**PRACTICAL NO-: 1**

**AIM-:** Fake news prediction system**.**

**INSTRUMENT REQUIRED-:** Laptop, Python compiler(interpreter),Power

**ABSTRACT-:**

Fake news prediction system is that type of system that predict whether the news is fake or real. For this we need a python interpreter and through the importing libraries we make this prediction system in this system there are following step are in included.

**STEP-1 -:**

In step 1 we import the libraries that’s are following-:

1. import numpy as np
2. import pandas as pd
3. import re
4. from nltk.corpus import stopwords
5. from nltk.stem.porter import PorterStemmer
6. from sklearn.feature\_extraction.text import TfidfVectorizer
7. from sklearn.model\_selection import train\_test\_split
8. from sklearn.linear\_model import LogisticRegression
9. from sklearn.metrics import accuracy\_score
10. import nltk

. now download the stopword that are use in the

Project for removing the not meaning data.

. nltk.download('stopwords')

**PRACTICAL NO-:2**

**DATA PRE-PROCESSING**

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

news\_dataset = pd.read\_csv('train.csv')

COUNTING THE NUMBER OF MISSING VALUES-:

news\_dataset.isnull().sum()

* REPLACING THE NULL VALUES WITH EMPTY STRING-:

news\_dataset = news\_dataset.fillna('')

* MERGING THE AUTHOR NAME AND TITLE FOR PRIDICTION OF NEWS

news\_dataset['content']=news\_dataset['author']+''+news\_dataset['title']

* SEPERATING THE DATA AND LABEL-:

x=news\_dataset.drop(columns='label',axis=1)

y=news\_dataset['label']

**PRACTICAL NO. :-3**

**STEMMING:-** Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words ***eating, eats, eaten*** is ***eat***.

Search engines use stemming for indexing the words. That’s why rather than storing all forms of a word, a search engine can store only the stems. In this way, stemming reduces the size of the index and increases retrieval accuracy.

* port\_stem = PorterStemmer()

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

* APPLYING THE THIS FUNCTION TO 'CONTENT' COLUMN

news\_dataset['content'] = news\_dataset['content'].apply(stemming)

* print(news\_dataset['content'] )
* x=news\_dataset['content'].values
* y=news\_dataset['label'].values

**PRACTICAL NO. :-4**

**CONVERTING TEXTUAL DATA TO NUMERIC DATA-:**

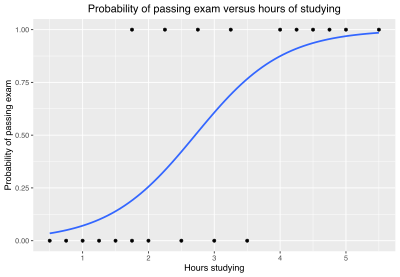
it is the process of converting the textual data into numeric data.because textual data not understandable by computer:

* vectorizer = TfidfVectorizer()
* vectorizer.fit(x)
* x=vectorizer.transform(x)
* SPLITTING THE DATASET TO TARINING AND TEST DATA:
* x\_train, X\_test, y\_train, Y\_test = train\_test\_split(x, y, test\_size = 0.2, stratify=y, random\_state=2)

**PRACTICAL NO. :-5**

**TRAINING THE MODEL : LOGISTIC REGRESSION**

Logit model" redirects here. Not to be confused with [Logit function](https://en.wikipedia.org/wiki/Logit_function).

[](https://en.wikipedia.org/wiki/File:Exam_pass_logistic_curve.svg)

Example graph of a logistic regression curve fitted to data. The curve shows the probability of passing an exam (binary dependent variable) versus hours studying (scalar independent variable). See  [Example](https://en.wikipedia.org/wiki/Logistic_regression#Example) for worked details.

In [statistics](https://en.wikipedia.org/wiki/Statistics), the logistic model (or logit model) is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that models the [probability](https://en.wikipedia.org/wiki/Probability) of an event taking place by having the [log-odds](https://en.wikipedia.org/wiki/Log-odds) for the event be a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of one or more [independent variables](https://en.wikipedia.org/wiki/Independent_variable). In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), logistic regression[[1]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-1) (or logit regression) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (the coefficients in the linear combination). Formally, in binary logistic regression there is a single [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), coded by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labeled "0" and "1", while the [independent variables](https://en.wikipedia.org/wiki/Independent_variable) can each be a binary variable (two classes, coded by an indicator variable) or a [continuous variable](https://en.wikipedia.org/wiki/Continuous_variable) (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling;[[2]](https://en.wikipedia.org/wiki/Logistic_regression#cite_note-Hosmer-2) the function that converts log-odds to probability is the [logistic function](https://en.wikipedia.org/wiki/Logistic_function), hence the name. The [unit of measurement](https://en.wikipedia.org/wiki/Unit_of_measurement) for the log-odds scale is called a [logit](https://en.wikipedia.org/wiki/Logit), from logistic unit, hence the alternative names. See [Background](https://en.wikipedia.org/wiki/Logistic_regression#Background) and [Definition](https://en.wikipedia.org/wiki/Logistic_regression#Definition) for formal mathematics, and [Example](https://en.wikipedia.org/wiki/Logistic_regression#Example) for a worked example.

* model=LogisticRegression()
* model.fit(x\_train,y\_train)

**PRACTICAL NO. :-6**

**MAKING A PRIDICTIVE SYSTEM:-**

* in this system we make the predictive system that check weather the news are fake or true

X\_new = X\_test[9]

prediction = model.predict(X\_new)

print(prediction)

if (prediction[0]==0):

print('The news is Real')

else:

print('The news is Fake')

**CODING:-**

#IMPORTING THE DEPENDENCIES:

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

#DOWNLOAD THE LIBRARY.'STOPWORDS'

import nltk

nltk.download('stopwords')

print(stopwords.words('english'))

#DATA PRE-PROCESSING

news\_dataset = pd.read\_csv('train.csv')

news\_dataset.shape

news\_dataset.head()

#COUNTING THE NUMBER OF MISSING VALUES

news\_dataset.isnull().sum()

#REPLACING THE NULL VALUES WITH EMPTY STRING

news\_dataset = news\_dataset.fillna('')

#MERGING THE AUTHOR NAME AND TITLE FOR PRIDICTION OF NEWS

news\_dataset['content']=news\_dataset['author']+''+news\_dataset['title']

print(news\_dataset['content'])

#SEPERATING THE DATA AND LABEL

x=news\_dataset.drop(columns='label',axis=1)

y=news\_dataset['label']

print(x)

print(y)

#STEMMING

port\_stem = PorterStemmer()

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

#APPLYING THE THIS FUNCTION TO 'CONTENT' COLUMN

news\_dataset['content'] = news\_dataset['content'].apply(stemming)

print(news\_dataset['content'] )

x=news\_dataset['content'].values

y=news\_dataset['label'].values

print(x)

print(y)

#COVERTING TEXTUAL DATA TO NUMERIC DATA

vectorizer = TfidfVectorizer()

vectorizer.fit(x)

x=vectorizer.transform(x)

print(x)

#SPLITTING THE DATASET TO TARINING AND TEST DATA:

x\_train, X\_test, y\_train, Y\_test = train\_test\_split(x, y, test\_size = 0.2, stratify=y, random\_state=2)

#training the model:logistic regression

model=LogisticRegression()

model.fit(x\_train,y\_train)

#EVALUTION:

#ACCURACY SCORE ON TRAINING DATA

# accuracy score on the training data

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print('Accuracy score of the training data : ', training\_data\_accuracy)

Accuracy score of the training data : 0.9719350961538461

# accuracy score on the test data

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print('Accuracy score of the test data : ', test\_data\_accuracy)

Accuracy score of the test data : 0.9548076923076924

OUTPUT

#MAKING A PRIDICTIVE SYSTEM:

X\_new = X\_test[9]

prediction = model.predict(X\_new)

print(prediction)

if (prediction[0]==0):

print('The news is Real')

else:

print('The news is Fake')

print(Y\_test[9])

Accuracy score of the training data : 0.9719350961538461

|  |
| --- |
|  |