



**Symbiosis Institute of Technology, Pune**

Project Report

of

**“Fake News Detection”**

**Submitted By**

**Vibhor Bhargava**

# **Index**

S.No.	Topics	Page Number
1.	Problem Statement	2
2.	Introduction	2
3.	Motivation/Significance	2 - 3
4.	Scope/Limitations	3 - 4
5.	Gantt Chart	4
6.	Time and Size Estimation	4 - 9
7.	Requirement Analysis	9 - 10
8.	Data Flow Diagrams	11 - 12
9.	Use Cases	13 - 16
10.	Use Case Diagram	17
11.	Flow Diagram	18
12.	ER Diagram	19
13.	Methodologies Used	20 - 21
14.	Software and Hardware Platforms Used	21
15.	Online Tools Used	22
16.	Code Snippets & Outputs	22 - 30
17.	Testing Methodology	30 - 40
18.	Beneficiaries	41

## **1) Problem Statement**

The problem statement is to develop a machine learning program to identify when a news source may be producing fake news. The project aims to use a corpus of labeled real and fake news articles to build a classifier that can make decisions about information based on the content from the corpus and predict whether the news is real or fake.

## **2) Introduction**

Since an increasing amount of our lives is spent interacting online through social media platforms, more and more people tend to seek out and consume news from social media rather than traditional news organizations. The reasons for this change in consumption behaviors are inherent in the nature of these social media platforms:

- (i) It is often more timely and less expensive to consume news on social media compared with traditional news media, such as newspapers or television.
- (ii) It is easier to further share, comment on, and discuss the news with friends or other readers on social media.

Despite the advantages provided by social media, the quality of news on social media is lower than traditional news organizations. However, because it is cheap to provide news online and much faster and easier to disseminate through social media, large volumes of fake news, i.e., those news articles with intentionally false information, are produced online for a variety of purposes, such as financial and political gain.

## **3) Motivation/Significance**

The extensive spread of fake news can have a serious negative impact on individuals and society. First, fake news can break the authenticity balance of the

news ecosystem. For example, it is evident that the most popular fake news was even more widely spread on Facebook than the most popular authentic mainstream news during the U.S. 2016 president election. Second, fake news intentionally persuades consumers to accept biased or false beliefs. Fake news is usually manipulated by propagandists to convey political messages or influence. For example, some report shows that Russia has created fake accounts and social bots to spread false stories. Third, fake news changes the way people interpret and respond to real news. For example, some fake news was just created to trigger people's distrust and make them confused, impeding their abilities to differentiate what is true from what is not. To help mitigate the negative effects caused by fake news—both to benefit the public and the news ecosystem, it is critical that we develop methods to automatically detect fake news on social media.

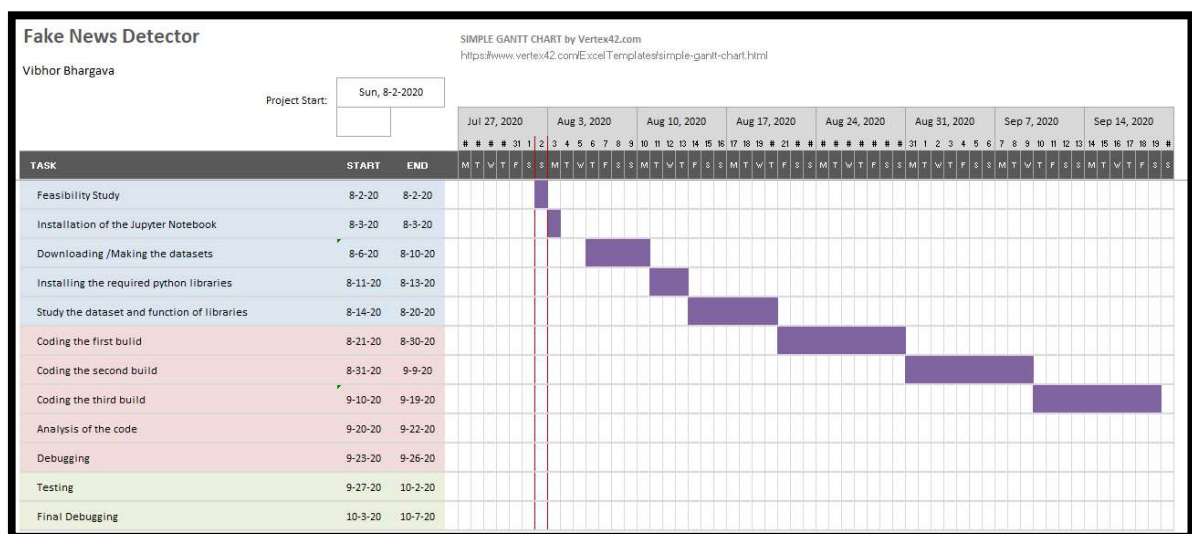
#### **4) Scope/Limitations**

Through the work done in this project, it is shown that machine learning certainly does have the capacity to pick up on language patterns that may be difficult for humans to pick up on. The next steps involved in this project come in three different aspects. The first of aspect that could be improved in this project is augmenting and increasing the size of the dataset. We feel that more data would be beneficial in ridding the model of any bias based on specific patterns in the source. There is also question as to whether or not the size of our dataset is sufficient. The second aspect in which this project could be expanded is by comparing it to humans performing the same task. Comparing the accuracies would be beneficial in deciding whether or not the dataset is representative of how difficult the task of separating fake from real news is. If humans are more accurate than the model, it may mean that we need to choose more deceptive fake news examples. Because we acknowledge that this is only one tool in a toolbox that would really be required for an end-to-end system for classifying fake news, we expect that its accuracy will never reach perfect. However, it may

be beneficial as a stand-alone application if its accuracy is already higher than human accuracy at the same task.

Since this project is not in the stage of deployment, but using various technologies and platforms the project can be deployed and may become beneficial for many individuals and organizations.

## 5) Gantt Chart



## 6) Time and Size Estimation

### a) Time Estimation:

Time required to complete and deploy a software engineering depends upon various factors like:

- Project's complexity
- Time needed in debugging the code
- Time needed to prepare a proper documentation
- Time taken for risk analysis (risk buffer)
- Time eaters which include team meetings, productivity drops etc.

To estimate the time for my project I will be using the organic mode of the COCOMO i.e. Cost Constructive Model. The reason for choosing this mode was that only member is doing all the tasks of the project i.e. me.

COCOMO is a regression model based on LOC, i.e. **number of Lines of Code**. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

The key parameters which define the quality of any software products, which are an outcome of the COCOMO are primarily Effort & Schedule:

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

**Effort :  $a(KLOC)^b$  , where a and b are constants**

Since my project mode is organic , therefore

$$a = 2.4$$

$$b = 1.05$$

My code would be approximately of 1000 – 1200 lines, so I have considered

$$KLOC = 1.1 \text{ (average amount)}$$

$$\text{So, } E = 2.4(1.1)^{1.05}$$

$$E = 2.4(1.105) = 2.652$$

**Effort(E) = 2.652 person-months**

Schedule or Development Time(D) =  $c(E)^d$

For organic type,  $c = 2.5$  and  $d = 0.38$

So,  $D = 2.5(2.652)^{0.38}$

$D = 2.5(1.429)$

**Development Time(D) = 3.57 months(approx.)**

Gathering all the information from above we can say:

Project's duration = Development Time(D) +  $D \times (\text{Risk Buffer})$

It is feasible to consider 15% time for risk analysis and debugging

Hence,

Project's duration =  $3.57 + 3.57(0.15)$

Project's duration =  $3.57 + 0.5355$

**Project's duration = 4.105 months (approx.)**

Since I have started working on my project since mid July I am considering that my project will finish approx. by mid November i.e. 15<sup>th</sup> November.

b) Size Estimation:

### **Using Function Point Analysis:**

The steps in function point analysis are:

1. Count the number of functions of each proposed type
2. Compute the Unadjusted Function Points (UFP).
3. Find Total Degree of Influence (TDI).

4. Compute Value Adjustment Factor (VAF).

5. Find the Function Point Count (FPC).

- External Inputs: Functions related to data entering the system.
- External outputs: Functions related to data exiting the system.
- External Inquiries: They leads to data retrieval from system but don't change the system.
- Internal Files: Logical files maintained within the system. Log files are not included here.
- External interface Files: These are logical files for other applications which are used by our system.

In my project,

- External Inputs will be the News entered by the user.
- External Outputs will be a label ( 0 or 1 ) .
- External Inquiries – No External Inquiries
- Internal Files will include all the Jupyter Notebook files and Python Files
- External Interface Files will include datasets used for the project
- Effort is calculated above
- Technical documents would explain the functionality of software development model used, data-flow diagrams, gantt charts, E-R diagram, SRS of project etc.
- User documents would be the user manual



So,

No. of external Inputs = 1

No. of external outputs = 1

No. of External Inquiries = 0

No. of Internal Files = 7

No. of External interface Files = 2

Effort = 2.652 (Calculated above)

Technical documents = 20 pages(approx)

User documents = 5 pages(approx)

The 14 general characteristics are:

Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change.

Various processing complexity factors are: 3,2,4,1,5,3,2,1,2,4,3,2,4,2

Measurement Parameter	Count	*	Weighing Factor
External Inputs	1	*	4 = 4
External Outputs	1	*	4 = 4
External Inquiries	0	*	3 = 0
Internal Files	7	*	15 = 105
External Interface Files	2	*	5 = 10
			<b>Total = 123</b>

So sum of all  $f_i (i \leftarrow 1 \text{ to } 14) = 3+2+4+1+5+3+2+1+2+4+3+2+4+2 = 38$

Function Point, FP = Count-total \*  $[0.65 + 0.01 * \sum(f_i)]$

$$= 123 * [0.65 + 0.01 * 38]$$

$$= 123 * [0.65 + 0.38]$$

$$= 123 * 1.03$$

$$\mathbf{FP = 126.69}$$

$$\text{Productivity} = \text{FP} / \text{Effort}$$

$$= 126.69 / 2.652$$

$$\mathbf{\text{Productivity} = 47.77}$$

$$\text{Total pages of documentation} = \text{technical document} + \text{user document}$$

$$= 20 + 5 = 25 \text{ pages}$$

$$\text{Documentation} = \text{Pages of documentation} / \text{FP}$$

$$= 25 / 126.69$$

$$\mathbf{\text{Documentation} = 0.197}$$

## **7) Requirements Analysis**

### **7.1 Functional Requirements**

#### **High Priority ->**

- I. All the required libraries and packages must be downloaded.
- II. The user will give the news (in the form of text) as the input to the software.
- III. All the required libraries and packages must be downloaded.
- IV. The entered text will be preprocessed or cleaned.
- V. NLP Techniques will be applied on the text.
- VI. The Logistic Regression classifier will be trained.

- VII.** The Pipeline will be created.
- VIII.** As an output the software will predict the news as “Fake” or “Real”.

**Medium Priority ->**

- I.** A word cloud will be created from the Train Dataset.

## **7.2 Non Functional Requirements**

### **I. Reliability :**

- a) The software should be operational atleast 90 % of the time.
- b) Any bug detected would be fixed within 24 hours.

### **II. Usability :**

Anyone can easily use this software due to its simple functionality (the user just needs to input the news and the result will be shown on the screen)

### **III. Performance :**

The software should practically be able to support infinite users at the same time because there is no network connectivity between each and every user making it easier to run.

### **IV. Security :**

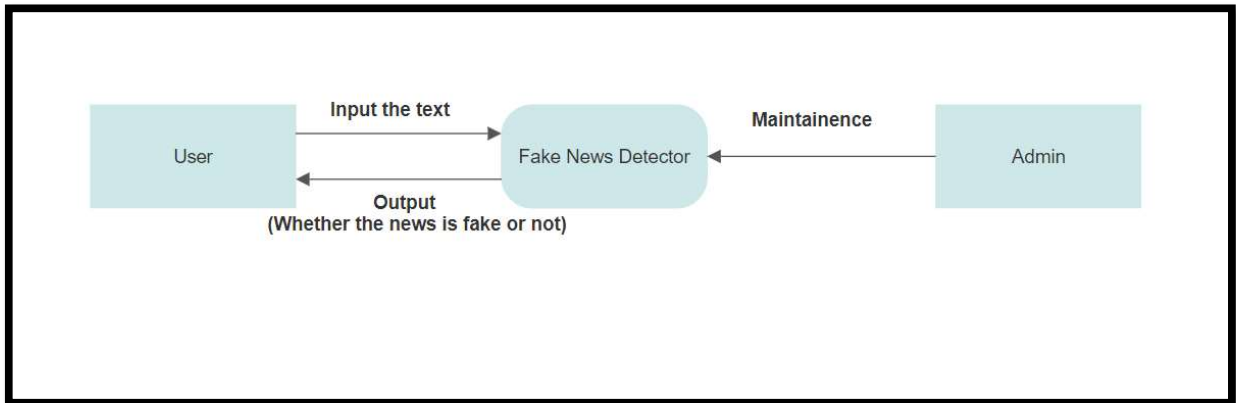
The user only enters the news and gets the result and since there is no need to login, the user's information is not stored anywhere and hence the system is secure.

### **V. Supportability :**

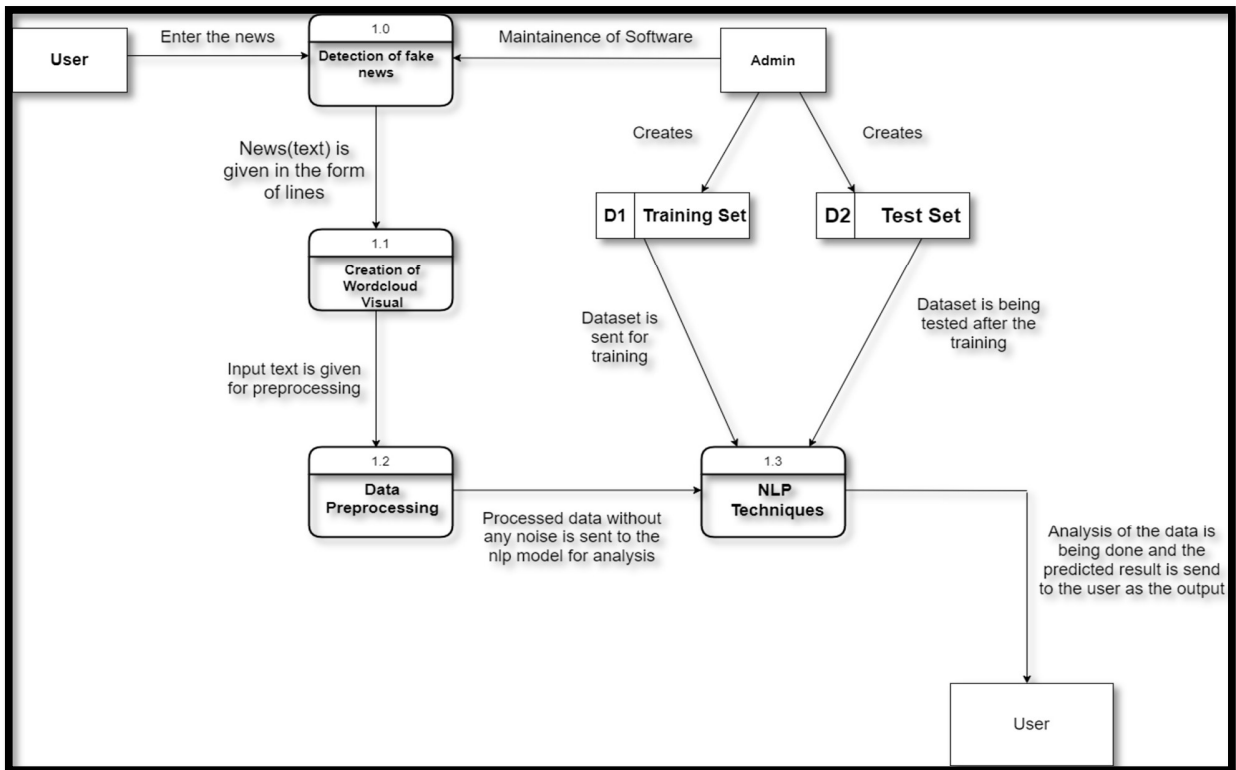
- a) The software can be used on Laptops and Computers only irrespective of the operating system used in the device.
- b) The device must have Anaconda/Jupyter Notebook (Frameworks for Python) installed.

## 8) Data Flow Diagrams (DFDs)

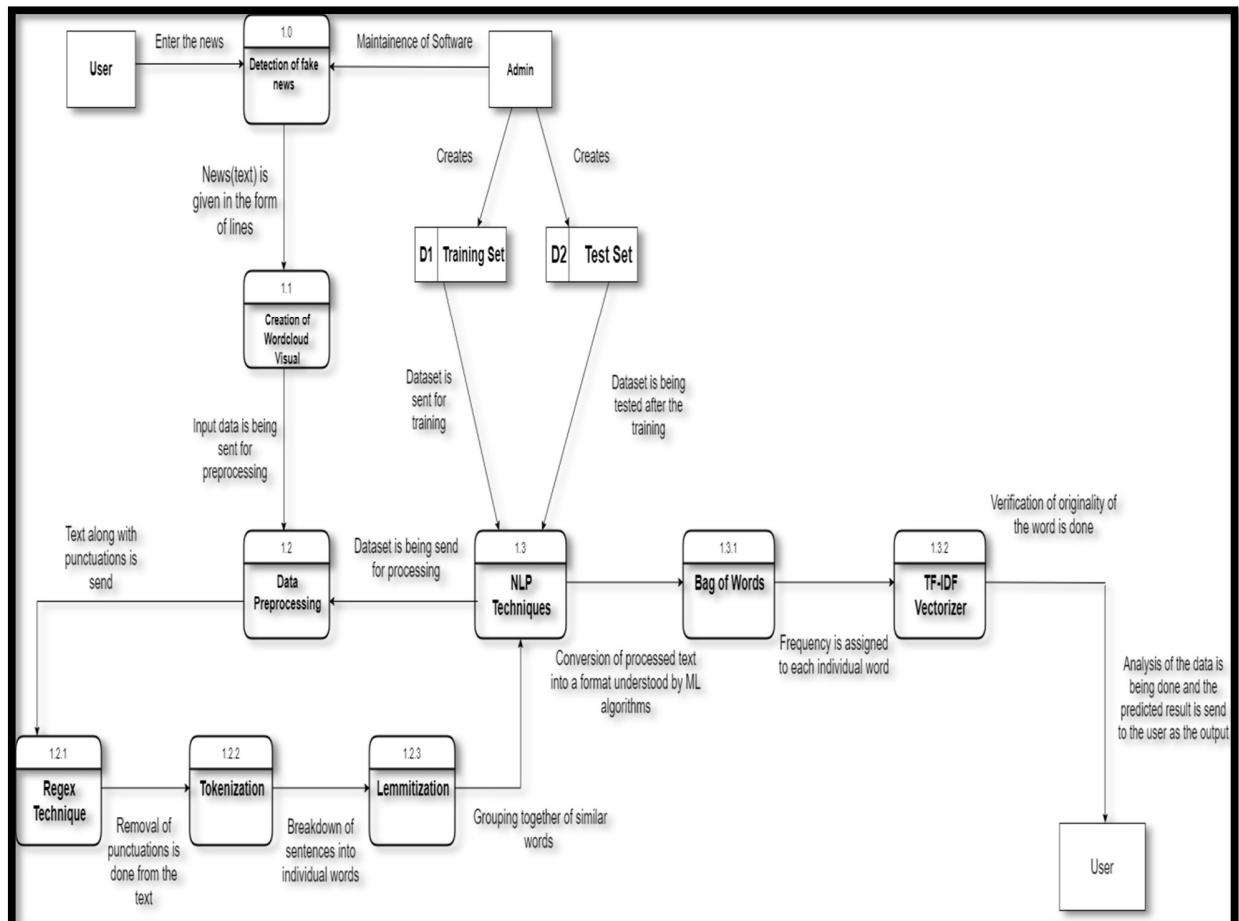
- Level 0 DFD



- Level 1 DFD



- **Level 2 DFD**



## 9) Use Cases

Since this is a machine learning project, the uses cases have been made on the basis of the code/program of the project.

### I. Datasets Creation

Use Case Name:	Datasets Creation
Summary:	The datasets that will be used in the software will be created.
Basic Flow:	<ol style="list-style-type: none"><li>1. The train dataset will be created which will contain the labels "0" or "1" corresponding to the news "fake" or "real" respectively.</li><li>2. The test dataset will be created which will only contain the news and not the labels</li></ol>
Alternate Flow:	-
Extension Points:	-
Preconditions:	The datasets must be created in a "csv" file format.
Postconditions:	The datasets formed will be used for doing all the analysis , training and making predictions.

### II. Input News

Use Case Name:	Input News
Summary:	The user will input the news.
Basic Flow:	<ol style="list-style-type: none"><li>1. The user will first open Jupyter Notebook .</li><li>2. The news will be then entered by the user from the Test Dataset.</li></ol>
Alternate Flow:	-
Extension Points:	-
Preconditions:	<ol style="list-style-type: none"><li>1. The user must have a PC/Laptop.</li><li>2. Jupyter Notebook must be installed in the PC/Laptop.</li></ol>

Postconditions:	The user will get the predicted result, whether the news is real or fake in the form of labels “1” or “0” respectively.
-----------------	---

**Note: This project contains 2 Datasets i.e. “Train Dataset” and “Test Dataset”. In the use cases below the word dataset has been used which refers to the “Train Dataset”. Wherever the “Test Dataset” is required it has been explicitly mentioned.**

### III. WordCloud Visual Creation

Use Case Name:	WordCloud Visual Creation
Summary:	A Word Cloud will be created using the dataset in which the size of each word will indicate its frequency in the dataset.
Basic Flow:	<ol style="list-style-type: none"> <li>1. The dataset will be read.</li> <li>2. While reading through the data set, each sentence is splitted into individual words and converted into lowercase.</li> <li>3. All the words are then stored in the form of a list.</li> <li>4. The background colour, height, weight and other parameters of the wordcloud are set and the wordcloud is created</li> </ol>
Alternate Flow:	-
Extension Points:	-
Preconditions:	The following libraries must be imported : <ol style="list-style-type: none"> <li>1. matplotlib</li> <li>2. wordcloud</li> </ol>
Postconditions:	A WordCloud will be displayed after executing the code.

#### IV. Data Preprocessing

Use Case Name:	Data Preprocessing
Summary:	All the noises, unwanted outliers, duplicate/wrong data, missing values and unimportant pieces of data from the dataset will be removed.
Basic Flow:	<ol style="list-style-type: none"><li>1. The removal of punctuations is done using Regular Expressions.</li><li>2. The long sentences are splitted into individual words using Tokenization.</li><li>3. The Stop words are removed from the dataset.</li><li>4. Similar words are grouped together using Lemmatization.</li></ol>
Alternate Flow:	-
Extension Points:	-
Preconditions:	The following libraries must be imported : <ol style="list-style-type: none"><li>1. re</li><li>2. nltk</li></ol>
Postconditions:	Cleaned dataset is obtained which will be processed further using the NLP techniques.

#### V. Applying NLP Techniques

Use Case Name:	Applying NLP Techniques
Summary:	NLP Techniques will be applied on the dataset to convert it into machine understandable format.
Basic Flow:	<ol style="list-style-type: none"><li>1. A bag of words/count vectorizer will be created to find out the frequency of each word in the input text.</li><li>2. The text is transformed into a representation of numbers using the TF-IDF Vectorizer.</li><li>3. This number representation of text will be used by the machine to do further processing</li></ol>
Alternate Flow:	-
Extension Points:	-



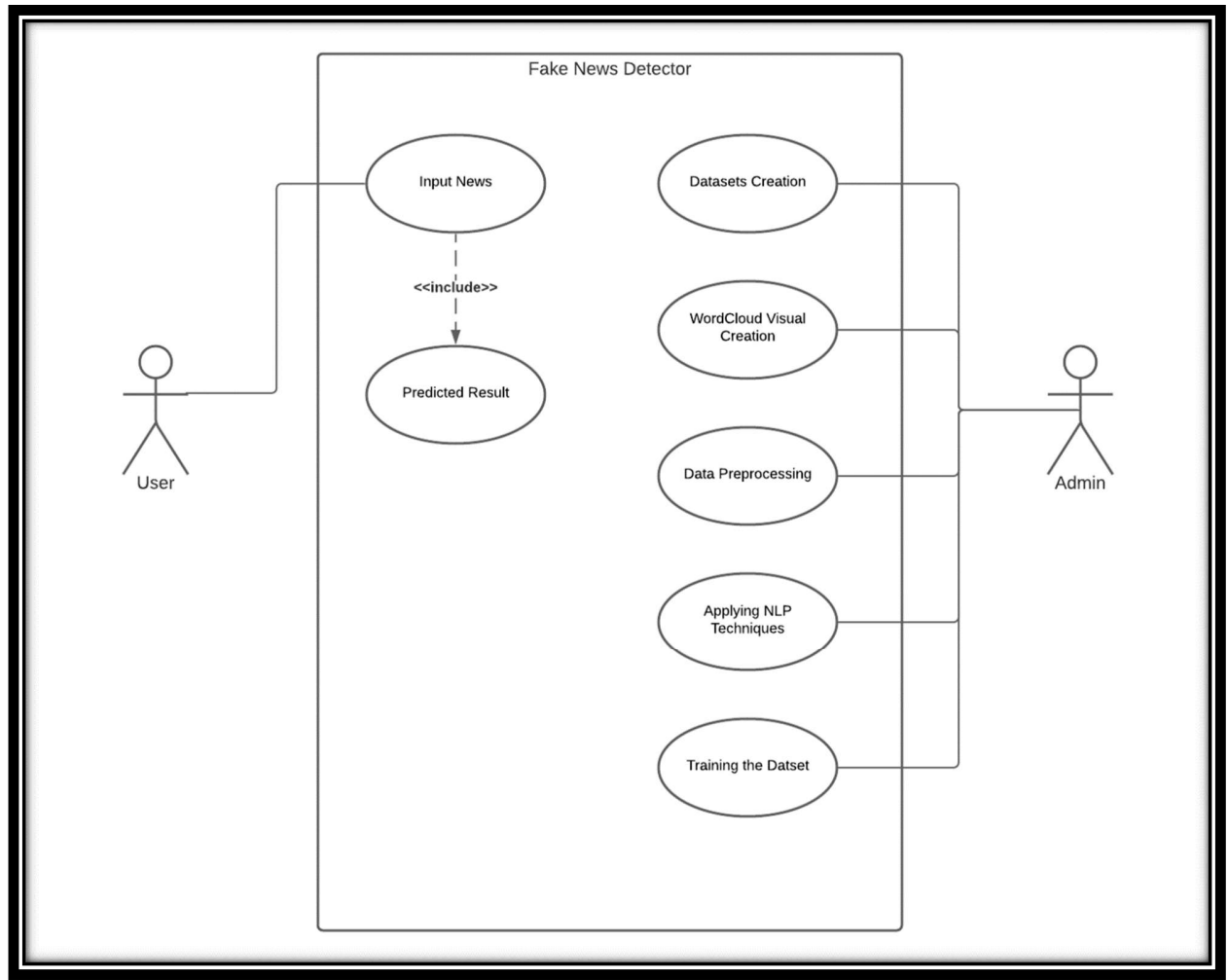
Preconditions:	<ol style="list-style-type: none"> <li>1. The following library must be imported : a) sklearn</li> <li>2. NLP Technique will only be applied to the preprocessed or cleaned data.</li> </ol>
Postconditions:	After applying the NLP Techniques dataset will be ready for training

## VI. Training the dataset

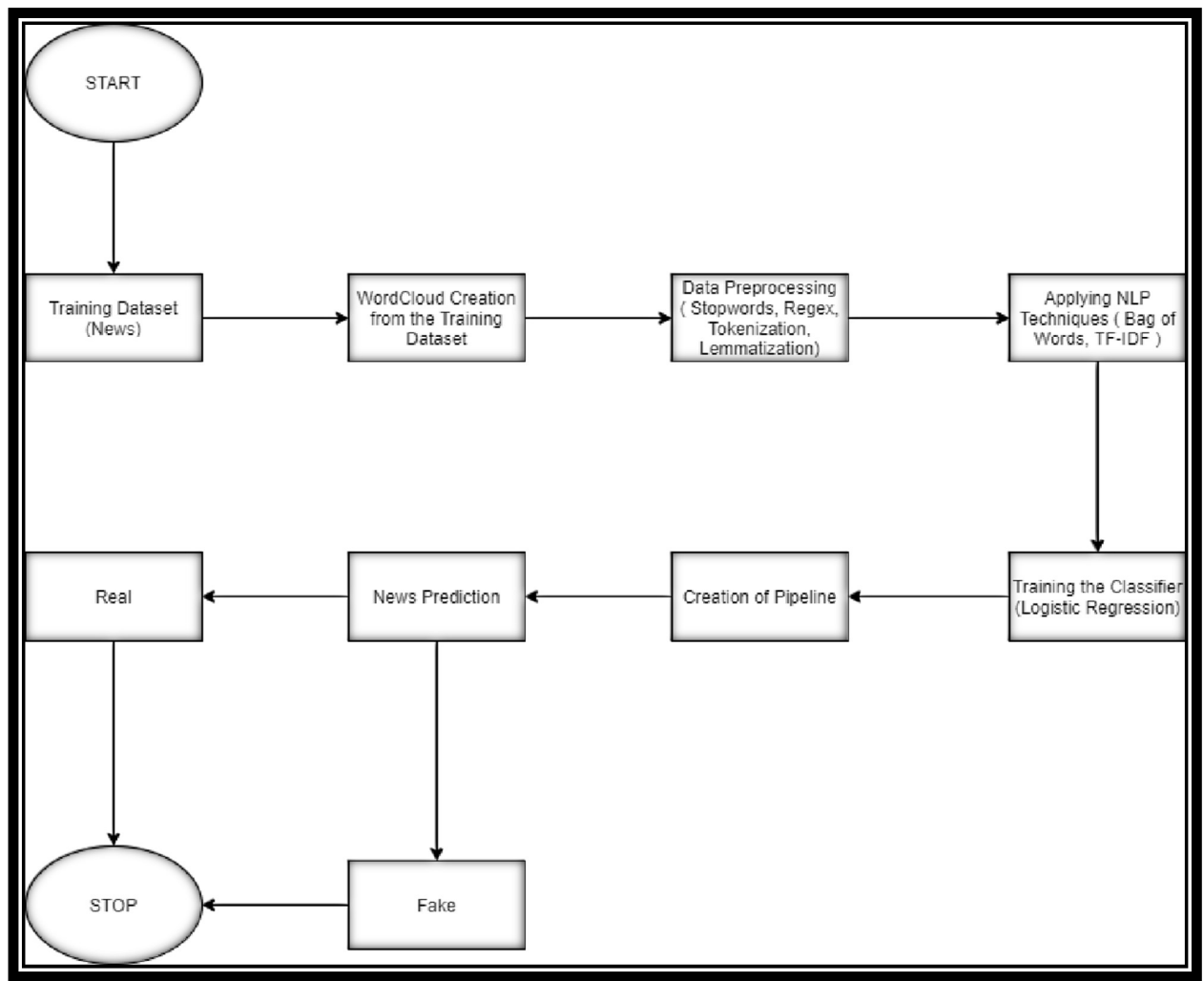
Use Case Name:	Training the dataset
Summary:	The dataset will be trained using the classification techniques.
Basic Flow:	<ol style="list-style-type: none"> <li>1. Two classification techniques namely Logistic Regression and Multinomial Naive Bayes will be performed on our datasets.</li> <li>2. The technique with higher accuracy will be used for prediction.</li> </ol>
Alternate Flow:	-
Extension Points:	-
Preconditions:	<ol style="list-style-type: none"> <li>(i) The following library must be imported : b) sklearn</li> <li>(ii) The data in numerical representation will be used for training.</li> </ol>
Postconditions:	After training the dataset, it will be used for the making predictions.

10)

## Use Case Diagram

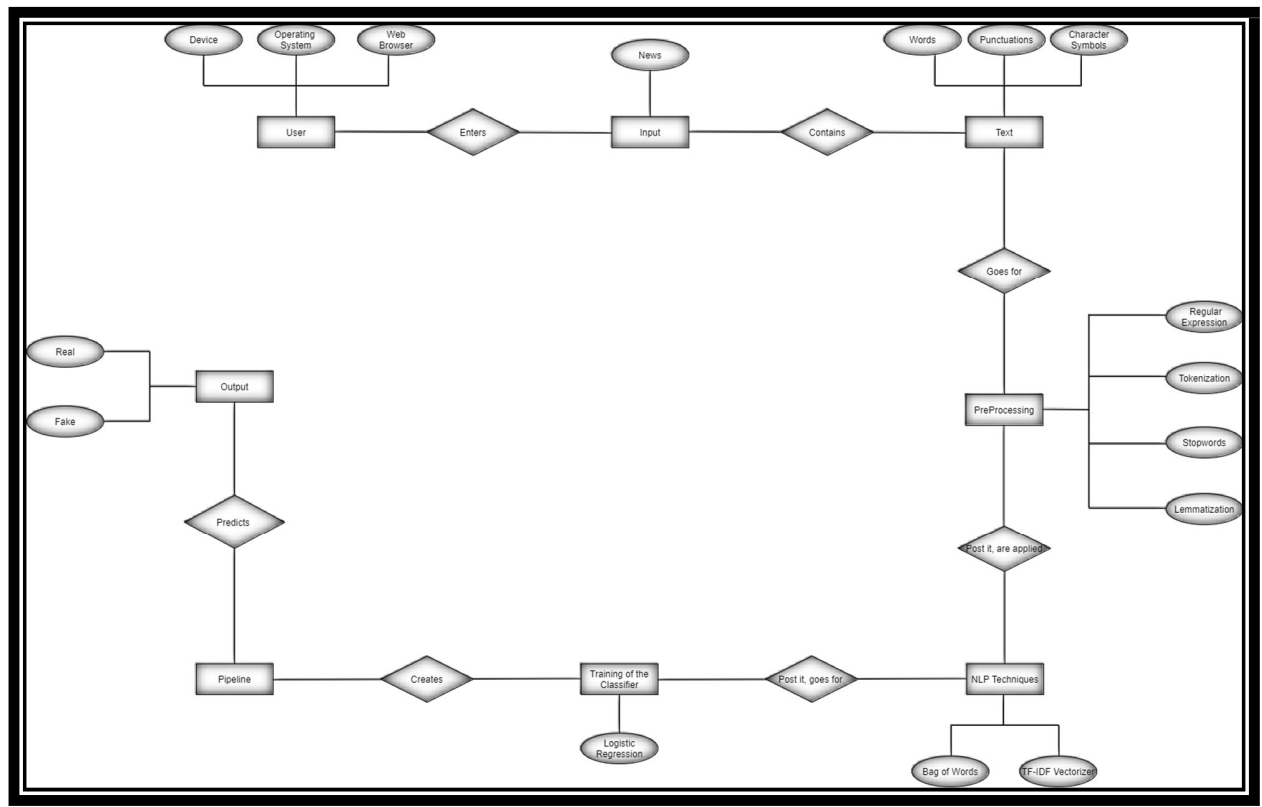


## 11) Flow Diagram



12)

## ER Diagram



## 13)

### Methodologies Used

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. So, the ML was used in Fake News Detector to predict the Output.

The specific methodology used in my project is **Natural Language Processing (NLP)**. It deals with how computers understand and translate human language. With NLP, machines can make sense of written or spoken text and perform tasks like translation, keyword extraction, topic classification, and more.

Natural Language Processing (NLP) applies two techniques to help computers understand text: syntactic analysis and semantic analysis. For my project the syntactic analysis is used which analyzes text using basic grammar rules to identify sentence structure, how words are organized, and how words relate to each other.

Some of its main sub-tasks include:

- **Tokenization** consists of breaking up a text into smaller parts called *tokens* (which can be sentences or words) to make text easier to handle.
- **Lemmatization** consist of reducing inflected words to their base form to make them easier to analyze.
- **Stop-word removal** removes frequently occurring words that don't add any semantic value, such as *I, they, have, like, yours, etc.*
- **Regular Expressions( Regex )** are a sequence of characters that forms a search pattern. It can be used to check if a string contains the specified search pattern.

Apart from it 3 important techniques were used in my project that are ->

- **Bag of Words** : A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
  1. A vocabulary of known words.
  2. A measure of the presence of known words.

It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

- **TF-IDF Vectorizer** : TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction.
- **Logistic Regression** : Logistic regression is one of the most popular Machine learning algorithm that comes under Supervised Learning techniques. is used to predict the categorical dependent variable with the help of independent variables.

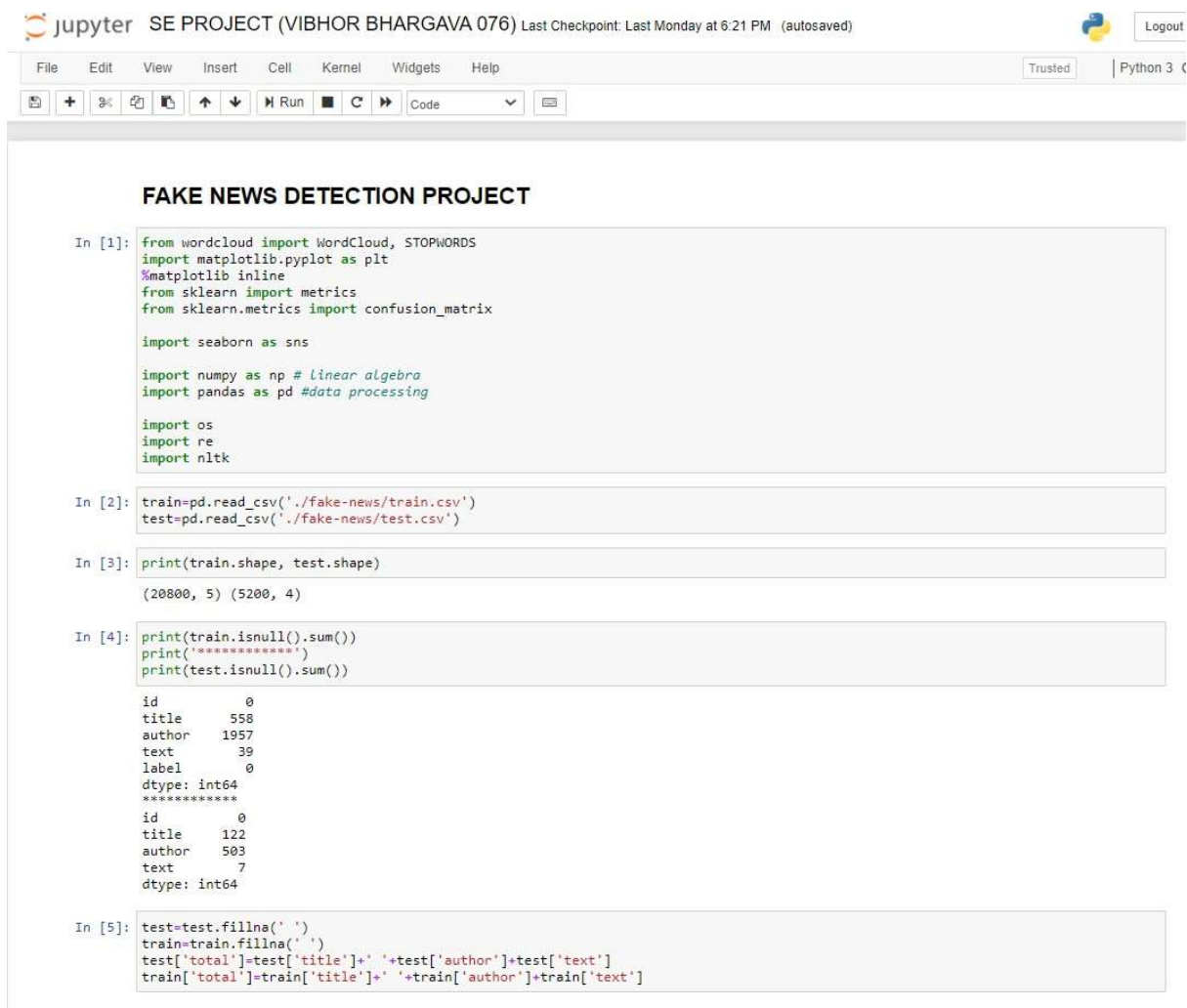
## 14) **Software and Hardware Platforms Used**

- a) **Software Platforms** -> Anaconda Navigator/Jupyter Notebook (6.0.3)
- b) **Hardware Platform** -> Inspiron 15 5000 Series; Intel core i5 – 8<sup>th</sup> Gen.
- c) **Technology** -> Python Programming Language

## 15) Online Tools Used

- a) **Draw.io** -> Use Case Diagram, Flow Diagram, ER Diagram
- b) **Lucidchart** -> Data Flow Diagrams (All Levels)

## 16) Code Snippets and Outputs



The screenshot shows a Jupyter Notebook titled "SE PROJECT (VIBHOR BHARGAVA 076)" with a "Last Checkpoint: Last Monday at 6:21 PM (autosaved)". The notebook contains five code cells. The first cell imports various libraries including WordCloud, matplotlib, sklearn, seaborn, numpy, pandas, os, re, and nltk. The second cell reads training and testing data from CSV files. The third cell prints the shapes of the training and testing data. The fourth cell prints the number of null values in the training and testing data. The fifth cell fills missing values and concatenates the features into a single 'total' column for both training and testing data.

```
In [1]: from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import metrics
from sklearn.metrics import confusion_matrix

import seaborn as sns

import numpy as np # linear algebra
import pandas as pd #data processing

import os
import re
import nltk

In [2]: train=pd.read_csv('./fake-news/train.csv')
test=pd.read_csv('./fake-news/test.csv')

In [3]: print(train.shape, test.shape)

(20800, 5) (5200, 4)

In [4]: print(train.isnull().sum())
print('*****')
print(test.isnull().sum())

id          0
title      558
author    1957
text        39
label       0
dtype: int64
*****
id          0
title     122
author    503
text        7
dtype: int64

In [5]: test=test.fillna(' ')
train=train.fillna(' ')
test['total']=test['title']+test['author']+test['text']
train['total']=train['title']+train['author']+train['text']
```







### 3. StopWords

```
from nltk.corpus import stopwords
st = stopwords.words('english')

['i', 'me', 'my', 'myself', 'us', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourselves', 'ourself', 'yourself', 'he', 'him', 'his', 'himselves', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'whose', 'son', 'this', 'that', 'these', 'those', 'an', 'is', 'a', 'ce', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'in', 'at', 'by', 'to', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'hurther', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'no', 'not', 'not only', 'only', 'or', 'so', 'than', 'too', 'very', 's', 't', 'e', 'an', 'will', 'just', 'don', 'don't', 'should', 'shouldn't', 'not', 'didn't', 'll', 'n', 'o', 're', 've', 'y', 'ain', 'ar', 'en', 'aren't', 'couldn't', 'couldn't', 'didn't', 'doesn't', 'hadn't', 'hasn't', 'haven't', 'have', 'n't', 'isn't', 'can't', 'mightn't', 'mightn't', 'mustn't', 'mustn't', 'needn't', 'needn't', 'shan't', 'shouln't', 'shouldn't', 'wasn't', 'wasn't', 'weren't', 'weren't', 'in', 'won't', 'wouldn't']

sentence = "Covid-19 pandemic has impacted many countries and what it did to economy is very stressful"

words = nltk.word_tokenize(sentence)
words = [w for w in words if w not in st]
```

```
In [17]: print(words)

['Covid-19', 'pandemic', 'impacted', 'many', 'countries', 'economy', 'stressful']
```

### 4. Lemmatization

```
In [18]: import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
input_str = "been had done languages cities mice"
nltk_tokens = nltk.word_tokenize(input_str)
for word in nltk_tokens:
    print(word, "->", lemmatizer.lemmatize(word))

been->been
done->done
languages->language
cities->city
mice->mouse
```

### Let's Apply

```
In [19]: lemmatizer = WordNetLemmatizer()
for index, row in train.iterrows():

    sentence = row['total']
    sentence = re.sub(r'[^\w\s]', '', sentence) #cleaning

    words = nltk.word_tokenize(sentence) #tokenization

    words = [w for w in words if w not in stopwords.words('english')] #stopwords removal

    for word in words:
        filter_sentence = filter_sentence + re.sub(r'lemmatize', lemmatizer.lemmatize(word)).lower()
```

### Applying NLP Techniques

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

## Bag-of-words / CountVectorizer

```
In [23]: corpus = [
        'This is the first document.',
        'This document is the second document.',
        'And this is the third one.',
        'Is this the first document?',
    ]
    vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(corpus)
    print(vectorizer.get_feature_names())

['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']

In [24]: print(X.toarray())

[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

## TF-IDF Vectorizer

```
In [25]: def vectorize_text(features, max_features):
        vectorizer = TfidfVectorizer(stop_words='english',
                                     decode_error='strict',
                                     analyzer='word',
                                     ngram_range=(1, 2),
                                     max_features=max_features,
                                     #max_df=0.5 # Verwendet im ML-Kurs unter Preprocessing
                                     )
        feature_vec = vectorizer.fit_transform(features)
        return feature_vec.toarray()

In [26]: tfidf_features = vectorize_text(['hello how are you doing', 'hi i am doing fine'], 30)

In [27]: tfidf_features

Out[27]: array([[0.44943642, 0., 0., 0.6316672, 0.6316672,
                  0., 0.],
                [0.33517574, 0.47107781, 0.47107781, 0., 0.,
                  0.47107781, 0.47107781]])
```

## Let's Apply

```
In [28]: #Feature extraction using count vectorization and tfidf.
        count_vectorizer = CountVectorizer()
        count_vectorizer.fit_transform(X_train)
        freq_term_matrix = count_vectorizer.transform(X_train)
        tfidf = TfidfTransformer(norm="l2")
        tfidf.fit(freq_term_matrix)
        tf_idf_matrix = tfidf.fit_transform(freq_term_matrix)

In [29]: tf_idf_matrix

Out[29]: <20800x220387 sparse matrix of type '<class 'numpy.float64'>'
        with 5987666 stored elements in Compressed Sparse Row format>
```

## Modelling

```
In [30]: test_counts = count_vectorizer.transform(test['total'].values)
        test_tfidf = tfidf.transform(test_counts)

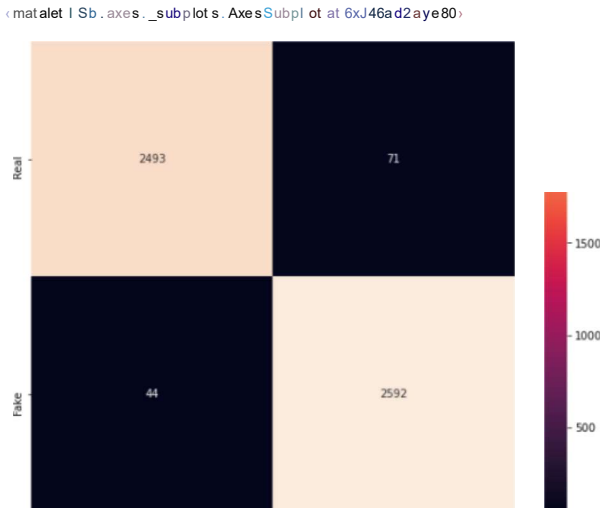
        #split in samples
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, Y_train, random_state=0)
```

## Logistic Regression

```
In [1]: from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(C=1e5)
logreg.fit(X_train, y_train)
pred = logreg.predict(X_test)
print('Accuracy of Lasso classifier on training set: {:.2f}'
      .format(logreg.score(X_train, y_train)))
print('Accuracy of Lasso classifier on test set: {:.2f}'
      .format(logreg.score(X_test, y_test)))

plt.figure(figsize=(10,8))
sns.heatmap(cm,annot=True,fmt='d', xticklabels = target_names,yticklabels = target_names)

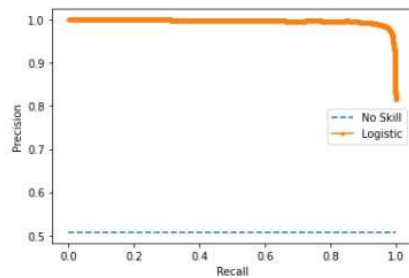
Accuracy of Lasso classifier on training set: 1.00
Accuracy of Lasso classifier on test set: 0.96
```



## Precision Recall Curve

```
In [32]: from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from matplotlib import pyplot
X_train, X_test, y_train, y_test = train_test_split(tf_idf_matrix, Y_train, random_state=0)
# fit a model
logreg = LogisticRegression(C=1e5)
a=logreg.fit(X_train, y_train)
# predict probabilities
lr_probs = a.predict_proba(X_test)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# predict class values
yhat = a.predict(X_test)
lr_precision, lr_recall, _ = precision_recall_curve(y_test, lr_probs)
lr_f1, lr_auc = f1_score(y_test, yhat), auc(lr_recall, lr_precision)
# summarize scores
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
# plot the precision-recall curves
no_skill = len(y_test[y_test==1]) / len(y_test)
pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
pyplot.plot(lr_recall, lr_precision, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

Logistic: f1=0.978 auc=0.997



## MultinomialNB

```
In [33]: from sklearn.naive_bayes import MultinomialNB

NB = MultinomialNB()
NB.fit(X_train, y_train)
pred = NB.predict(X_test)
print('Accuracy of NB classifier on training set: {:.2f}'
      .format(NB.score(X_train, y_train)))
print('Accuracy of NB classifier on test set: {:.2f}'
      .format(NB.score(X_test, y_test)))
cm = confusion_matrix(y_test, pred)
cm
```

Accuracy of NB classifier on training set: 0.88  
Accuracy of NB classifier on test set: 0.83

```
Out[33]: array([[2558,    6],
                [ 853, 1783]], dtype=int64)
```

## Pipeline

```
In [34]: #ss T g l n g the worE o f es ego t n as once trans/oroed sector*s non 'l 6e trozts/-armed again s l n g p Ape L i n e.
X_train = Craln ['total']
Y_train = Craln ['Label']

In [35]: from sklearn.pipeline import Pipeline
#from sklearn.externals import joblib
from sklearn import linear_model
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

In [36]: from sklearn import linear model
pipeline = Pioeline([
    ('vect', CDuntVectorizer()),
    ('tfidf', TO idfT ransowe r{n o rm='l2'}),
    ('clf', l i n e a r m o d e l . L o g i s t i c R e g r e s s i o n (C=1e5)),
])

In [37]: pipeline.fit(X_train, Y_train)

Out [37]: P i p e l i n e ( s t e p s = [ ( ' v e c t ' , C o u n t V e c t o r i z e r ( ) ) , ( ' t f i d f ' , I f i d * T r a n s f o r m e r ( ) ) , ( ' c l f ' , L o g i s t i c R e g r e s s i o n ( C = 1 0 W B . 0 ) ) ] )

In [38]: pipeline.predict(["flynn hillary clinton big woman campus breitbart daniel j flynnnever get feeling life circle roundabout rather
array([0, dtype=int64)

Out [38]: a r r a y ( [ 0 ] , d t y p e = i n t 6 4 )

In [39]: #saving the pipeline

filename = 'pipeline.sav'
joblib.dump(pipeline, filename)

Out [39]: ['pipeline.sav']

In [40]: filename = './pipeline.sav'
```

## Prediction

```
In [41]: loaded_model = joblib.load(filename)
result = loaded_model.predict(["Specter of Trump Loosens Tongues, if Not Purse Strings, in Silicon Valley - The New York Times"]);
print(result)

In [42]: loaded_model = joblib.load(filename)
result = loaded_model.predict(["Russian warships ready to strike militants near Aleppo"]);
print(result)

result = loaded_model.predict(["Trump Heats Down in December The US Postal Service Defeating The Election For Clinton"]);
print(result)

In [44]: loaded_model = joblib.load(filename)
result = loaded_model.predict(["Hueka bee Responds to Flaag- Burning ieum9 Hat ené | ñé&hataf's Happened No This Country ?ñé•"]);
print(result)

result = loaded_model.predict(["The Rise of Flan date ry Va cc i n a t i o n s F l e a n s l h e E n d o f M e d i c a l F r e e d o m "])
print(result)

In [46]: loaded_model = joblib.load(filename)
result = loaded_model.predict(["Bill Weld, Running as a Libertarian, Likens Donald Trump's Immigration Plan to Kristallnacht -
print(result)
```

```
In [47]: loaded_model = joblib.load(filename)
result = loaded_model.predict(["Senate Passes Bill Exposing Saudi Arabia to 9/11 Legal Claims - The New York Times"])
print(result)

[0]

In [48]: loaded_model = joblib.load(filename)
result = loaded_model.predict(["The Failure of Democracy: How The Oligarchs Plan To Steal The Election"])
print(result)

[1]
```

## 17) Testing Methodology

### A. Test Case Designs

#### i) Unit Testing ->

Test Case ID	TC-1A
Test Case Summary	To verify that the wordcloud visual is created successfully.
Prerequisites	None
Test Procedure	<ol style="list-style-type: none"> <li>1. The dataset will be read.</li> <li>2. While reading through the data set, each sentence is splitted into individual words and converted into lowercase.</li> <li>3. All the words are then stored in the form of a list.</li> <li>4. The background colour, height, width and other parameters of the wordcloud are set and the wordcloud is created.</li> </ol>
Test Data	Training Dataset
Expected Result	A wordcloud visual will be displayed with the specified height and width.
Actual Result	A wordcloud is displayed of the specified height and width.
Status	Passed

Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)

Test Case ID	TC-1B
Test Case Summary	To verify that the wordcloud visual is created successfully.
Prerequisites	None
Test Procedure	<ol style="list-style-type: none"> <li>1) The dataset will be read.</li> <li>2) While reading through the data set, each sentence is splitted into individual words and converted into lowercase.</li> <li>3) All the words are then stored in the form of a list.</li> <li>4) The background colour, height, width and other parameters of the wordcloud are set and the wordcloud is created.</li> </ol>
Test Data	Training Dataset
Expected Result	A wordcloud visual will be displayed with the specified height and width.
Actual Result	Attribute Error
Status	Failed
Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)

Test Case ID	TC-2A
--------------	-------



Test Case Summary	To verify that the data pre processing techniques are applied successfully.
Prerequisites	All the Datasets must be created.
Test Procedure	<ol style="list-style-type: none"> <li>1) Remove unwanted outliers, duplicate/wrong data, missing values and punctuations from the text.</li> <li>2) Split the text into words using Tokenization.</li> <li>3) Remove all the Stopwords.</li> <li>4) Group together of inflected form of words.</li> </ol>
Test Data	<ol style="list-style-type: none"> <li>1) "!&lt;/&gt; hello please\$\$ &lt;/&gt;^s!!!u%%bs&amp;&amp;%\$cri@@@@be^^^&amp;&amp;!&amp; &lt;/&gt;*to@# the&amp;&amp;\ cha@@@@n##%^&amp;nel!@# %%\$"</li> <li>2) "Hello how are you"</li> <li>3) "Covid-19 pandemic has impacted many countries and what it did to economy is very stressful"</li> <li>4) "been had done languages cities mice"</li> </ol>
Expected Result	<ol style="list-style-type: none"> <li>1) All the characters will be removed.</li> <li>2) The text must be splitted into words.</li> <li>3) All the stopwords will be removed.</li> <li>4) Inflected form of words will be represented as one word.</li> </ol>
Actual Result	<ol style="list-style-type: none"> <li>1) hello please subscribe to the channel</li> <li>2) ['Hello', 'how', 'are', 'you']</li> <li>3) ['Covid-19', 'pandemic', 'impacted', 'many', 'countries', 'economy', 'stressful']</li> <li>4) been had done language city mouse</li> </ol>
Status	Passed

Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)
------------------	---

Test Case ID	TC-2B
Test Case Summary	To verify that the data pre processing techniques are applied successfully.
Prerequisites	All the Datasets must be created.
Test Procedure	<ol style="list-style-type: none"> <li>1) Remove unwanted outliers, duplicate/wrong data, missing values and punctuations from the text.</li> <li>2) Split the text into words using Tokenization.</li> <li>3) Remove all the Stopwords.</li> <li>4) Group together of inflected form of words.</li> </ol>
Test Data	<ol style="list-style-type: none"> <li>1) "!&lt;/&gt; hello please\$\$ &lt;/&gt;^s!!!u%%bs&amp;&amp;%\$cri@@@@be^^^&amp;&amp;!&amp; &lt;/&gt;*to@# the&amp;&amp;\ cha@@@@n###^^&amp;nel!@# %%"</li> <li>2) "Hello how are you"</li> <li>3) "Covid-19 pandemic has impacted many countries and what it did to economy is very stressful"</li> <li>4) "been had done languages cities mice"</li> </ol>
Expected Result	<ol style="list-style-type: none"> <li>1) All the characters will be removed.</li> <li>2) The text must be splitted into words.</li> <li>3) All the stopwords will be removed.</li> <li>4) Inflected form of words will be represented as one word.</li> </ol>
Actual Result	Error -> Due to Incorrect Syntax
Status	Failed
Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10

	Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)
--	---

Test Case ID	TC-3A
Test Case Summary	To verify that user entered text is successfully converted into a machine understandable format.
Prerequisites	The dataset must be preprocessed or cleaned.
Test Procedure	<ol style="list-style-type: none"> <li>1) A bag of words/count vectorizer will be created to find out the frequency of each word in the input text.</li> <li>2) The text is transformed into a representation of numbers using the TF-IDF Vectorizer.</li> </ol>
Test Data	<ol style="list-style-type: none"> <li>1) corpus = [ <ul style="list-style-type: none"> <li>'This is the first document.'</li> <li>'This document is the second document.'</li> <li>'And this is the third one.'</li> <li>'Is this the first document?'</li> </ul> </li> <li>2) (['hello how are you doing','hi i am doing fine'],30)</li> </ol>
Expected Result	<ol style="list-style-type: none"> <li>1) A bag of words/count vectorizer will be created to with frequency of each word in the input text.</li> <li>2) The text will be transformed into a representation of numbers.</li> </ol>
Actual Result	<ol style="list-style-type: none"> <li>1) ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']    [[0 1 1 1 0 0 1 0 1]  [0 2 0 1 0 1 1 0 1]  [1 0 0 1 1 0 1 1 1]  [0 1 1 1 0 0 1 0 1]] </li> <li>2)  array([[0.44943642, 0. , 0. , 0.6316672 , 0.6316672 ,  0. , 0. ],  [0.33517574, 0.47107781, 0.47107781, 0. , 0. ,  0.47107781, 0.47107781]]) </li> </ol>
Status	Passed

Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)

Test Case ID	TC-3B
Test Case Summary	To verify that user entered text is successfully converted into a machine understandable format.
Prerequisites	The dataset must be preprocessed or cleaned.
Test Procedure	<ol style="list-style-type: none"> <li>1) A bag of words/count vectorizer will be created to find out the frequency of each word in the input text.</li> <li>2) The text is transformed into a representation of numbers using the TF-IDF Vectorizer.</li> </ol>
Test Data	<ol style="list-style-type: none"> <li>1) corpus = [ <ul style="list-style-type: none"> <li>'This is the first document.'</li> <li>'This document is the second document.'</li> <li>'And this is the third one.'</li> <li>'Is this the first document?'</li> </ul> </li> <li>2) (['hello how are you doing','hi i am doing fine'],30)</li> </ol>
Expected Result	<ol style="list-style-type: none"> <li>1) A bag of words/count vectorizer will be created to with frequency of each word in the input text.</li> <li>2) The text will be transformed into a representation of numbers.</li> </ol>
Actual Result	Error in Output -> Since the text was not preprocessed.
Status	Failed
Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)

**ii) Integration Testing ->**

Test Case ID	TC-4A
Test Case Summary	To verify that the system is producing/predicting output.
Prerequisites	Each and every step should be performed in order.
Test Procedure	<ol style="list-style-type: none"><li>1) Text will be preprocessed or cleaned.</li><li>2) Text will be converted into a machine understandable format using NLP Techniques.</li><li>3) Classifier will be trained using Logistic Regression.</li><li>4) Pipeline will be created.</li><li>5) Prediction.</li></ol>
Test Data	News entered by the user.
Expected Result	The news will be displayed as “Real” or “Fake”.
Actual Result	The news will be displayed as “Real” or “Fake”.
Status	Passed
Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)

Test Case ID	TC-4B
Test Case Summary	To verify that the system is producing/predicting output.
Prerequisites	Each and every step should be performed in order.
Test Procedure	<ol style="list-style-type: none"><li>1) Text will be preprocessed or cleaned.</li></ol>

	2) Text will be converted into a machine understandable format using NLP Techniques. 3) Classifier will be trained using Logistic Regression. 4) Pipeline will be created. 5) Prediction.
Test Data	News entered by the user.
Expected Result	The news will be displayed as “Real” or “Fake”.
Actual Result	The output will not be displayed due to error in the code or the steps are not being performed order wise.
Status	Failed
Test Environment	Device- Dell Inspiron 15 5000 Series; Intel core i5 – 8 <sup>th</sup> Gen. OS- Windows 10 Browser- Google Chrome; Version – 87.0.4280.66 (64 bit)

### iii) Regression Testing ->

Regression Tests replicate bugs that we have previously encountered and fixed. I have not encountered any bugs that needed to be fixed and tested, therefore no regression tests have been performed for my project.

## B. Performance Metrics

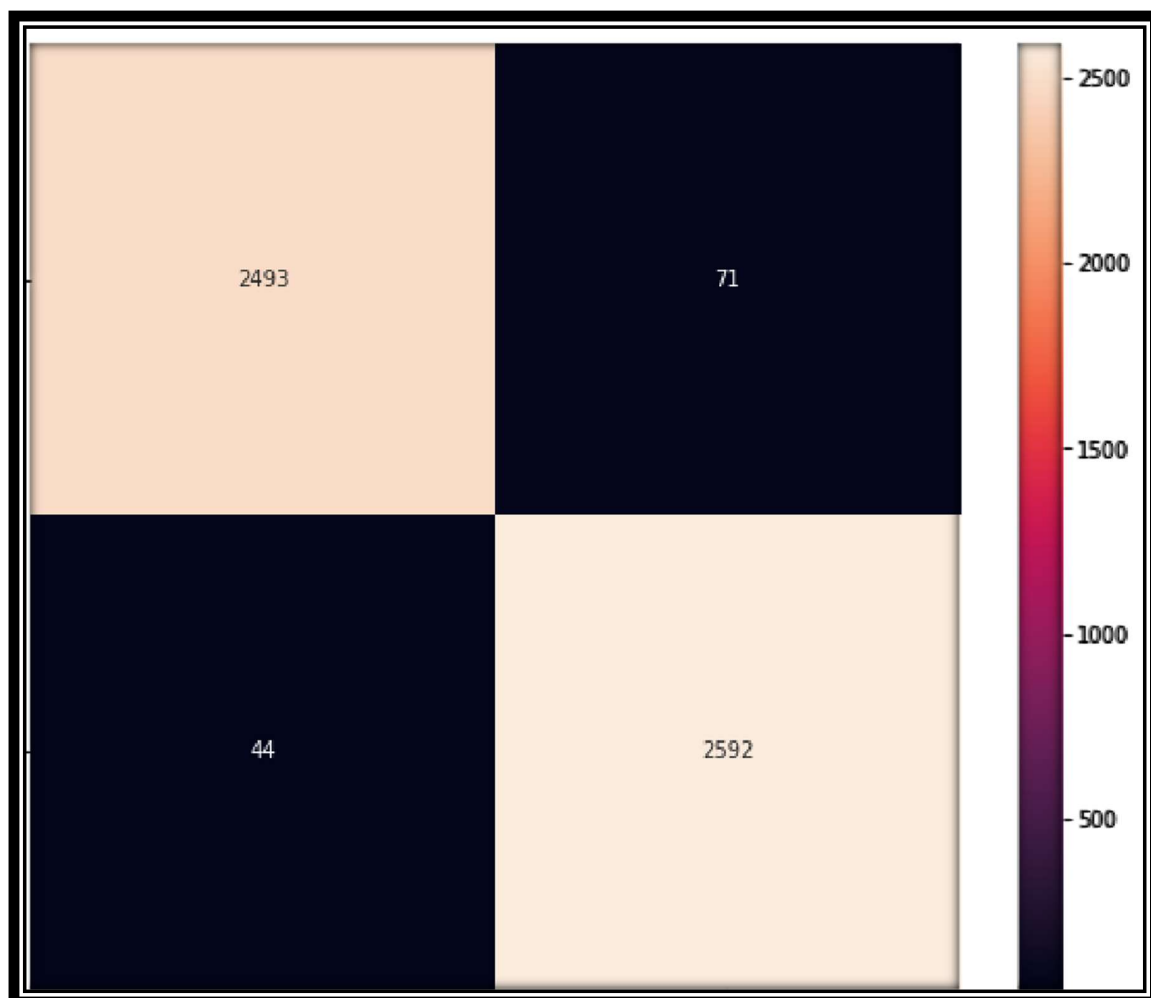
### i) Confusion Matrix ->

It is a square matrix table of  $N \times N$  where  $N$  is the number of classes that the model needs to classify. It's best used for classification models that categorizes an outcome into a finite set of values. These values are known as labels. One axis is the label that the model predicted and the other is the actual label.

## Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

The confusion matrix for my project is as follows ->



From the above matrix,

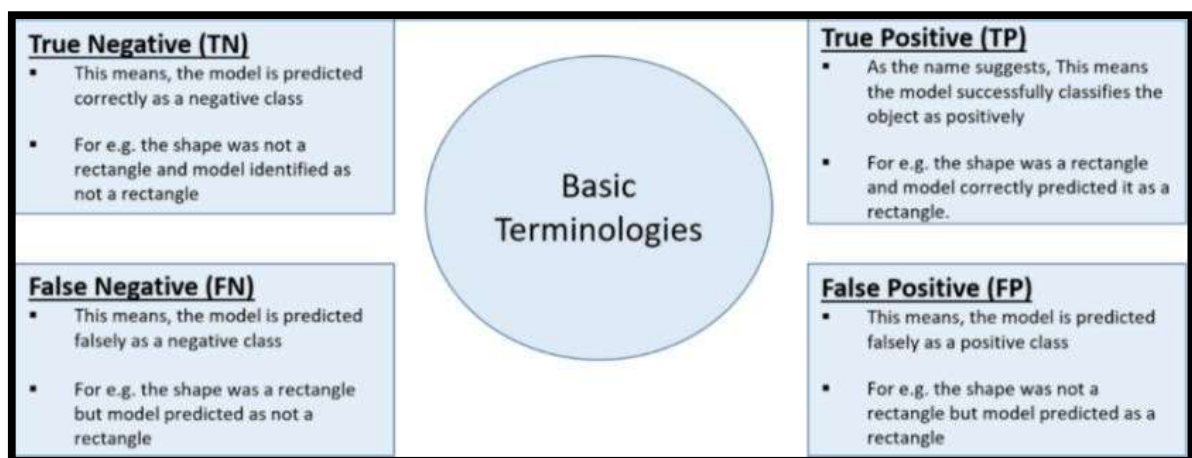
**True Positives (TPs) = 2493**

**False Positives (FPs) = 71**

**False Negatives (FNs) = 44**

**True Negatives (TNs) = 2592**

Now let us understand what these terms actually mean,



Let us calculate some important measures using these values,

- i) **Precision** -> It identifies the frequency with which a model was correct when predicting the positive class. This means the prediction frequency of a positive class by the model.

$$\text{Precision} = (\text{True Positives}) / (\text{True Positives} + \text{False Positives})$$

$$\text{Precision} = 2493 / (2493+71) = 0.9723 = 97.23 \%$$

- ii) **Recall** -> It means, the percentage of correctly identified actual True Positive class. In other words, recall measures the number of correct predictions, divided by the number of results that should have been predicted correctly.

$$\text{Recall} = (\text{True Positives}) / (\text{True Positives} + \text{False Negatives})$$



$$\text{Recall} = 2493 / (2493+44) = 0.9826 = 98.26 \%$$

- iii) **Accuracy** -> It is the most basic way of evaluating the learning model. It is a ratio between the positive (TN+TP) predictions vs the total number of predictions. If the ratio is high then the model has a high prediction rate.

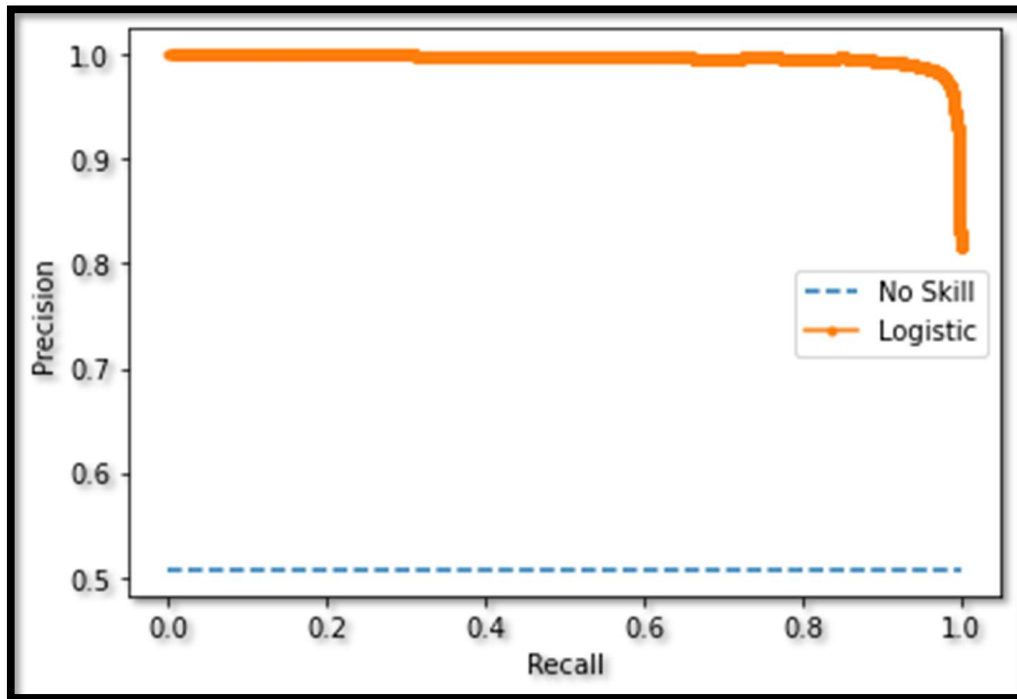
$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$$

$$\text{Accuracy} = (2493+2592) / (2493+71+44+2592) = 0.9778 = 97.78\% \sim 98 \%$$

Measure	Value	Derivations
Sensitivity	0.9827	$TPR = TP / (TP + FN)$
Specificity	0.9733	$SPC = TN / (FP + TN)$
Precision	0.9723	$PPV = TP / (TP + FP)$
Negative Predictive Value	0.9833	$NPV = TN / (TN + FN)$
False Positive Rate	0.0267	$FPR = FP / (FP + TN)$
False Discovery Rate	0.0277	$FDR = FP / (FP + TP)$
False Negative Rate	0.0173	$FNR = FN / (FN + TP)$
Accuracy	0.9779	$ACC = (TP + TN) / (P + N)$
F1 Score	0.9775	$F1 = 2TP / (2TP + FP + FN)$
Matthews Correlation Coefficient	0.9558	$TP \cdot TN - FP \cdot FN / \sqrt{((TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN))}$

**Table for all the Important Measures**

ii) Precision Recall Curve ->



18) **Beneficiaries**

Since the project is being made on a general topic, any person including children, middle-aged people and senior citizens all can use this software since it is important for everyone to know the Real and Correct News.

To be specific, this project could be practically used by any media company to automatically predict whether the circulating news is fake or not.