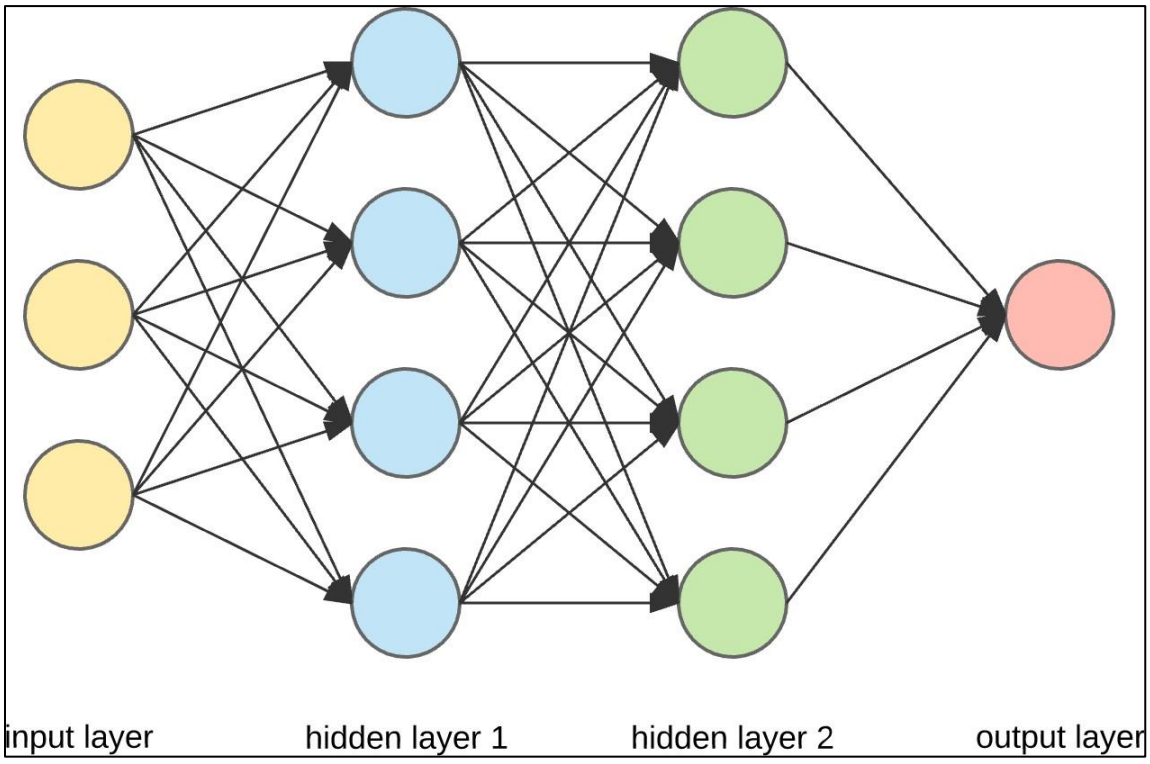


Figure 2: Block Diagram of ANN

Glossary of Artificial Neural Network Model:

1) Inputs

The data first fed into the neural network from the source is called the input. Its goal is to give the network data to make a decision or prediction about the information fed into it. The neural network model usually accepts real value sets of inputs and it should be fed into a neuron in the input layer.

2) Training set

The inputs for which you already know the correct outputs are called training sets. These are used to help the neural network get trained and memorize the result for the given input set.

3) Outputs

Every neural network generates an output as a prediction or decision about the data fed into it. This output is in the form of real values set or Boolean decisions. Only of the neurons in the output layer generates the output value.

4) Neuron

Also known as a perceptron, a neuron is the basic unit of a neural network. It accepts an input value and generates an output based on it.

As discussed before, every neuron receives a part of the input and passes it through the non-linear activation function to the node in the next layer. These activation functions can be TanH, sigmoid, or ReLu. The non-linear feature of these functions helps to train the network.

5) Weight space

Every neuron has a numeric weight. When it delivers input to another note, its weight is totaled with the others to generate an output. By making small changes to these weights are how neural networks are trained. The fine-tuning of weights helps determine the correct set of weights and biases that would generate the best outcome. This is where back-propagation comes in.

Back-propagation

One of the ways to successfully find out the small changes that need to be made to the weights to minimize the loss of the entire network is back-propagation.

- At first, the activations are to be propagated in the upward or feed forward direction.
- Now, the cost function derivatives have to be propagated in the downward or reverse direction.

This way, we will be able to determine the partial cost derivative against each weight. We can then compute the cost that would be reduced by making the adjustments.

CHAPTER 4: APPROACHES TO DETECT FAKE / FALSE NEWS

4.1 FACT CHECKING

Fact checking is a form of –knowledge-based study of fake news which focuses on assessing the truthfulness of news. There are two types of fact checking, namely manual and automatic.

- **Manual fact checking**

The process of manual fact checking is one that is done by humans and it can be done by either experts or regular people.

- **Expert-based**

This method depends on professionals in the fact-checking field, also called fact-checkers, toauthenticate a particular news content. It is typically done by a few but very reliable fact-checkers. This approach is relatively simple to conduct and is also very accurate. However, the disadvantages of this method are that it is expensive and the system is likely to be overwhelmed as the amount of news content to be verified increases.

- **Crowd-sourced**

This alternative type of fact checking requires a huge group of normal individuals who serve as fact checkers. This form of fact checking is not as easy to conduct and the results are likely to be less reliable and accurate due to the biases of the fact checkers as well as possibleclashes between them in annotations of the news content. However, as compared to expert- based fact checking, it is less likely for this crowdsourcing fact checking system to be overwhelmed when the volume of news content to be authenticated increases. In this type of fact checking, it is important to sieve out unreliable users and iron out any results that may contrast each other. These concerns would become more crucial as the fact checking population expands. Nevertheless, individuals who fact check on these crowd-sourcing sites are more able to provide more comprehensive feedback such as including their attitudes or opinions.

- **Automatic fact checking**

  A big problem with manual fact-checking is that the systems are easily overwhelmed by growing numbers of fresh news content that needs to be checked, which is very prevalent in the case of social media. Hence, automatic fact checking methods have been created to combat this problem. These approaches mostly depend on –Information Retrieval (IR) and Natural Language Processing (NLP) techniques, as well as on network/graph theory. Automatic fact checking methods generally comprise two steps, fact extraction and fact checking. In fact, extraction, also known as knowledge-based construction, knowledge is takenfrom the Web as –raw facts‖ and it is typically unnecessary, obsolete, conflicting, inaccurate or not complete. They will then be refined and cleaned up by –knowledge processing tasks to build a knowledge-base or a knowledge graph‖. Secondly, fact checking, also known as knowledge comparison, is done to assess the veracity of the news content. This is accomplished by matching the knowledge taken from the to-be-checked news content againstthe facts found in the current –knowledge-base(s) or knowledge graph(s).

4.2 DEEP SYNTAX ANALYSIS

Deep syntax can be analyzed using Probabilistic context-free grammar (PCFG). Syntax structures are described by changing sentences into parse trees. Nouns, verbs etc. are rewritten into their syntactic constituent parts. Probabilities are assigned to the parse tree. This method identifies the rule categories like lexicalization and parent nodes etc. It detects deception with 85-91% accuracy, depending on the category used in the analysis.

4.3 ACCOUNT ANALYSIS

The credibility in Twitter events by creating a data set of tweets that is relevant to trending topics was detected. Using crowd sourcing, they annotated the data sets regarding the veracity of each tweet. 4 features, namely message, user, topic and propagation were analyzed using a Decision Tree Model. This method achieved 86% accuracy. Benevuto etal came up with a model that detects spammers by constructing a manual annotated dataset of 1000 records of spam and non-spam accounts. Attributes on content and user behavior were extracted and analyzed. This method successfully detected 70% of spam accounts and 96% of non-spam accounts. Chu et al developed a similar detection model which distinguished bot accounts. 3 groups were categorized - humans, bots and cyborgs. A system was built with 4 features of analysis, namely entropy measures, spam detection, account properties and decision making. This method successfully identified the human’ class at 96% accuracy.

4.4 CREDIBILITY-BASED FAKE NEWS

This approach looks at fake news ―based on news-related and social-related information. For instance, intuitively, a news article published on unreliable website(s) and forwarded by unreliable user(s) is more likely to be fake news than news posted by authoritative and credible users‖. In other words, this approach focuses on the source of the news content. As such, the credibility perspective of studying fake news generally overlaps with a propagation-based study of fake news.

- **Assessing news headline credibility**

  This method typically revolves around identifying click-bait, which are headers that aim to capture users' attention and lead them to click on a link to a certain web page. Existing click- bait detection studies use both ―linguistic features such as term frequencies, readability, and forward references and non-linguistic features such as webpage links. ―user interests, ―and headline stance ―within a supervised learning framework such as gradient boosted decision
  trees ―to identify or block click-baits‖. Empirical studies have suggested that click-baits are typically defined by ―a cardinal number, easy readability, strong nouns and adjectives to convey authority and sensationalism.

- **Assessing news source credibility**

  This approach generally looks at the ―quality, credibility, and political bias of source websites in order to assess the quality and reliability of news content.

- **Assessing news comments credibility**

  The credibility of news content can also be evaluated via the credibility of the comments associated with it. ―User comments on news websites and social media carry invaluable information on stances and opinion‖ although it is very common for them to be overlooked. There are a few models that can be used to assess comment credibility and they can be classified into three types, content-based, behavior-based and graph(network)-based.

- **Content-based-models**

  These models evaluate comment credibility by leveraging on language features taken from user comments and the strategy it adopts is comparable to that of style-based fake news detection.

- **Behavior-based-models**

These models often make use of the ―indicative features of unreliable comments extracted from the metadata associated with user behavior. Looking at review spam detection studies, these related behavioral attributes can be sorted into five categories, namely, burstiness, activity, timeliness, similarity, and extremity.

- **Assessing news spreader credibility**

- Lastly, the credibility of news content can also be evaluated by looking at the users who spread the particular news content and assessing their reliability. Users are a vital part of the propagation of deceptive news as they can spread fake news through various ways such as sharing, forwarding, liking and reviewing. In this process, users can be categorized into two types, malicious users who typically have low reliability and normal users who generally have higher reliability. Malicious users deliberately spread deceptive news in search of monetary and/or non-monetary benefits such as power and popularity.

- This group of users can be split into three categories. Firstly, bots, which are software applications ―that run automated tasks or scripts over the Internet. Secondly, trolls, which are people who bicker or agitate other users with the aim of distracting and ruining relationships between people. They generally do this by posting provocative, digressive or irrelevant messages in order to instigate other users to respond with strong emotional content. The last category is cyborgs which are accounts that are registered by humans as a cover in order to run ―automated programs performing online activities.

- On the contrary, naïve users are regular users who inadvertently join in on the spreading of deceptive news as they misinterpret deceitful news to be the truth. There are two main factors that have been studied that may help explain why naïve users may participate in the spreading of fake news. The first factor is social influence, which ―refers to environmental and exogenous factors such as network structure or peer pressure that can influence the dynamics of fake news‖. This is demonstrated by ―the bandwagon effect, normative influence theory and social identity theory which illustrates that ―peer pressure psychologically impacts user behavior towards fake-news-related activities. The second factor is self-influence. This refers to the intrinsic characteristics of users that can affect how they react to or handle deceptive news. For instance, according to confirmation bias and naïve realism, users are more likely to believe in deceptive news or participate in its related activities if it validates their pre-existing knowledge.

CHAPTER 5: FAKE NEWS DETECTION

5.1 OVERVIEW

This is the advanced python project of distinguishing fake news and real news. We instate a Logistic Regression, Decision Tree Classifier, Gradient Boosting Classifier, Random Forest Classifier and fit the model. Eventually, the exact score and the confusion matrix reveal to us how well our model passages.

5.2 DATASETS

Data-Set URL: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

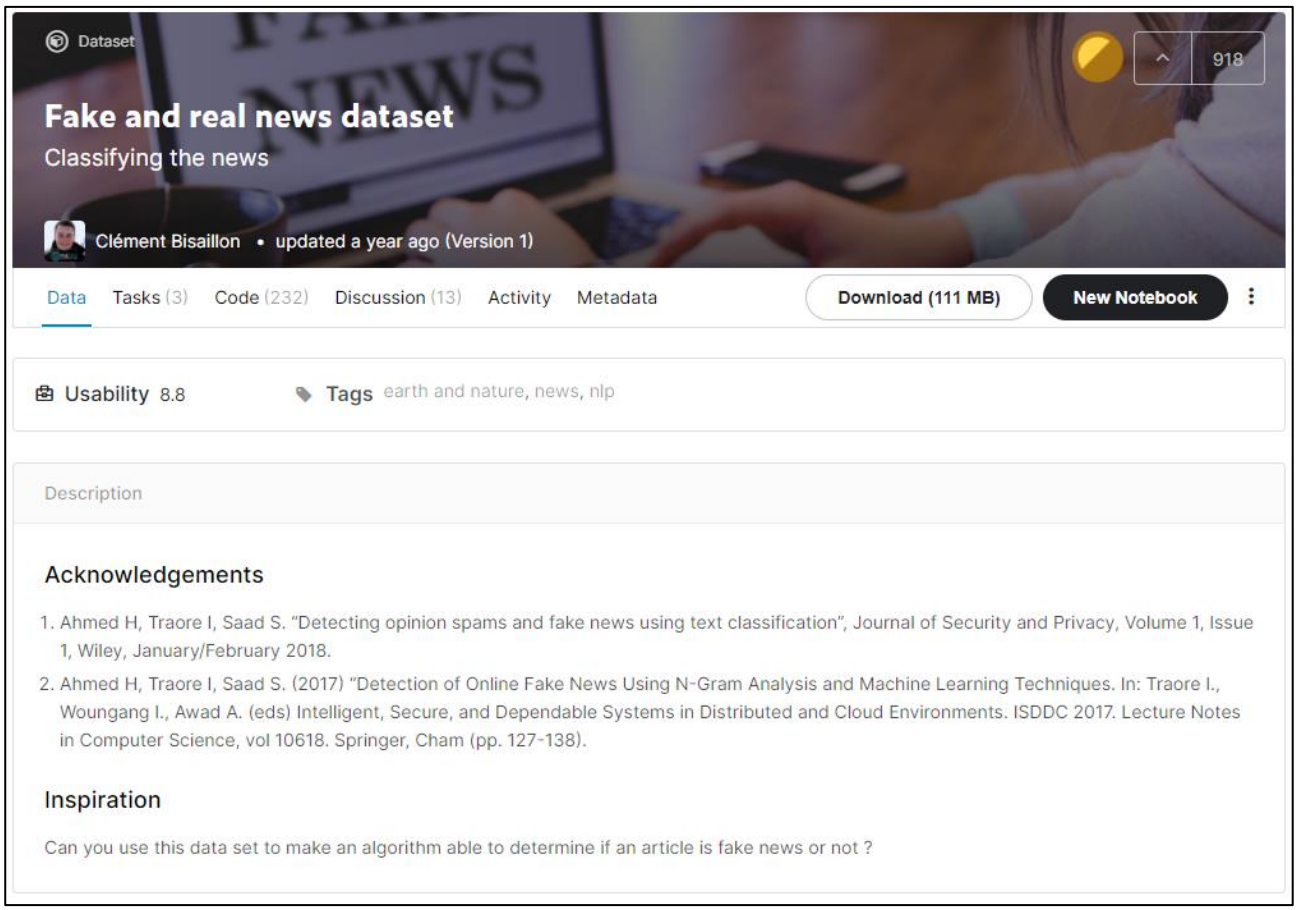


Figure 3: Fake News Dataset (Kaggle)

This dataset has True.csv and Fake.csv files which contains True news and Fake news respectively. Both files has around 22000 of Unique News and 4 Attributes (i.e. Title, Text, Subject and Date). Combined size of both csv is 110.98 MB.

5.3 BLOCK DIAGRAM

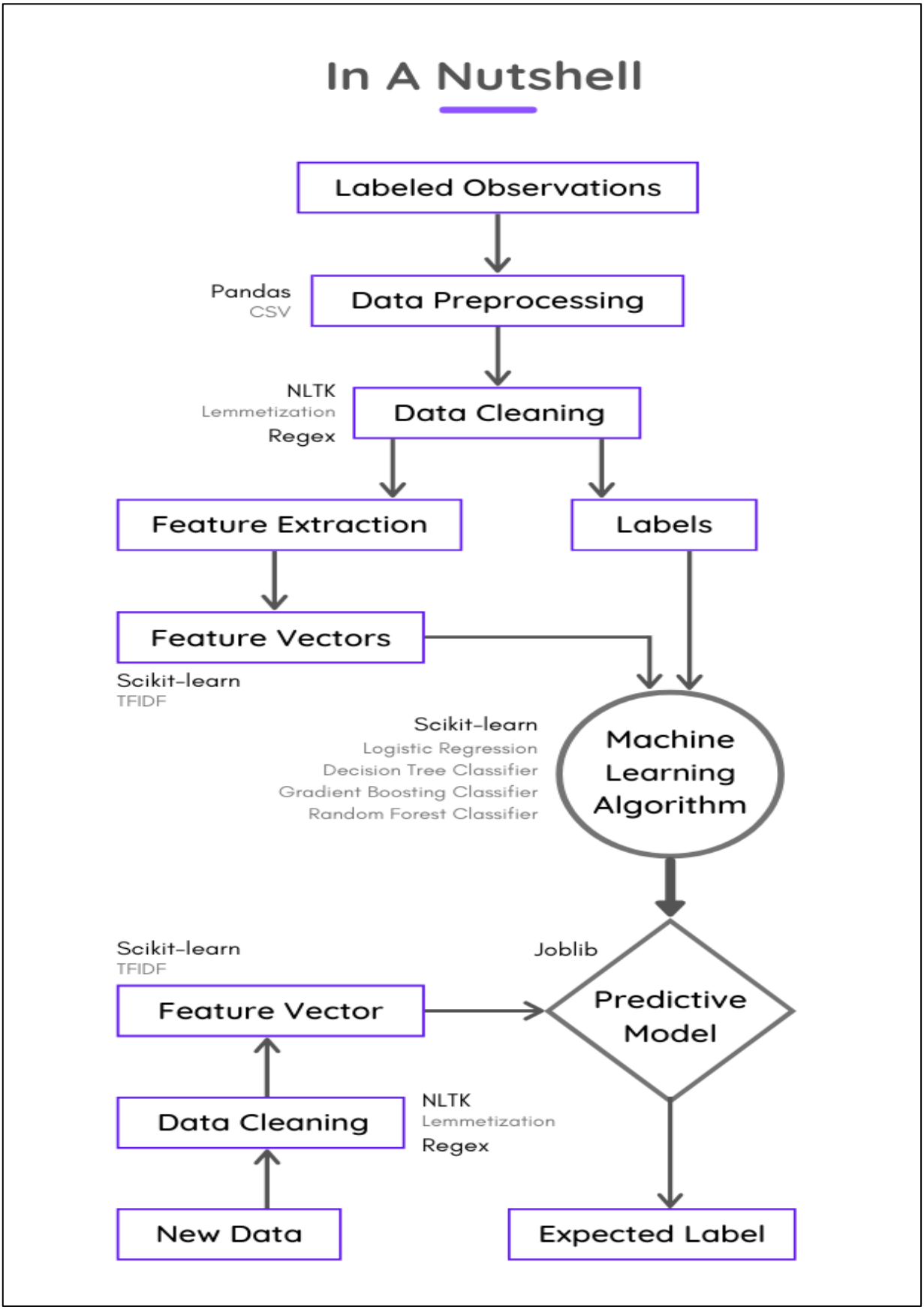


Figure 4: Block Diagram of Fake News Detection

5.4 IMPLEMENTATION

1) Make necessary import.

```
In [1]: import pandas as pd
        import numpy as np
        import re
        import string
        import itertools
        import seaborn as sns
        import matplotlib.pyplot as plt
        import joblib
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score,roc_curve,roc_auc_score
        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score,precision_score,recall_score,f1_score
        from sklearn.utils import shuffle
        from sklearn.pipeline import Pipeline
```

Figure 5: Making Necessary Imports

2) Now, let's read the data into a data frame.

```
In [2]: df_true = pd.read_csv("../dataset/True.csv")
        df_fake = pd.read_csv("../dataset/Fake.csv")
```

Figure 6: Reading the Data Frame

3) And get the shape of the data and the first 5 records.

```
In [3]: df_true.head(5)
```

Out[3]:

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

```
In [4]: df_fake.head(5)
```

Out[4]:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

Figure 7: Getting the first 5 records

4) Cleaning Data

We can't use text data directly because it has some unusable words and special symbols and many more things. If we used it directly without cleaning then it is very hard for the ML algorithm to detect patterns in that text and sometimes it will also generate an error. So that

we have to always first clean text data. In this project, we are making one function ‘wordclean’ which cleans the data. Cleaned text data using Regex and some NLP methods like Lemmatization to get uniform and semantically meaningful text data for the next steps.

```
In [23]: # Data cleaning function
         def wordclean(text):
             cleantext = ""
             text = text.lower()
             #simplifying text
             text=re.sub(r"i'm","i am",text)
             text=re.sub(r"he's","he is",text)
             text=re.sub(r"she's","she is",text)
             text=re.sub(r"that's","that is",text)
             text=re.sub(r"what's","what is",text)
             text=re.sub(r"where's","where is",text)
             text=re.sub(r"\'ll"," will",text)
             text=re.sub(r"\'ve"," have",text)
             text=re.sub(r"\'re"," are",text)
             text=re.sub(r"\'d"," would",text)
             text=re.sub(r"won't","will not",text)
             text=re.sub(r"can't","cannot",text)

             text = re.sub('https?://\S+|www\.\S+','',text) #removes links eg. http://url.com/bla1
             # text = re.sub('\[.*?\]','',text)
             text = re.sub(r'\W',' ',text) #removes non word character
             text = re.sub(r'<.*?>+','',text)
             text = re.sub('[%s]'% re.escape(string.punctuation),'',text) #removes punctuation marks
             text = re.sub(r'\n','',text) #removes new line character
             text = re.sub('\w*\d\w*','',text) #removes word wich contain number eg. 12is, trum5
             text=re.sub(r"\s+"," ",text)

             for word in text.split():
                 if word not in stopwordslist:
                     cleantext+=lemma.lemmatize(word)+" "  # Removing stopwords as well as lemmetizing words
             return cleantext

In [24]: # Cleaning text Data
         df["total"] = df["total"].apply(wordclean)
```

Figure 8: Cleaning the data

5) Feature Extraction & Train-Test splitting

Kept necessary features and labels by dropping unnecessary data fields.
Splitting the data is the most essential step in machine learning. We train our model on the trainset and test our data on the testing set. We split our data in train and test using the train_test_split function from Scikit learn.

```
In [25]: # Defining x and y as feature and label respectively
         x= df['total']
         y = df['label']

In [88]: # Train-Test splitting
         x_train,x_test,y_train,y_test = train_test_split(x,y,test_size =0.33)
```

Figure 9: Feature Extraction & Train-Test splitting

6) Data Vectorizing

Text data can't be directly processed by ML models, So it is necessary to convert news text into vectors. Here TFIDF(Term Frequency Inverse Document Frequency) technique was used to vectorize data.

Tfidf-Vectorizer : (Term Frequency * Inverse Document Frequency)

1.Term Frequency: The number of times a word appears in a document divided by the total number of words in the document. Every document has its own term frequency.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

2. Inverse Document Frequency: The log of the number of documents divided by the number of documents that contain the word w. Inverse data frequency determines the weight of rare words across all documents in the corpus.

$$idf(w) = log(\frac{N}{df_t})$$

Finally Tfidf vectorizer

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

```
In [27]: # Importing TFIDFvectorizer transformer
         from sklearn.feature_extraction.text import TfidfVectorizer

In [28]: TV = TfidfVectorizer() # Intializing TFIDFVectorization object
         xv_train = TV.fit_transform(x_train) # Fitting TV model with train text data
         xv_test = TV.transform(x_test) # Transforming test text data in TV model
```

Figure 10: Data Vectorizing

5.4.1 IMPLEMENTATION USING LOGISTIC REGRESSION

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

For example,
To predict whether an email is spam (1) or (0)
Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

```
In [32]: # Importing Logistic Regression Model
         from sklearn.linear_model import LogisticRegression

In [33]: LR = LogisticRegression(n_jobs=-1) # Intializing Logistic Regression object
         LR.fit(xv_train,y_train) # Fitting model with vectorized text data and binary labels
Out[33]: LogisticRegression(n_jobs=-1)

In [34]: LRpred = LR.predict(xv_test) # Testing data

In [35]: # Generating intutive report on trained Logistic Regression model
         print("-"*60)
         print("FND USING LOGISTIC REGRESSION".center(60))
         print("-"*60)
         get_matrics(y_test,LRpred)
         print("-"*60)
         print("Classification_report".center(60))
         print("-"*60)
         print(classification_report(y_test,LRpred))
         print("-"*60)
         print("ROC".center(60))
         print("-"*60)
         get_ROC(LR,y_test,xv_test)
         print("-"*60)
         cmLR = confusion_matrix(y_test, LRpred)
         plot_confusion_matrix(cmLR,classes=['FAKE', 'REAL'])
         print("-"*60)
```

Figure 11: Implementation using Logistic Regression

Classification Report

To check how well our model we use some metrics to find the accuracy of our model. There are many types of classification metrics available in Scikit learn

- Confusion Matrix
- Accuracy Score
- Precision
- Recall
- F1-Score

Confusion matrix: Basically this metrics how many results are correctly predicted and how many results are not correctly predicted

Accuracy Score: It is the number of correct prediction over the total no. of predictions

```
------------------------------------------------------------
               FND USING LOGISTIC REGRESSION
------------------------------------------------------------
              Accuracy : 0.986919600054787
             Precision : 0.9831176139561058
                Recall : 0.9899419181187137
                    F1 : 0.9865179642831933
------------------------------------------------------------
                  Classification_report
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      7543
           1       0.98      0.99      0.99      7059

    accuracy                           0.99     14602
   macro avg       0.99      0.99      0.99     14602
weighted avg       0.99      0.99      0.99     14602
```

Figure 12: Classification Report of Logistic Regression

As you can see we have a very good score of Precision, Recall, and F1 Score. So we can say our model performs excellently on unseen data. The accuracy score on Test Dataset is 99% which is very good.

Confusion Matrix

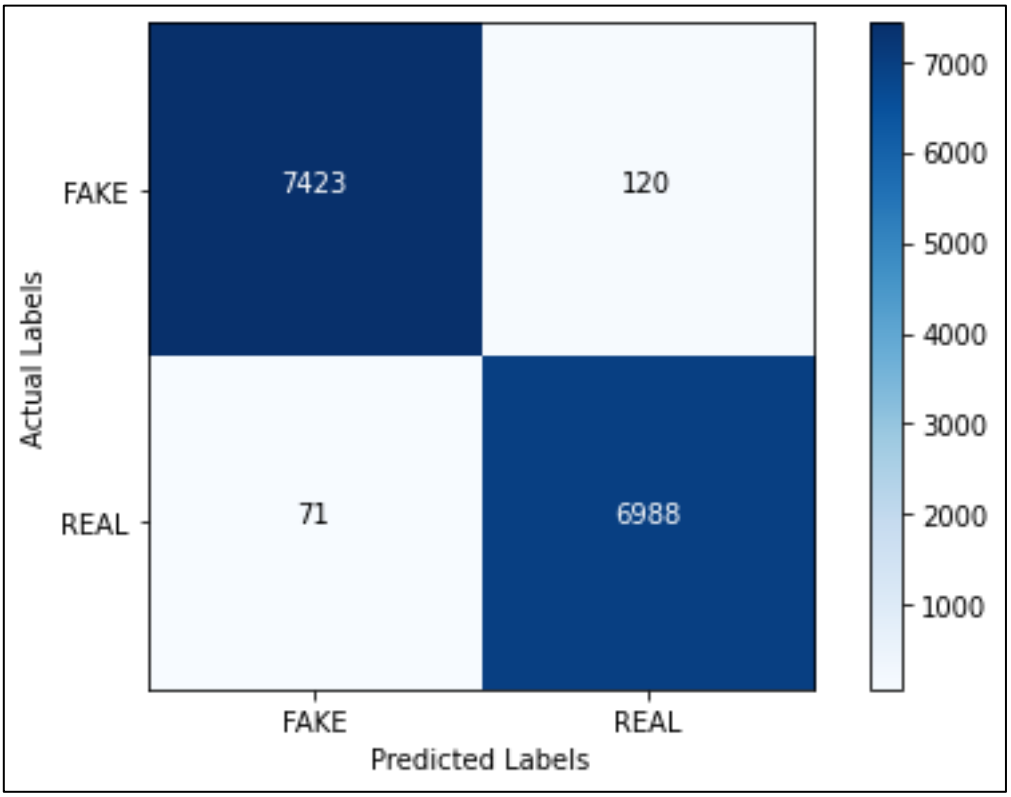


Figure 13: Confusion Matrix of Logistic Regression

5.4.2 IMPLEMENTATION USING DECISION TREE CLASSIFIER

A decision tree is a flowchart-like tree structure where an internal node represents feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

```
In [36]: # Importing Decision Tree Classifier
         from sklearn.tree import DecisionTreeClassifier

In [37]: DTC = DecisionTreeClassifier() # Intializing Decision Tree Classifier object
         DTC.fit(xv_train, y_train) # Fitting model with vectorized text data and binary labels

Out[37]: DecisionTreeClassifier()

In [38]: DTCpred = DTC.predict(xv_test) # Testing data

In [39]: # Generating intutive report on trained Decision Tree Classifier
         print("-"*60)
         print("FND USING DECISION TREE CLASIFIER".center(60))
         print("-"*60)
         get_matrics(y_test,DTCpred)
         print("-"*60)
         print("Classification_report".center(60))
         print("-"*60)
         print(classification_report(y_test,DTCpred))
         print("-"*60)
         print("ROC".center(60))
         print("-"*60)
         get_ROC(DTC,y_test,xv_test)
         print("-"*60)
         cmDTC = confusion_matrix(y_test, DTCpred)
         plot_confusion_matrix(cmDTC,classes=['FAKE', 'REAL'])
         print("-"*60)
```

Figure 14: Implementation using Decision Tree Classifier

Classification Report

```
------------------------------------------------------------
            FND USING DECISION TREE CLASIFIER
------------------------------------------------------------
               Accuracy : 0.9957540063005068
              Precision : 0.9968754438290016
                 Recall : 0.9943334749964584
                     F1 : 0.9956028368794326
------------------------------------------------------------
                  Classification_report
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      7543
           1       1.00      0.99      1.00      7059

    accuracy                           1.00     14602
   macro avg       1.00      1.00      1.00     14602
weighted avg       1.00      1.00      1.00     14602
```

Figure 15: Classification Report of Decision Tree Classifier

Confusion Matrix

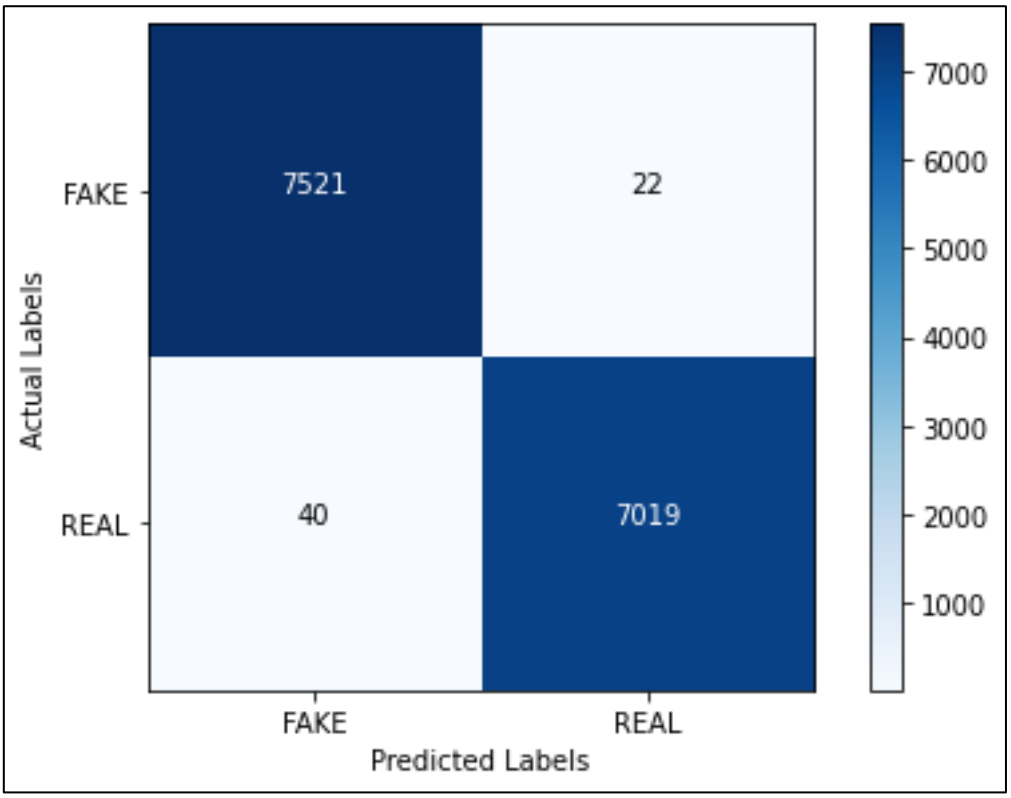


Figure 16: Confusion Matrix of Decision Tree Classifier

5.4.3 IMPLEMENTATION USING GRADIENT BOOSTING CLASSIFIER

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets.

Gradient boosting classifiers are specific types of algorithms that are used for classification tasks, as the name suggests.

Features are the inputs that are given to the machine learning algorithm, the inputs that will be used to calculate an output value. In a mathematical sense, the features of the dataset are the variables used to solve the equation. The other part of the equation is the label or target, which are the classes the instances will be categorized into. Because the labels contain the target values for the machine learning classifier, when training a classifier you should split up the data into training and testing sets. The training set will have targets/labels, while the testing set won't contain these values.

```
In [40]: # Importing Gradient Boosting Classifer
         from sklearn.ensemble import GradientBoostingClassifier

In [41]: GBC = GradientBoostingClassifier(random_state=0,learning_rate=0.001) # Intializing Gradient Boosting Classifer object
         GBC.fit(xv_train, y_train) # Fitting model with vectorized text data and binary labels

Out[41]: GradientBoostingClassifier(learning_rate=0.001, random_state=0)

In [42]: GBCpred = GBC.predict(xv_test) # Testing data

In [43]: # Generating intutive report on trained Gradient Boosting Classifer model
         print("-"*60)
         print("FND USING GRADIENT BOOSTING CLASSIFIER".center(60))
         print("-"*60)
         get_matrics(y_test,GBCpred)
         print("-"*60)
         print("Classification_report".center(60))
         print("-"*60)
         print(classification_report(y_test,GBCpred))
         print("-"*60)
         print("ROC".center(60))
         print("-"*60)
         get_ROC(GBC,y_test,xv_test)
         print("-"*60)
         cmGBC = confusion_matrix(y_test, LRpred)
         plot_confusion_matrix(cmGBC,classes=['FAKE', 'REAL'])
         print("-"*60)
```

Figure 17: Implementation Using Gradient Boosting Classifier

Classification Report

```
------------------------------------------------------------
          FND USING GRADIENT BOOSTING CLASSIFIER
------------------------------------------------------------
               Accuracy : 0.9945897822216134
              Precision : 0.9909960607765897
                 Recall : 0.9978750531236719
                     F1 : 0.9944236606197501
------------------------------------------------------------
                  Classification_report
------------------------------------------------------------
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      7543
           1       0.99      1.00      0.99      7059

    accuracy                           0.99     14602
   macro avg       0.99      0.99      0.99     14602
weighted avg       0.99      0.99      0.99     14602
```

Figure 18: Classification Report of Gradient Boosting Classifier

Confusion Matrix

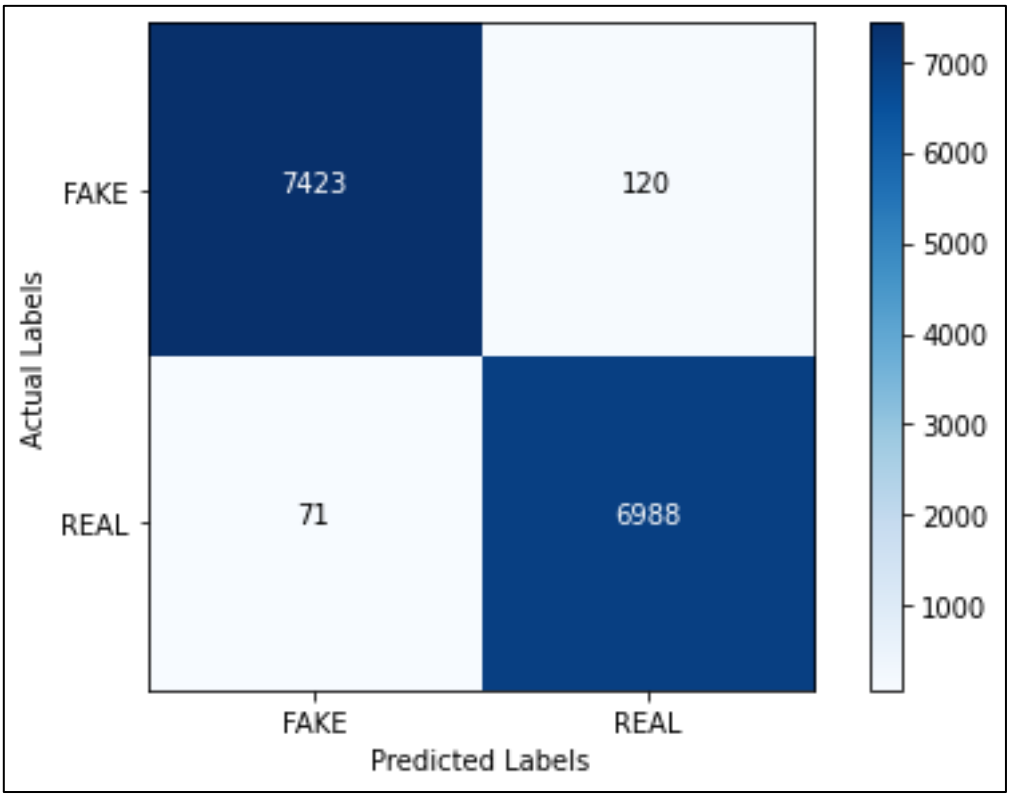


Figure 19: Confusion Matrix of Gradient Boosting Classifier

5.4.4 IMPLEMENTATION USING RANDOM FOREST CLASSIFIER

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:
A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

```
In [44]: # Importing Random Forest Classifier
         from sklearn.ensemble import RandomForestClassifier

In [45]: RFC = RandomForestClassifier(random_state=0,n_jobs=-1) # Intializing Random Forest Classifier object
         RFC.fit(xv_train, y_train) # Fitting model with vectorized text data and binary labels

Out[45]: RandomForestClassifier(n_jobs=-1, random_state=0)

In [46]: RFCpred = RFC.predict(xv_test) # Testing data

In [47]: # Generating intutive report on trained Random Forest Classifier
         print("-"*60)
         print("FND USING RANDOM FOREST CLASSIFIER".center(60))
         print("-"*60)
         get_matrics(y_test,RFCpred)
         print("-"*60)
         print("Classification_report".center(60))
         print("-"*60)
         print(classification_report(y_test,RFCpred))
         print("-"*60)
         print("ROC".center(60))
         print("-"*60)
         get_ROC(RFC,y_test,xv_test)
         print("-"*60)
         cmRFC = confusion_matrix(y_test, RFCpred)
         plot_confusion_matrix(cmRFC,classes=['FAKE', 'REAL'])
         print("-"*60)
```

Figure 20: Implementation using Random Forest Classifier

Classification Report

```
------------------------------------------------------------
            FND USING RANDOM FOREST CLASSIFIER
------------------------------------------------------------
               Accuracy : 0.9914395288316669
              Precision : 0.98982763492512
                 Recall : 0.9924918543703074
                     F1 : 0.9911579543043078
------------------------------------------------------------
                  Classification_report
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7543
           1       0.99      0.99      0.99      7059

    accuracy                           0.99     14602
   macro avg       0.99      0.99      0.99     14602
weighted avg       0.99      0.99      0.99     14602
```

Figure 21: Classification Report of Random Forest Classifier

Confusion Matrix

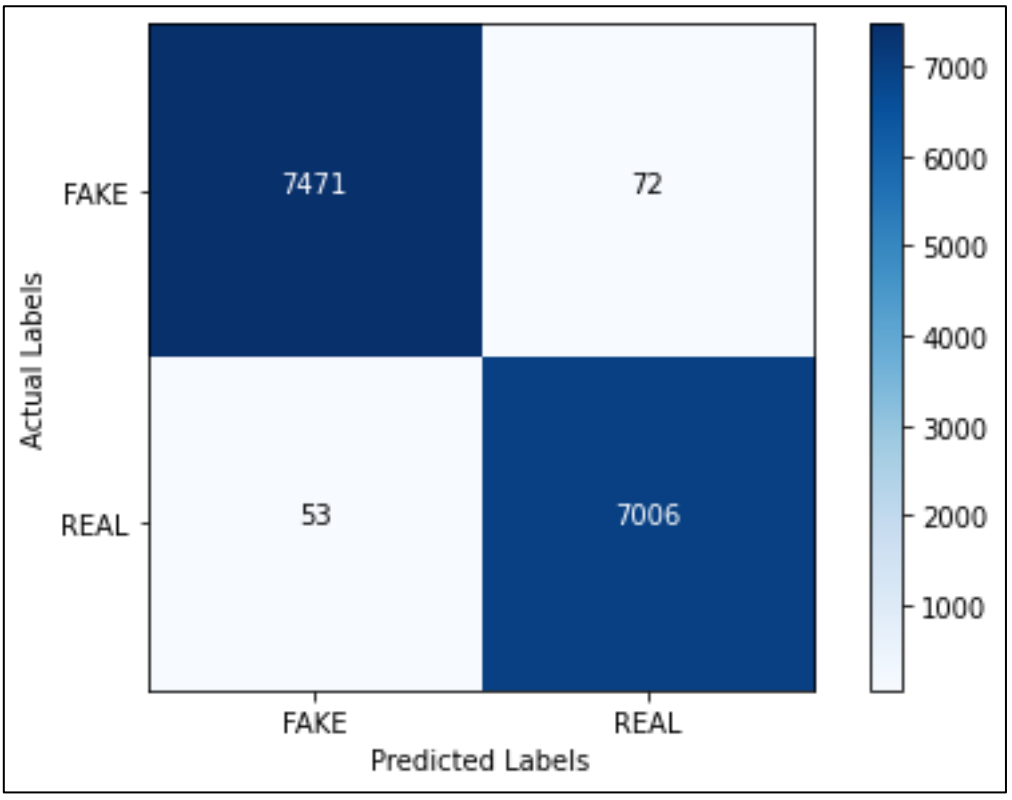


Figure 22: Confusion Matrix of Random Forest Classifier

5.5 MATRICS OF IMPLEMENTED MODELS

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.9865	0.9846	0.9873	0.9859
Decision Tree Classifier	0.9960	0.9970	0.9948	0.9959
Gradient Boosting Classifier	0.9950	0.9915	0.9983	0.9948
Random Forest Classifier	0.9917	0.9930	0.9897	0.9913

CHAPTER 6: FINAL PRODUCT

6.1 OVERVIEW

Website – URL: https://fake-news-detection-using-ml.herokuapp.com

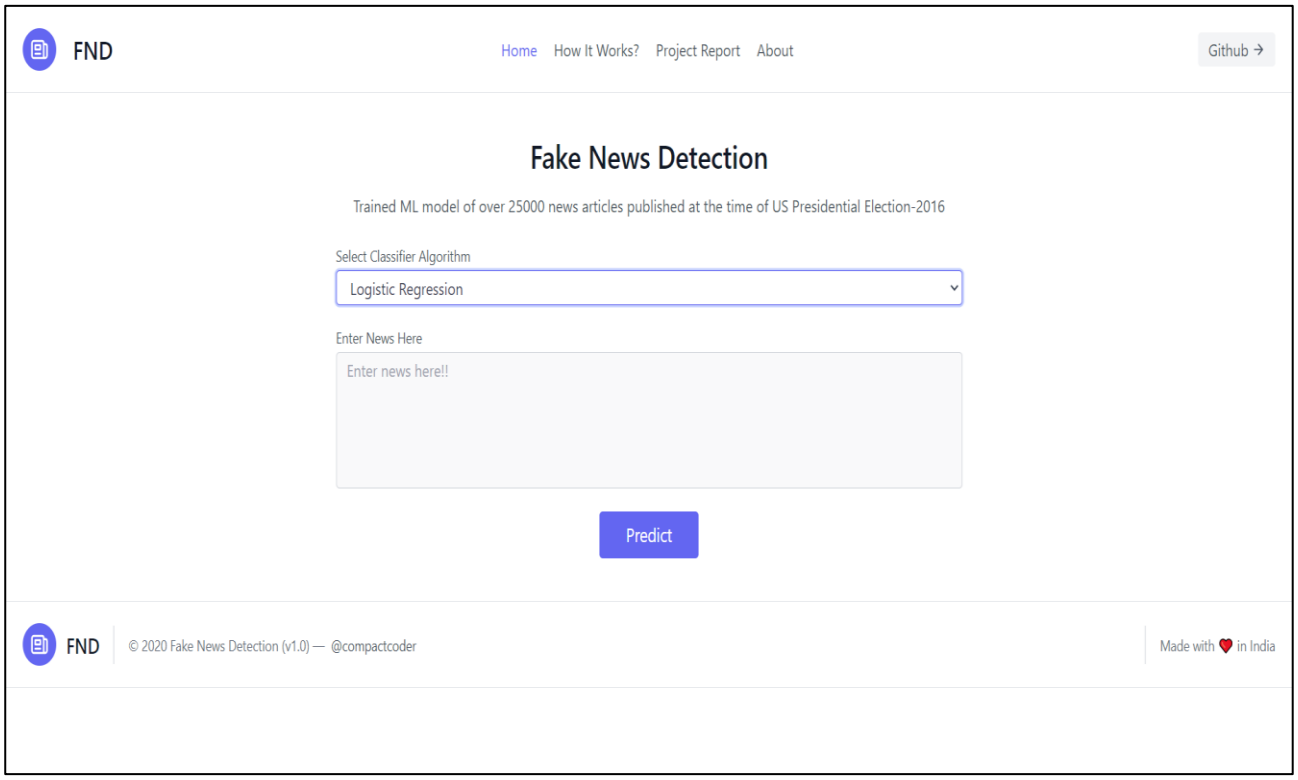


Figure 23: Fake News Detection using Machine Learning Web-App

6.2 TECHNOLOGIES USED

6.2.1 HTML

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

6.2.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

6.2.2.1 TAILWIND CSS

According to the official documentation, Tailwind CSS is a utility-first CSS framework for rapidly building custom user interfaces. It is a cool way to write inline styling and achieve an awesome interface without writing a single line of your own CSS.

Tailwind isn’t the first utility CSS library, but it is the most popular at the moment.

6.2.3 FLASK

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Features

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

6.2.4 DEPLOYMENT

6.2.4.1 HEROKU

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages. Heroku was acquired by Salesforce.com in 2010 for $212 million.

Prerequisite:

1. You must have installed Git in your system.
2. You must have installed Python in your system.

Step-1: Install Heroku CLI

```
% brew tap heroku/brew && brew install heroku
```

Step-2: Create Python Virtual Environment

```
% python3 -m venv foldername
% source foldername/bin/activate
% cd foldername
```

Step-3: Install Flask & Gunicorn

```
% pip3 install flask gunicorn
```

Step-4: Create an app folder and simple python app

```
% mkdir app
% cd app
% vi main.py
```

main.py

```
from flask import Flask
app= Flask(__name__)
@app.route('/')
def index():
  return "<h1>Welcome to CodingX</h1>"
```

Step-5: Create an entry point to the application, wsgi.py

```
% cd ../
% vi wsgi.py
```

wsgi.py

```
from app.main import app
if __name__ == "__main__":
  app.run()
```

Step-6: Run the application in your local system

```
% python wsgi.py
```

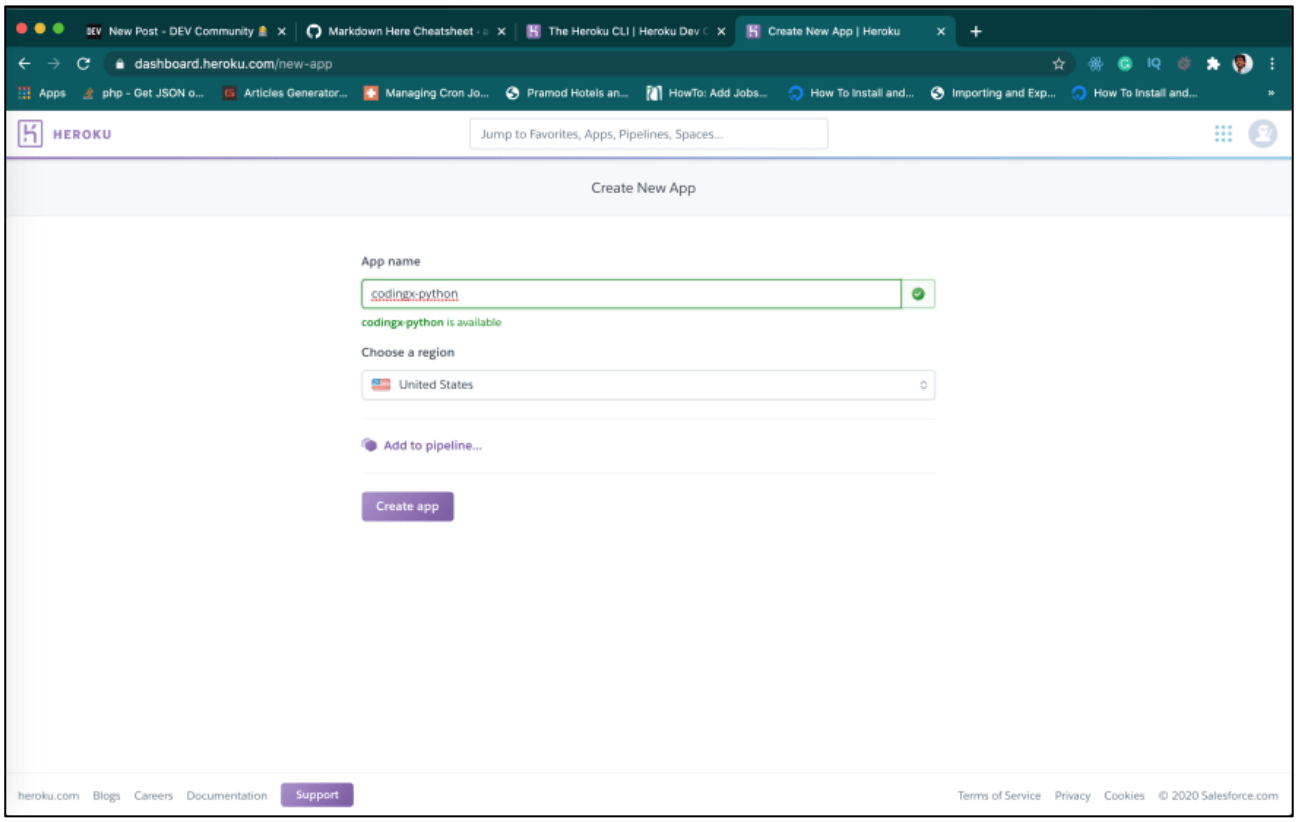
Step-7: Create requirements.txt and Procfile file

```
% pip3 freeze
% pip3 freeze > requirements.txt
% vi Procfile
```

Procfile

```
web: gunicorn wsgi:app
```

Step-8: Create an app in Heroku



Step-9: Deploy your app to Heroku

```
% heroku login
% git init
% heroku git:remote -a codingx-python
% git add.
% git commit -am "First python app"
% git push heroku master
```

6.2.4.2 GITHUB

GitHub URL: https://github.com/compactcoder

To be very crisp about what exactly is GitHub, it is a file or code-sharing service to collaborate with different people.

GitHub is a highly used software that is typically used for version control. It is helpful when more than just one person is working on a project. Say for example, a software developer team wants to build a website and everyone has to update their codes simultaneously while working on the project. In this case, GitHub helps them to build a centralized repository where everyone can upload, edit, and manage the code files.

Why is GitHub so popular?

GitHub has various advantages but many people often have a doubt as to why not use dropbox or any cloud based system? Let me take the same example forward to answer this question. Say more than two software developers are working on the same file and they want to update it simultaneously. Unfortunately, the person who save the file first will get precedence over the others. While in GitHub, this is not the case. GitHub document the changes and reflect them in an organized manner to avoid any chaos between any of the files uploaded.

Therefore using GitHub centralized repository, it avoids all the confusion and working on the same code becomes very easy.

CHAPTER 7: REFERENCES

[1] https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/

[2] https://www.pantechsolutions.net/fake-news-detection-using-machine-learning

[3] https://opendatascience.com/how-to-build-a-fake-news-classification-model/

[4] https://matheo.uliege.be/bitstream/2268.2/8416/1/s134450_fake_news_detection_using_ machine_learning.pdf

[5] https://dspace.mit.edu/bitstream/handle/1721.1/119727/1078649610-MIT.pdf

[6] https://towardsdatascience.com/detecting-fake-news-with-and-without-code- dd330ed449d9

[7] https://en.wikipedia.org/wiki/Detecting_fake_news_online

[8] https://www.tutorialspoint.com/natural_language_processing/index.htm

[9] https://www.upgrad.com/blog/artificial-neural-networks-data-mining/

[10] https://www.kaggle.com/c/fake-news/data

[11] https://github.com/krishnaik06/Fake-News-Classifier

[12] https://www.geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/

[13] https://www.freepatentsonline.com/y2016/0164888.html

[14] Fake News and Cyber Propaganda: A Study of Manipulation and Abuses on Social Media (July 2018 In book: Mediascape in 21st Century (pp.535-544) Publisher: Kanishka Publisher)