

A Project Report on

**FAKE NEWS DETECTION USING MACHINE LEARNING**

Submitted by

V.S.PAVITHRA -R180258

M.SOUMYA -R180919

Submitted to

IIIT RK Valley Idupulapaya, Vempalli, YSR  
Kadapa Andhra Pradesh, India PIN 516330.



Under the guidance of

**Mrs. V.SRAVANI**

**Assistant Professor**

as a part of

Partial fulfillment of the degree of Bachelor of  
Technology in Computer Science and Engineering

Date: 14-08-2023.

# **CERTIFICATE**

This is to certify that the report entitled “ FAKE NEWS DETECTION USING MACHINE LEARNING” submitted by V.S.Pavithra bearing ID.No. R180258 and M.Soumya bearing ID.No R180919 in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out by them under my supervision and guidance. The report has not been submitted previously in part or in full to this or another University or Institution for the award of any degree or diploma.

V.Sravani,  
Project Internal Guide,  
Internal Guide,  
Computer Science and Engineering

N.Satyanandaram,  
Head of the department  
Project  
Computer Science and engineering  
R.K Valley, RGUKT

## **Acknowledgement**

I would like to express my sincere gratitude to **Ms.M.SRAVANI** , my project internal guide for valuable suggestions and keen interest throughout the progress of my course of research.I am grateful to **N.Sathyanandaram** , HOD CSE , for providing excellent computing facilities and a congenial atmosphere for progressing with my project.At the outset, I would like to thank Rajiv Gandhi University of Knowledge Technologies and iB Hubs Team , for providing all the necessary resources for the successful completion of my course work.

## **ABSTRACT**

In recent years, due to the booming development of online social networks, fake news for various commercial and political purposes has been appearing in large numbers and widespread in the online world. With deceptive words, online social network users can get infected by these online fake news easily, which has brought about tremendous effects on the offline society already. An important goal in improving the trustworthiness of information in online social networks is to identify the fake news timely. This paper aims at investigating the principles, methodologies and algorithms for detecting fake news articles, creators and subjects from online social networks and evaluating the corresponding performance. Information preciseness on Internet, especially on social media, is an increasingly important concern, but web-scale data hampers, ability to identify, evaluate and correct such data, or so called "fake news," present in these platforms. In this paper, we propose a method for "fake news" detection and ways to apply it on Facebook, one of the most popular online social media platforms. This method uses Naive Bayes classification model to predict whether a post on Facebook will be labeled as real or fake. The results may be improved by applying several techniques that are discussed in the paper. Received results suggest, that fake news detection problem can be addressed with machine learning methods.

# Table Of Contents

<b>1.Introduction.....</b>	<b>6</b>
<b>2.Literature Survey.....</b>	<b>7-8</b>
2.1 Weakly Supervised Learning For Fake News Detection on Twitter	
2.2 Misleading Online Content	
<b>3.System Requirements.....</b>	
3.1 Hardware Requirement	
3.2 Software Requirement	
<b>4.Software Environment.....</b>	
4.1 Anaconda	
4.2 Jupyter	
<b>3.Data Collection.....</b>	<b>9</b>
<b>4. Data preprocessing.....</b>	<b>10-</b>
4.1 Importing Libraries	
4.2 Handling Missing Values	
4.3 Stop-word elimination	
4.4 Stemming/Lemmatization	
4.5 Features Extraction	
4.6 Splitting Dataset	
4.7 Prediction of the result	
4.8 Logistic Regresion Method	
<b>5.Result.....</b>	
<b>6.Test Data.....</b>	
<b>7.Conclusion and Future Work.....</b>	

## Introduction

These days“ fake news is creating different issues from sarcastic articles to a fabricated news and plan government propaganda in some outlets. Fake news and lack of trust in the media are growing problems with huge ramifications in our society. Obviously, a purposely misleading story is “fake news “ but lately blathering social media“s discourse is changing its definition. Some of them now use the term to dismiss the facts counter to their preferred viewpoints.

The importance of disinformation within American political discourse was the subject of weighty attention , particularly following the American president election . The term 'fake news' became common parlance for the issue, particularly to describe factually incorrect and misleading articles published mostly for the purpose of making money through page views. In this paper,it is seeked to produce a model that can accurately predict the likelihood that a given article is fake news.Facebook has been at the epicenter of much critique following media attention. They have already implemented a feature to flag fake news on the site when a user sees“s it ; they have also said publicly they are working on to to distinguish these articles in an automated way. Certainly, it is not an easy task. A given algorithm must be politically unbiased – since fake news exists on both ends of the spectrum – and also give equal balance to legitimate news sources on either end of the spectrum. In addition, the question of legitimacy is a difficult one.However, in order to solve this problem, it is necessary to have an understanding on what Fake News.

## OVERVIEW OF PROJECT

With the advancement of technology, digital news is more widely exposed to users

globally and contributes to the increment of spreading and disinformation online. Fake news can be found through popular platforms such as social media and the Internet. There have been multiple solutions and efforts in the detection of fake news where it even works with tools. However, fake news intends to convince the reader to believe false information which deems these articles difficult to perceive. The rate of producing digital news is large and quick, running daily at every second, thus it is challenging for machine learning to effectively detect fake news

## **II . Literature survey**

The available literature has described many automatic detection techniques of fake news and deception posts. Since there are multidimensional aspects of fake news detection ranging from using chatbots for spread of misinformation to use of clickbaits for the rumor spreading . There are many clickbaits available in social media networks including facebook which enhance sharing and liking Proceedings of posts which in turn spreads falsified information. Lot of work has been done to detect falsified information.

### ***WEAKLY SUPERVISED LEARNING FOR FAKE NEWS DETECTION ON TWITTER***

The problem of automatic detection of fake news in social media, e.g., on Twitter, has recently drawn some attention. Although, from a technical perspective, it can be regarded as a straight-forward, binary classification problem, the major challenge is the collection of large enough training corpora, since manual annotation of tweets as fake or non-fake news is an expensive and tedious endeavor. In this paper, we discuss a weakly supervised approach, which automatically collects a large-scale, but very noisy training dataset comprising hundreds of thousands of tweets. During collection, we automatically label tweets by their source, i.e., trustworthy or untrustworthy source, and train a classifier on this dataset. We then use that classifier for a different classification target, i.e., the classification of fake and non-fake tweets. Although the labels are not accurate according to the new classification target (not all tweets by an untrustworthy source need to be fake news, and vice versa), we show that despite this unclean inaccurate dataset, it is possible to detect fake news with an F1 score of up to 0.9.

## ***MISLEADING ONLINE CONTENT***

Tabloid journalism is often criticized for its propensity for exaggeration, sensationalization, scare-mongering, and otherwise producing misleading and low quality news. As the news has moved online, a new form of tabloidization has emerged: „clickbaiting.“ „Clickbait“ refers to “content whose main purpose is to attract attention and encourage visitors to click on a link to a particular web page” (clickbait) and has been implicated in the rapid spread of rumor and misinformation online. This paper examines potential methods for the automatic detection of clickbait as a form of deception. Methods for recognizing both textual and non-textual clickbaiting cues are surveyed, leading to the suggestion that a hybrid approach may yield best results.

Big Data Analytics and Deep Learning are two high-focus of data science. Big Data has become important as many organizations both public and private have been collecting massive amounts of domain-specific information, which can contain useful information about problems such as national intelligence, cyber security, fraud detection, marketing, and medical informatics. Companies such as Google and Microsoft are analyzing large volumes of data for business analysis and decisions, impacting existing and future technology. Deep Learning algorithms extract high-level, complex abstractions as data representations through a hierarchical learning process. Complex abstractions are learnt at a given level based on relatively simpler abstractions formulated in the preceding level in the hierarchy. A key benefit of Deep Learning is the analysis and learning of massive amounts of unsupervised data, making it a valuable tool for Big Data Analytics where raw data is largely unlabeled and un-categorized. In the present study, we explore how Deep Learning can be utilized for addressing some important problems in Big Data Analytics, including extracting complex patterns from massive volumes of data, semantic indexing, data tagging, fast information retrieval, and simplifying discriminative tasks. We also investigate some aspects of Deep Learning research that need further exploration to incorporate specific challenges introduced by Big Data Analytics, including streaming data, high-dimensional data, scalability of models, and distributed computing. We conclude by presenting insights into relevant future works by posing some questions, including defining data sampling criteria, domain adaptation modeling, defining criteria for obtaining useful data abstractions, improving semantic indexing, semi-supervised learning, and active learning.



## **SYSTEM REQUIREMENTS**

### **HARDWARE REQUIREMENTS:**

- System – AMD-A9
- Memory – 3.7 GiB
- Graphics – AMDr stoney
- Disk Capacity-500 GB

### **SOFTWARE REQUIREMENTS:**

- Operating System - UBUNTU 18.04
- Coding language – PYTHON

## SOFTWARE ENVIRONMENT:

### ANACONDA :

Anaconda constrictor is bundle director. Jupyter is an introduction layer. Boa constrictor endeavors to explain the reliance damnation in python where distinctive tasks have diverse reliance variants, in order to not influence distinctive venture conditions to require diverse adaptations, which may meddle with one another. Jupyter endeavors to fathom the issue of reproducibility in investigation by empowering an iterative and hands-on way to deal with clarifying and imagining code by utilizing rich content documentations joined with visual portrayals, in a solitary arrangement. Boa constrictor is like pyenv, venv and miniconda, it's intended to accomplish a python situation that is 100% reproducible on another condition, autonomous of whatever different forms of a task's conditions are accessible. It's somewhat like Docker, however limited to the Python biological system. Jupyter is an astounding introduction device for expository work, where we can display code in squares, joins with rich content depictions among squares, and the consideration of organized yield from the squares, and charts created in an all around planned issue by method for another square's code. Jupyter is extraordinarily great in expository work to guarantee reproducibility in somebody's exploration, so anybody can return numerous months after the fact and outwardly comprehend what somebody attempted to clarify and see precisely which code drove which representation and end. Regularly in diagnostic work we will finish up with huge amounts of half-completed note pads clarifying Proof-of-Concept thoughts, of which most won't lead any place at first.

## JUPYTER NOTEBOOK:

The Jupyter Notebook App is a server-customer application that permits altering and running note pad records by means of an internet browser. The Jupyter Notebook App can be executed on a nearby work area requiring no web access as portrayed in this report or can be introduced on a remote server and got to through the web. A scratch pad part is a computational motor that executes the code contained in a Notebook record. When you open a Notebook report, the related part is consequently propelled. At the point when the scratch pad is executed either cell-by-cell, the portion plays out the calculation and produces the outcomes. Contingent upon the sort of calculations, the piece may expend critical CPU and RAM. Note that the RAM isn't discharged until the part is closed down, the Notebook Dashboard is the part which is indicated first when you dispatch Jupyter Notebook App. The Notebook Dashboard is essentially used to open note pad archives, and to deal with the running portions. The Notebook Dashboard has different highlights like a record director, in particular exploring organizers, renaming and erasing documents.

## **Data collection**

In this study, we utilized the reference dataset ".traincsv" for analyzing the news is fake or real. The dataset contains the 20800 type of news with the 5 features. The features are id, title of the news, author of the news, content of the news and finally the labels for final reference. In the train.csv dataset some percentage is used for training set and remaining for the testing set. Finally based on the combination of title and text of the news we predict that the news is fake or real

Jupyter python ml Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

```
[79]: import pandas as pd
```

```
[4]: df=pd.read_csv("train.csv")
```

```
[5]: df
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Aistr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1
...	...	...	...	...	...
20795	20795	Rapper T.I.: Trump a 'Poster Child For White S...	Jerome Hudson	Rapper T. I. unloaded on black celebrities who...	0
20796	20796	N.F.L. Playoffs: Schedule, Matchups and Odds - ...	Benjamin Hoffman	When the Green Bay Packers lost to the Washing...	0
20797	20797	Macy's Is Said to Receive Takeover Approach by...	Michael J. de la Merced and Rachel Abrams	The Macy's of today grew from the union of sev...	0
20798	20798	NATO, Russia To Hold Parallel Exercises In Bal...	Alex Ansary	NATO, Russia To Hold Parallel Exercises In Bal...	1
20799	20799	What Keeps the F-35 Alive	David Swanson	David Swanson is an author, activist, journa...	1

20800 rows x 5 columns

## Data preprocessing

Preprocessing is a crucial step in many machine learning tasks, including fake news detection. It involves preparing and cleaning the raw data before feeding it into a machine learning model. Here's why preprocessing is important in fake news detection:

1. **Text Cleaning:** Fake news detection often deals with textual data from news articles, headlines, etc. Text data can contain various forms of noise such as punctuation, special characters, and irrelevant words. Preprocessing helps remove these noise elements, making the text more consistent and easier for the model to understand.
2. **Tokenization:** Tokenization involves breaking down text into individual words or tokens. This step is essential to convert text into a format that a machine learning model can work with effectively. Each token becomes a feature that the model can learn from.
3. **Stopword Removal:** Stopwords are common words like "and," "the," "is," etc., that don't carry much meaning. Removing them can help reduce the dimensionality of the data and improve model efficiency.
4. **Stemming and Lemmatization:** Stemming and lemmatization are techniques that reduce words to their base or root forms. This helps the model treat similar words as the same, which is especially important when words appear in different forms (e.g., "running" and "run").
5. **Feature Extraction:** Preprocessing prepares the text data for feature extraction. This could involve techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embeddings, which convert text data into numerical representations that the model can work with.
6. **Handling Imbalanced Data:** Fake news datasets can sometimes be imbalanced, with more instances of one class (real news) than the other (fake news). Preprocessing techniques like oversampling, undersampling, or generating synthetic samples can help balance the dataset and improve model performance.
7. **Data Normalization/Scaling:** If your model uses numerical features alongside text data, preprocessing can include normalizing or scaling these features to ensure that they are on a similar scale. This can improve the training process and model performance.
8. **Handling Missing Data:** Preprocessing addresses missing values in the dataset by imputing or removing them in a way that doesn't negatively impact the model's performance.

9. **Reducing Dimensionality:** High-dimensional data can lead to the curse of dimensionality. Preprocessing methods like feature selection or dimensionality reduction (e.g., PCA) can help reduce the number of features while retaining essential information

## Importing Libraries

When working on fake news detection using machine learning, you'll need to import various libraries to handle data manipulation, text preprocessing, modeling, and evaluation. Here are some common libraries you might need to import:

```
39]: import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

40]: import nltk
nltk.download('stopwords')
```

- `pandas` and `numpy`: For data manipulation and handling tabular data.
- `re`: For regular expressions to clean and preprocess text data.
- `stopwords` and `PorterStemmer` from `nltk`: For text preprocessing, removing stopwords, and stemming words.
- `TfidfVectorizer` from `sklearn`: For converting text data into numerical features using TF-IDF.
- `train_test_split` from `sklearn`: For splitting the dataset into training and testing sets.
- `accuracy_score` from `sklearn.metrics`: For evaluating model performance.
- `LogisticRegression` from `sklearn`: For building machine learning models.

# Handling Null Values

**Identify Null Values:** Use methods like `isnull()` or `info()` to identify which columns have missing values.

1. **Imputation or Removal:** Depending on the context and the dataset, you have a few options:

- **Imputation:** Fill the missing values with a suitable value. For numerical features, you might use the mean, median, or a custom value. For categorical features, you might use the mode or a placeholder value like "unknown."
- **Removal:** If the missing values are a small portion of the data and don't carry crucial information, you can remove the rows or columns with missing values.

2. **Handling Text Data:** If you're dealing with text data, missing values might be represented as empty strings. In such cases, you can consider imputing them with a placeholder word or removing them.

```
: news_dataset.isnull().sum()
```

```
: id          0
  title      558
  author    1957
  text       39
  label      0
  dtype: int64
```

```
: news_dataset=news_dataset.fillna(' ')
```

```
: news_dataset.isnull().sum()
```

```
: id          0
  title      0
  author      0
  text        0
  label      0
  dtype: int64
```

## Stop Word Elimination

Stop-word elimination Stop-words are functional terms that appear often in the text's language (for example, "a," "the," "an," and "of" in the English language), making them unusable for categorization. We reduce stop- words by utilizing the Natural Language Toolkit package. We don't want these terms to eat up important storage space or processing time in our database. This is

simply accomplished by keeping a list of words that you regard as stop words. In Python, NLTK stores a list of stopwords in 16 distinct languages.

```
[7]: import nltk
      nltk.download('stopwords')

[nltk_data] Downloading package stopwords to
[nltk_data] /home/rgukt123/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

[7]: True

[8]: print(stopwords.words('english'))

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

## Stemming/Lemmatization

Stemming is the process of reducing inflected words to their root (or stem) in information retrieval, such that similar terms map to the same stem. This approach decreases the number of words linked with each document organically, therefore reducing the feature space. In our tests, we employ a Porter stemming algorithm implementation. For instance, the English word "generalizations" would be stemmed as "generalizations → generalization → generalize → general → gener".



```
[104]: port_stem=PorterStemmer()

[105]: def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]', ' ', content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not word in stopwords.words('english')]
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content

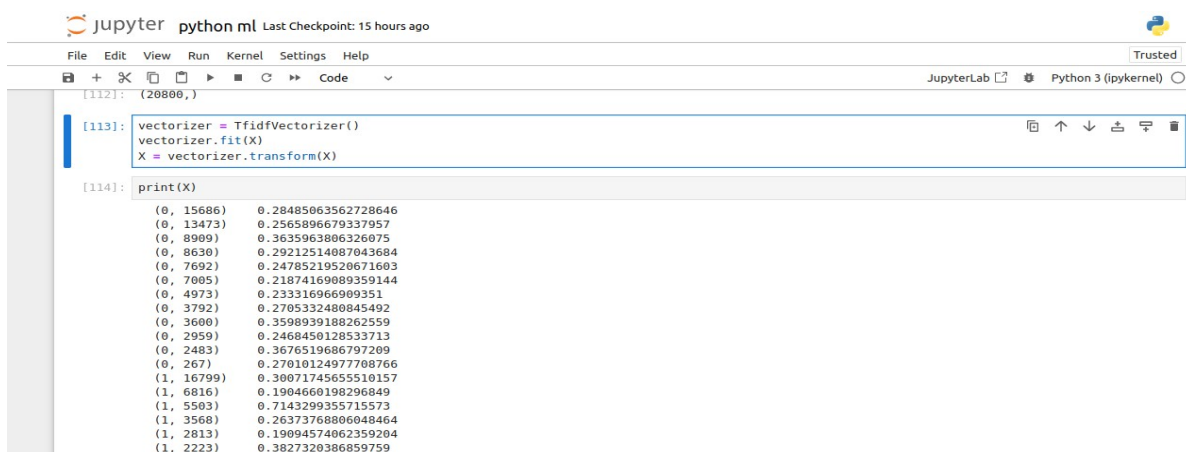
[117]: news_dataset['contenttt'] = news_dataset['contenttt'].apply(stemming)

[107]: print(news_dataset['contenttt'])

0      darrel lucu hous dem aid even see come letter...
1      daniel j flynn flynn hillari clinton big woman...
2      consortiumnew com truth might get fire
3      jessica purkiss civilian kill singl us airstri...
4      howard portnoy iranian woman jail fiction unpu...
...
20795  jerom hudson rapper trump poster child white s...
20796  benjamin hoffman n f l playoff schedul matchu...
20797  michael j de la merc rachel abram maci said re...
20798  alex ansari nato russia hold parallel exercis ...
20799  david swanson keep f aliv
Name: contenttt, Length: 20800, dtype: object
```

## Feature Extraction

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used for feature extraction from text data. It transforms a collection of documents (text data) into a numerical representation that can be used for machine learning models. TF-IDF takes into account the importance of each word in a document relative to the entire corpus. Here's how TF-IDF feature extraction works



The screenshot shows a JupyterLab window with a Python 3 kernel. The code in the first cell defines a `TfidfVectorizer` and fits it to a dataset `X`. The second cell prints the resulting matrix `X`.

```
[112]: (20800,)
```

```
[113]: vectorizer = TfidfVectorizer()
vectorizer.fit(X)
X = vectorizer.transform(X)
```

```
[114]: print(X)
```

(0, 15686)	0.28485063562728646
(0, 13473)	0.2565896679337957
(0, 8909)	0.3635963806326075
(0, 8630)	0.29212514087043684
(0, 7692)	0.24785219520671603
(0, 7005)	0.21874169089359144
(0, 4973)	0.233316966909351
(0, 3792)	0.2705332480845492
(0, 3600)	0.3598939188262559
(0, 2959)	0.2468450128533713
(0, 2483)	0.3676519686797209
(0, 267)	0.27010124977708766
(1, 16799)	0.3007174565510157
(1, 6816)	0.1904660198296849
(1, 5503)	0.7143299355715573
(1, 3568)	0.26373768806048464
(1, 2813)	0.19094574062359204
(1, 2223)	0.3827320386859759

After executing these lines of code, you will have `X` transformed into a TF-IDF representation, where each row of `X` represents a document (news article), and each column represents a unique term from the entire corpus of documents. The values in `X` represent the TF-IDF scores, indicating

the importance of each term in each document. This TF-IDF matrix is often used as input for machine learning algorithms in text-related tasks.

## Splitting Dataset

Splitting the data set into two halves is a critical component of the Machine Learning model. Machine Learning's primary objective is to generalize beyond the data examples used to train models. We wish to test the model to determine the quality of its pattern generalization on untrained data. However, because future instances will have unknown target values and we will be unable to verify the accuracy of our predictions for future instances at this time, we will need to use some of the data for which we already know the answer as a proxy for future data, which will be referred to as our Test Set. When dealing with big datasets, the most common method is to divide them into training and test subsets, often with a ratio of 70-80% for training and 20-30% for testing. The Train test split function, which is loaded from the scikit-learn package, does this splitting randomly.

### Training Set :

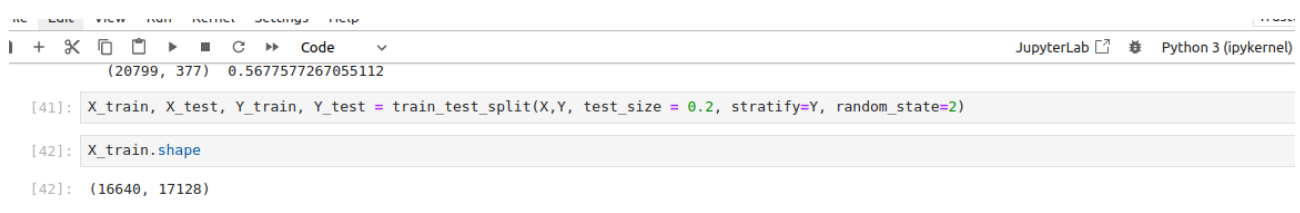
We've incorporated 80% of the data from 20800 reviews into our train set. Both the independent variable (x\_train) and the dependent variable (y\_train) are known in the training set.

### Testing Set:

The test set contains 20% of the data from 20800 reviews, where the dependent variable is denoted by (x\_test) and the independent variable is denoted by (y\_test).

### Splitting:

from sklearn.model\_selection import train\_test\_split



```

(20799, 377) 0.5677577267055112

[41]: X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)

[42]: X_train.shape

[42]: (16640, 17128)
```

## Prediction of the Result

The Machine Learning system utilizes the training data to train models to recognize patterns, and the test data to assess the trained model's prediction ability. The machine learning system measures

predictive performance by comparing predicted values on the evaluation data set to real values using a number of criteria. We will use Logistic Regression method to forecast.

```
[45]: X_train_prediction=model.predict(X_train)
      training_data_accuracy = accuracy_score(X_train_prediction,Y_train)

[46]: print('Accuracy score of the training data:',training_data_accuracy)
      Accuracy score of the training data: 0.9865985576923076

[47]: X_test_prediction=model.predict(X_test)
      test_data_accuracy = accuracy_score(X_test_prediction,Y_test)

[48]: print('Accuracy score of the testing data:',test_data_accuracy)
      Accuracy score of the testing data: 0.9790865384615385

[49]: y_pred=y_test[0]
```

## LOGISTIC REGRESSION ALGORITHM

Logistic Regression is a type of classification algorithm used for binary classification problems, where the goal is to predict one of two possible outcomes (in your case, whether a news article is fake or real). Despite its name, logistic regression is used for classification, not regression.

1. **Sigmoid Function:** Logistic Regression uses the logistic (or sigmoid) function to model the relationship between the input features and the binary outcome. The sigmoid function maps any input to a value between 0 and 1, which can be interpreted as a probability.
2. **Decision Boundary:** The logistic regression model learns a decision boundary that separates the two classes (fake and real news). This boundary is represented by a linear equation that combines the input features with their respective coefficients.
3. **Training:** During training, the logistic regression algorithm adjusts the coefficients of the linear equation to minimize a cost function (usually the cross-entropy loss) that measures the difference between the predicted probabilities and the actual labels.
4. **Prediction:** After training, the logistic regression model can make predictions by calculating the probability that a new input belongs to a particular class. If the probability is greater than a chosen threshold (often 0.5), the model predicts the positive class; otherwise, it predicts the negative class.

In our project:

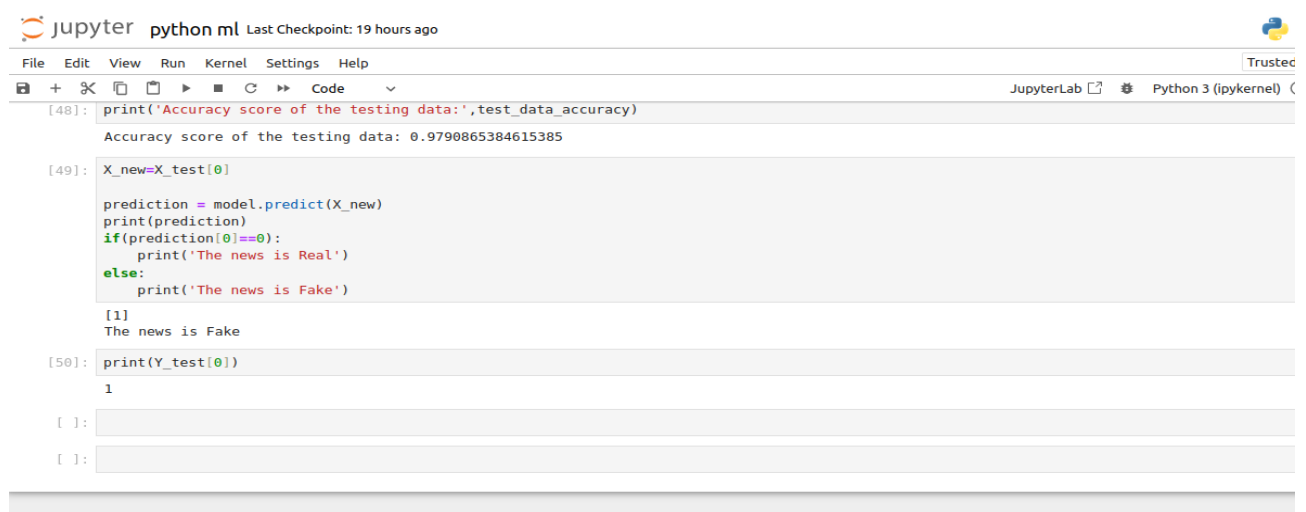
- We preprocess the text data and transform it into TF-IDF features.
- We split the data into training and testing sets.
- We train a Logistic Regression model on the training data using `model.fit(X_train, Y_train)`.

- We predict the labels of both the training data and testing data using `model.predict(X_train)` and `model.predict(X_test)`.
- We evaluate the model's accuracy on both the training and testing sets to check for overfitting or underfitting.

## RESULT

In the beginning, we analyzed the "train.csv" file, which contains 20800 different news. Then to predict the data is real or fake we combine the title and author column. Because to make predictions on text is difficult as it has a lot of data. Then we did some preprocessing on the data to give the data into a valid form for machine learning algorithms. Some of the operations we performed to preprocess the data are handling null values, removal of stopwords, stemming, splitting data, making prediction and finally testing the data. After learning the patterns from training data by machine through the Logistic Regression method, the output is given to the testing data. If the data is real then the output label is 0 otherwise, it is 1.

## Give Test Data



```
jupyter python ml Last Checkpoint: 19 hours ago
File Edit View Run Kernel Settings Help Trusted
[48]: print("Accuracy score of the testing data:", test_data_accuracy)
      Accuracy score of the testing data: 0.9790865384615385

[49]: X_new=X_test[0]
      prediction = model.predict(X_new)
      print(prediction)
      if(prediction[0]==0):
          print('The news is Real')
      else:
          print('The news is Fake')

[1]
The news is Fake

[50]: print(Y_test[0])
      1

[ ]:
[ ]:
```

## Conclusion and Future Work

Many people consume news from social media instead of traditional news media. However, social media has also been used to spread fake news, which has negative impacts on individual people and society. In this paper, an innovative model for fake news detection using machine learning algorithms has been presented. This model takes news events as an input and based on twitter reviews and classification algorithms it predicts the percentage of news being fake or real.

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.