A Report on:

“News Credibility Detection : Fake or Nor Fake”

Prepared by : Bhagya Vinod Rana

Roll. No. : U19CS012

Class : B.Tech –III (Computer Science & Engineering) 5th Semester

Year : 2021-22

Guided by : Shivangi Modi Mam

Department of Computer Engineering

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

Sardar Vallabhbhai National Institute of Technology,

Surat -395007 (Gujarat), India

CERTIFICATE

This is to certify that the Seminar Report entitled

News Credibility Detection : Fake or Nor Fake

is prepared and presented by **Mr. Bhagya Vinod Rana** bearing Roll No. : **U19CS012**, 3rd Year of **B.Tech (Computer Science and Engineering)** and his work is satisfactory.

GUIDE	**JURY**	**HOD**
Shivangi Modi		**COED**

Contents

Abstract

Consumption of social media information to stay up with world news and verify its veracity has become a significant concern in recent years. While social media allows us to readily access news from anywhere at any time, it also facilitates the propagation of fake news, resulting in the dissemination of misleading information. This has a bad effect on society as well. As a result, it is vital to establish whether the news being circulated on social media is true. This will prevent social media users from becoming confused, which is critical for good social growth.In this report, we aim to perform a binary classification of various news articles available online with the help of concepts pertaining to Artificial Intelligence, Natural Language Processing and Machine Learning.

Chapter 1

Introduction

Fake news is an **Infodemic**, a disease worse than anything else we've ever seen. And it's been around for longer than Covid19. Sometimes, fake news has little impact. But when times are uncertain, and a global crisis is in effect, people look for information that alleviates their fear. Unfortunately, fear leaves people more susceptible to accepting **misleading information** as the real deal. A limited amount of reliable data in some instances also encourages people to take misinformation willingly.

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for **information dissemination** that has never been witnessed in the human history before. With the current usage of social media platforms, consumers are creating and sharing more information than ever before, some of which are misleading with **no relevance to reality**.

The Internet has boosted the spread of fake news from a single place to all over the world. Social media is free to use, and therefore, **Anyone** can post a story that goes viral. While most articles are manually written, innovative fake news creators use **Natural Language Generation** techniques to create excellent, realistic counterfeit ones. These models have created an urgent need to distinguish fake from real news and detect human-generated and machine-generated fake news.

With the outburst of information, it is seemingly **tedious** for a layman to distinguish whether the news he consumes is real or fake. Fake news is typically published with an intent to mislead or create bias to acquire **political or financial gains**. Hence it may tend to have luring headlines or interesting content to increase viewership.

1.1 Characteristics of Fake News

- They often have **grammatical mistakes**.
- They are often **emotionally coloured**. They often try to affect readers‘ opinion on some topics. Their content is not always true.
- They often use **attention seeking words** and news format and click baits.
- They are **too good to be true**.
- Their **sources** are **not genuine** most of the times

Chapter 2

Why Data Science?

2.1 Domain Knowledge Requirement

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. Data Science can be used to Extract Useful Features [Feature Engineering] that will help to Classify Fake News.

2.2 Vast Volume of Data

With Very Large Volume of Data on social media platforms (such as Facebook and Twitter), online news platforms, blogs, social media feeds, and other digital media formats, No Human Force can manually identify Fake News. Therefore, Data Science comes for Rescue, since it is known for Handling Large Amount of Data [Big Data] and processing it efficiently.

2.3 Rate of Data Production is Very High

The Pace at which News, Articles, Posts, Tweets and Information shared on Various Social Media is Rapid and Needs to be Classified as Fake or Genuine, Before the News Reaches to Larger Audience, which may Lead to Panic among the People. We don't have much time to Process the Data and Classify the Given Text to be Genuine or Not. So, Real Time Classification at Source is required.

2.4 News is Not Always in Structured Form

News can be presented in Form of images [News to be Extracted using OCR] or Video or Audio Form or it might be obfuscated [Hidden]. Thus, News can be considered as Semi-Structured or Unstructured Data and therefore Data Science Plays a Key Role in its Analysis.

2.5 The Process Needs Automation

The Classification Task is Repetitive and Boring [Learning from Fake News and Finding the Similarity Features in Current One], So We need Data Science to create smarter systems that can take autonomous decisions based on historical fake news datasets.

Chapter 3

Data Science Approach

3.1 Understanding Problem Statement

Given the social news engagements **E** among n users for news article **a**, the task of fake news detection is to predict whether the news article a is a fake news piece or not, such that,

$$\mathcal{F}(a) = \begin{cases} 1, & \text{if } a \text{ is a piece of fake news,} \\ 0, & \text{otherwise.} \end{cases}$$

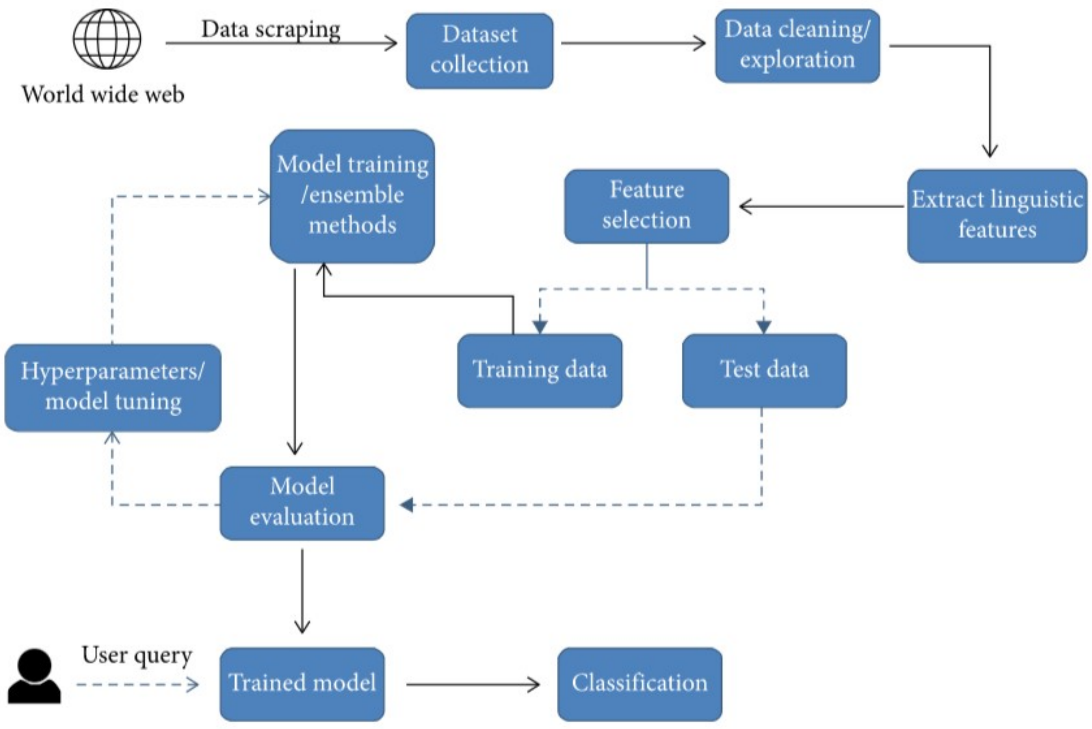
where **F** is the prediction function we want to learn.

We can observe that Fake News Detection is **Binary Classification** Problem.

Next, we propose a general data mining framework for fake news detection which includes two phases:

1. feature extraction
2. model construction

The **feature extraction** phase aims to represent news content and related auxiliary information in a formal mathematical structure, and **model construction** phase further builds machine learning models to better differentiate fake news and real news based on the feature representations.



3.2 Data Mining

We can get online news from different sources like social media websites, search engine, homepage of news agency websites or the fact-checking websites. On the Internet, there are a few publicly available datasets for Fake news classification like Buzzfeed News, LIAR, BS Detector etc. These datasets have been widely used in different research papers for determining the veracity of news.

Online news can be collected from different sources, such as news agency homepages, search engines, and social media websites. However, manually determining the veracity of news is a challenging task, usually requiring annotators with domain expertise who performs careful analysis of claims and additional evidence, context, and reports from authoritative sources.

Generally, news data with annotations can be gathered in the following ways: Expert journalists, Fact-checking websites, Industry detectors, and Crowd sourced workers. However, there are no agreed upon benchmark datasets for the fake news detection problem. Data gathered must be pre-processed that is, cleaned, transformed and integrated before it can undergo training process.

3.3 Data Cleaning

Social media data is highly unstructured majority of them are informal communication with typos, slangs and bad-grammar etc. Quest for increased performance and reliability has made it imperative to develop techniques for utilization of resources to make informed decisions. To achieve better insights, it is necessary to clean the data before it can be used for predictive modelling. For this purpose, basic pre- processing was done on the News training data.

Cleaning (or pre- processing) the data typically consists of a number of steps:

- **Remove punctuation**
  - Punctuation can provide grammatical context to a sentence which supports our understanding. But for our vectorizer which counts the number of words and not the context, it does not add value, so we remove all special characters.
    Eg: How are you? -> How are you
- **Tokenization**
  - Tokenizing separates text into units such as sentences or words. It gives structure to previously unstructured text. Eg: Plata o Plomo -> Plata,o,Plomo.
- **Remove stopwords**
  - Stopwords are common words that will likely appear in any text. They don't tell us much about our data so we remove them. Eg: silver or lead is fine for me-> silver, lead, fine.
- **Stemming**
  - Stemming helps reduce a word to its stem form. It often makes sense to treat related words in the same way. It removes suffices, like ing, ly, s, etc. by a simple rule-based approach. It reduces the corpus of words but often the actual words get neglected. Eg: Entitling, Entitled -> Entitle. Note: Some search engines treat words with the same stem as synonyms

Dataset : https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

Each sample in the train and test set has the following information:

- The title of the new article.
- The text of the new article against each title.
- The subject for the news article.
- Date of the new article

I am predicting whether a given news is about a real news or not. If so, predict a 1. If not, predict a 0.

Data Pre Processing

Importing all the required libraries

```
#Fundamental pre processing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Lib for NLP processing
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
import re
import nltk
nltk.download('stopwords')
```

Downloading the dataset from Kaggle

```
# Upload kaggle.jason
# please follow this link incase not aware: https://www.kaggle.com/general
    /74235
from google.colab import files
files.upload()

! pip install opendatasets --upgrade
import opendatasets as od

```

```
dataset_url = 'https://www.kaggle.com/clmentbisaillon/fake-and-real-news-
    dataset'
od.download(dataset_url)
```

3.4 Data Exploration and Feature Extraction

We can use text data to generate a number of features like word count, frequency of large words, frequency of unique words, n-grams etc. By creating a representation of words that capture their meanings, semantic relationships, and numerous types of context they are used in, we can enable computer to understand text and perform Clustering, Classification etc.

- Bag-Of-Words
  - Bag of Words (BoW) or CountVectorizer describes the presence of words within the text data. It gives a result of 1 if present in the sentence and 0 if not present. It, therefore, creates a bag of words with a document matrix count in each text document.
- TF-IDF
  - It computes relative frequency that a word appears in a document compared to its frequency across all documents TF-IDF weight represents the relative importance of a term in the document and entire corpus.
  - TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.
  - It is Used for search engine scoring, text summarization, document clustering.

So now we will go through an exploratory data analysis to get insights from the news article. The aim here is to divide this session into topics so we can explore graphics for each subject.

Labels distribution

Size of dataset downloaded

```
print(real_news.shape)
print(fake_news.shape)
```

We add a column each in both the dataset to identify the real and fake new articles. We denote real news as 1 and fake with 0

```
real_news['FakeOrNot'] = 1
fake_news['FakeOrNot'] = 0
```

Now we combine both the individual dataset so that we can analyses the complete dataset

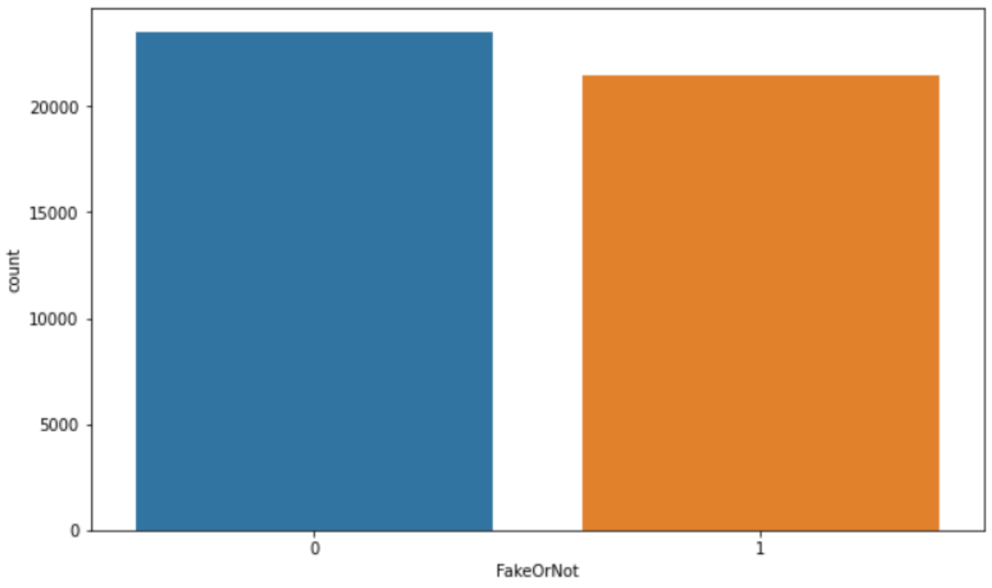
```
News_Dataset = pd.concat([real_news, fake_news], ignore_index=True)
News_Dataset.head()
```

As we are running the analysis with the news titles hence the remaining columns are not required.

```
News_Dataset = News_Dataset.drop(['text', 'subject', 'date'], axis = 1)
News_Dataset
```

```
News_Dataset.FakeOrNot.value_counts()
```

```
fig, ax = plt.subplots(figsize=(10, 6))
sns.countplot(data = News_Dataset, x='FakeOrNot')
plt.show()
```



The labels seems to be evenly distributed. This is a good sign and confirms that the dataset is not biased.

Cleaning, Formatting and Lemmatization

Before processing the text let us check if any row/columns are having null values

```
News_Dataset[News_Dataset.isnull().any(axis=1)]
```

Let are now remove all the string punctuation, We can achieve this by simple keeping the work from [a-z] and [A-Z] and replacing the rest of the words with space.

Also, let us lower all the text so when stemming/lemmatization is applied words spell in capitals are not treated differently with the same words present in small letters. We apply stopwords to safely ignore the meaningless words without sacrificing the meaning of the sentences.

In the end we implement Lemmatization for converting a word to its base form.

```
nltk.download('wordnet')
wordnet=WordNetLemmatizer()
corpus = []

for i in range(0, len(News_Dataset)):
    review = re.sub('[^a-zA-Z]', ' ', News_Dataset['title'][i])
    review = review.lower()
    review = review.split()

    review = [wordnet.lemmatize(word) for word in review if not word in
    stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

3.5 Training and Validation

```
## Applying Countvectorizer
# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,ngram_range=(2,2))
X = cv.fit_transform(corpus).toarray()
```

We split the data into training and validation set

```
X.shape
y=News_Dataset['FakeOrNot']
## Divide the dataset into Train and Test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
    random_state=0)
```

```python
cv.get_feature_names()[:20]

count_df = pd.DataFrame(X_train, columns=cv.get_feature_names())
count_df.head()
```

3.6 Predictive Modelling

Applying ML Models using Sklearn (Bag of words)

Multinomial models are more suited for processing text related features extracted using Bag of words

```python
from sklearn.naive_bayes import MultinomialNB
classifier=MultinomialNB()
```

```python
from sklearn import metrics
import numpy as np
import itertools
```

```python
classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
```

accuracy: 0.809

3.7 Data Visualization

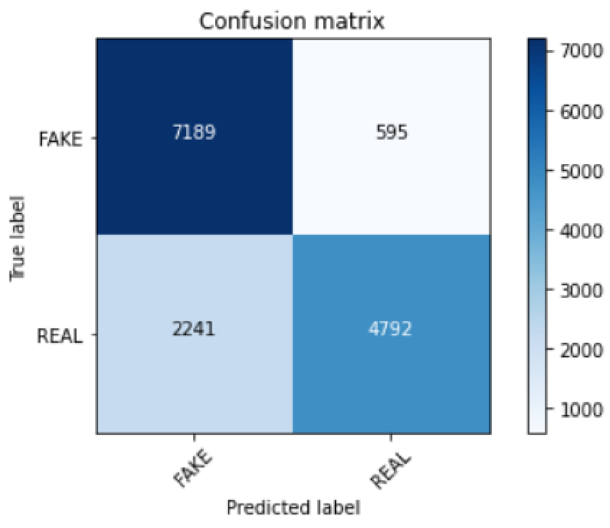
Evaluating Results(Bag of Words)

We write the below functions to build the confusion matrix visualizations.

```python
import matplotlib.pyplot as plt
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    See full source and example:
    http://scikit-learn.org/stable/auto_examples/model_selection/
   plot_confusion_matrix.html
```

```python

    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

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

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

cm = metrics.confusion_matrix(y_test, pred)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix, without normalization
```



```python
classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
score = metrics.accuracy_score(y_test, pred)
score
```

0.8085982317608152

with bag of words we achieved maximum accuracy of **81 percent**.

Implementing hyper parameterization (Bag of Words)

Multinomial Classifier with Hyperparameter

```python
classifier=MultinomialNB(alpha=0.1)



previous_score=0
for alpha in np.arange(0,1,0.1):
    sub_classifier=MultinomialNB(alpha=alpha)
    sub_classifier.fit(X_train,y_train)
    y_pred=sub_classifier.predict(X_test)
    score = metrics.accuracy_score(y_test, y_pred)
    if score>previous_score:
        classifier=sub_classifier
    print("Alpha: {}, Score : {}".format(alpha,score))
```

Still Maximum Accurary with Bag of Words is 81 percent,

Implementing TF-IDF

Implementing TFIDF in the corpus extracted eariler

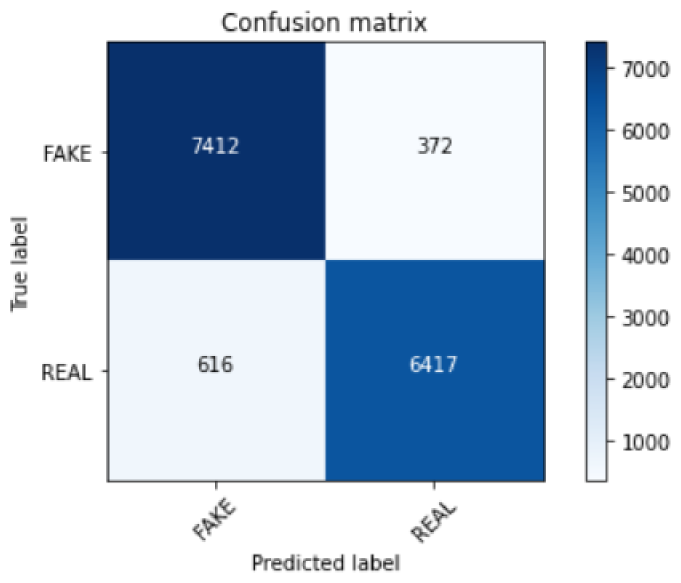
```
# Creating the TF-IDF model
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()
X = cv.fit_transform(corpus).toarray()

X.shape
```

```
count_df = pd.DataFrame(X_train, columns=cv.get_feature_names())

count_df.head()
```

Applying ML Models using SKlearn Evaluating Results (TF IDF)

```
from sklearn.naive_bayes import MultinomialNB
classifier=MultinomialNB()

classifier.fit(X_train, y_train)
pred = classifier.predict(X_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
cm = metrics.confusion_matrix(y_test, pred)
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
accuracy:   0.933
Confusion matrix, without normalization
```

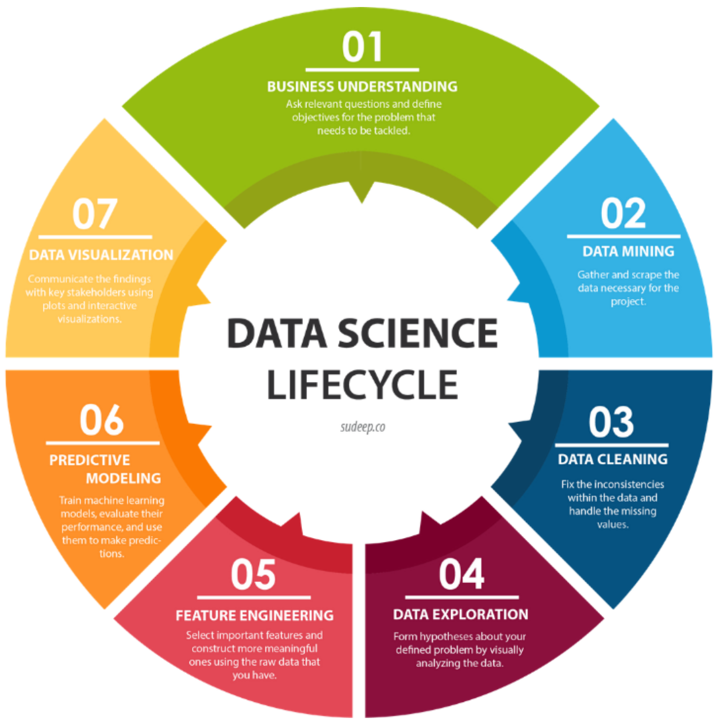


Implementing hyper parameterization(TF IDF)

Multinomial Classifier with Hyperparameter

```python
classifier=MultinomialNB(alpha=0.1)

previous_score=0
for alpha in np.arange(0,1,0.1):
    sub_classifier=MultinomialNB(alpha=alpha)
    sub_classifier.fit(X_train,y_train)
    y_pred=sub_classifier.predict(X_test)
    score = metrics.accuracy_score(y_test, y_pred)
    if score>previous_score:
        classifier=sub_classifier
    print("Alpha: {}, Score : {}".format(alpha,score))
```

With TF IDF we are able to achieve an accuracy on **93 Percent**.


DATA SCIENCE LIFECYCLE
sudeep.co
01 BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02 DATA MINING
Gather and scrape the data necessary for the project.
03 DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04 DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05 FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06 PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07 DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.

Chapter 4

Summary

- We downloaded the Fake News Dataset from Kaggle.
- We performed the NLP preprocess and EDA to understand the labels distribution.
- We trained the model using both Bag of words and TF IDF.
- Implemented hyperparameters to achieve maximum accuracy.
- We analyzed the accuracy, For TFIDF we achieved the accuracy of 93 percent and for Bag of words it is 80 percent and conclude **TF IDF** have performed better than bag of words.
- We have Finally Understood and Implemented all the Steps in **Data Science Life Cycle** to Detect Fake News in Real Life.

Chapter 5

Future Scope

- Researchers can further analyze and compare several **language generation models** with the **human writing style** and inter-model styles. It will increase their scope and help with better fake news detection.
- Even though **TF-IDF classifiers** worked well, there are possibilities of exploring other features to improve the model and make it a generic fit.
- While this Method focused on text-based news articles and language models, AI algorithms can also **Analyze other features** such as images, videos, date and time, sources, website, and domain for valuable information.
- Teaching the detection model to **trace the source** of a machine-generated fake news article to the language model from which it originated will be a huge step forward. This development will ensure mitigating and blocking the spread of misinformation.
- In order to reduce the spread of fake news, identifying **key elements** involved in the spread of news is an important step. **Graph theory** and **machine learning techniques** can be employed to identify the key sources involved in spread of fake news.
- **Real time fake news** identification in **videos** can be another possible future direction.