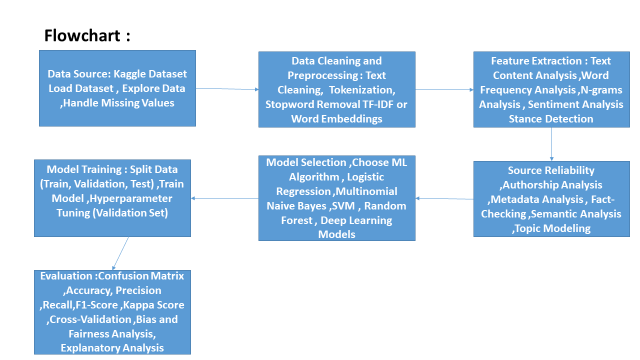
**Fake News Detection Using NLP**

**Problem Definition:** The problem is to develop a fake news detection model using a Kaggle dataset. The goal is to distinguish between genuine and fake news articles based on their titles and text. This project involves using natural language processing (NLP) techniques to preprocess the text data, building a machine learning model for classification, and evaluating the model's performance.

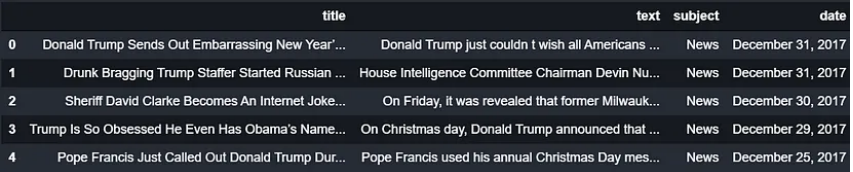
**Abstract:**

This study explores the use of Natural Language Processing (NLP) techniques to detect fake news articles. We employ text preprocessing, feature extraction, and machine learning algorithms to differentiate between genuine and fake news. Our approach demonstrates promising results in accurately identifying fake news, contributing to the ongoing effort to combat misinformation in the digital era.



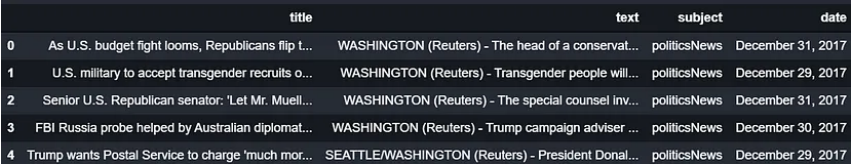
**Dataset for Fake news**

**Head()**



**Dataset for Real news**

**Head()**



**Classification:**

#splitting data for training and testing

import sklearn

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(data['text'],data['label'],test\_size=0.2, random\_state = 1)

**MultiNomial Naive Bayes**

Naive Bayes are mostly used in natural language processing. Naive Bayes classifier algorithm is a family of algorithms which use [Bayes Theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem#:~:text=In%20probability%20theory%20and%20statistics,be%20related%20to%20the%20event.). It uses the naive assumption that all the features are independent of each other. Bayes theorem calculates the probability P(c|x) where c is the class of possible outcomes and x is the given instance which has to be classified.

**P(c|x) = P(x|c) \* P(c) / P(x)**

**Support Vector Machine**

Support Vector Machine or SVM is a linear model for classification and regression problems. SVM model takes the data in the training set, and maps it to data points in space so that there is a clear gap between points belonging to different categories

**Passive Aggressive Classifier**

Passive aggressive classifier is an online algorithm that learns from massive streams of data.

**Count Vectorizer**

The count vectorizer tokenizes a collection of documents and builds a vocabulary of unique words. It can also encode new documents using this vocabulary.

**Tfidf Transformer**

Consider the following set of words-‘the’, ‘in’, ‘on’,‘a’, ‘an’. This is an example of a set of words which doesn’t have any meaning on it’s own, but they occur a lot in every document.We use 4 measures to evaluate the performance.

* True positive: The cases in which the predicted values and the actual values are the same, and the value is positive.
* True Negative: The cases in which the predicted values and the actual values are the same, and the value is negative.
* False Positive: The cases in which the prediction is ‘YES’ ,but the actual value is ‘NO’.
* False Negative: The cases in which the prediction is ‘NO’, but the actual value is ‘YES’.

#Multinomial NB

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.pipeline import Pipeline

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score

import sklearn.metrics as metrics

from mlxtend.plotting import plot\_confusion\_matrix

from sklearn.metrics import confusion\_matrix

pipe = Pipeline([ ('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf',MultinomialNB())])

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

prediction = model.predict(x\_test)

score = metrics.accuracy\_score(y\_test, prediction)

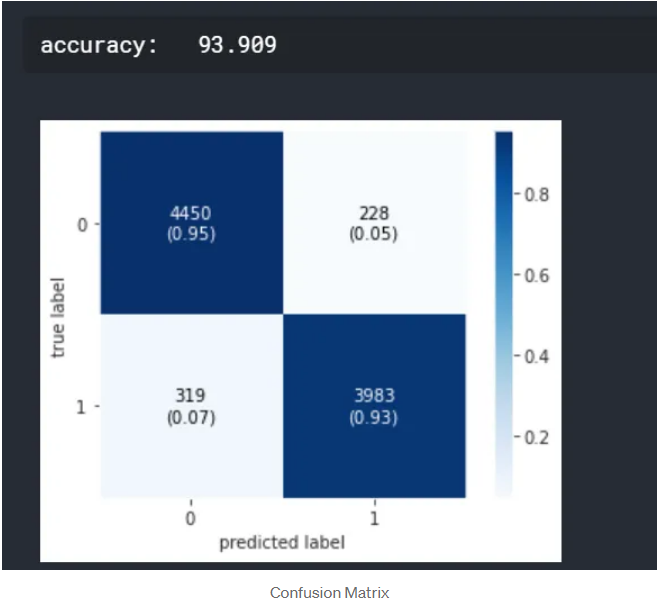
print("accuracy: %0.3f" % (score\*100))

cm = metrics.confusion\_matrix(y\_test, prediction, labels=[0,1])

fig, ax = plot\_confusion\_matrix(conf\_mat=confusion\_matrix(y\_test, prediction),

show\_absolute=True,show\_normed=True,colorbar=True)

plt.show()



#SVM

from sklearn.svm import LinearSVC

pipe = Pipeline([ ('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', LinearSVC())

])

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

prediction = model.predict(x\_test)

score = metrics.accuracy\_score(y\_test, prediction)

print("accuracy: %0.3f" % (score\*100))

cm = metrics.confusion\_matrix(y\_test, prediction, labels=[0,1])

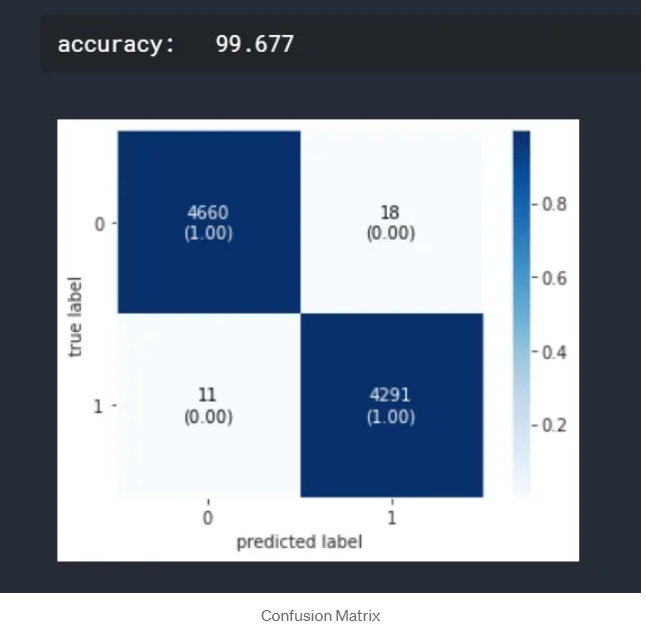
fig, ax = plot\_confusion\_matrix(conf\_mat=confusion\_matrix(y\_test, prediction),

show\_absolute=True,

show\_normed=True,

colorbar=True)

plt.show()



#Passive Aggressive Classifier

from sklearn.linear\_model import PassiveAggressiveClassifier

pipe = Pipeline([ ('vect', CountVectorizer()), ('tfidf', TfidfTransformer()), ('clf', PassiveAggressiveClassifier())

])

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

prediction = model.predict(x\_test)

score = metrics.accuracy\_score(y\_test, prediction)

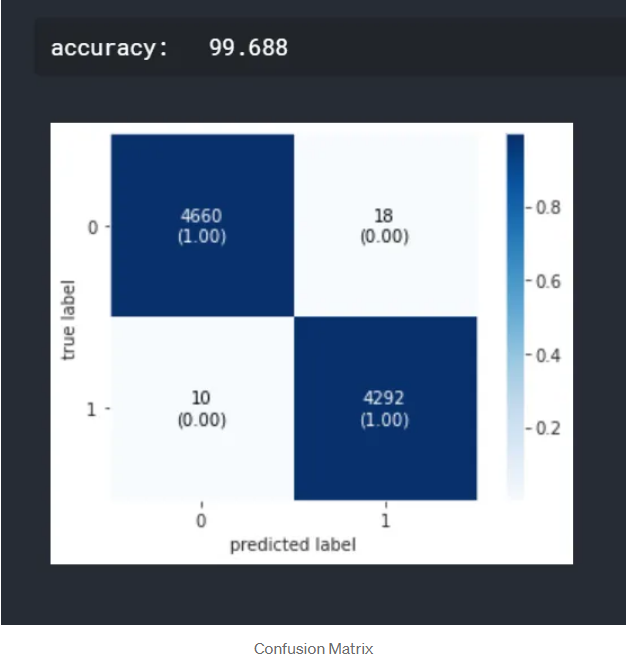
print("accuracy: %0.3f" % (score\*100))

cm = metrics.confusion\_matrix(y\_test, prediction, labels=[0,1])

fig, ax = plot\_confusion\_matrix(conf\_mat=confusion\_matrix(y\_test, prediction),

show\_absolute=True, show\_normed=True,colorbar=True)

plt.show()



**Conclusion**

We have classified our news data using three classification models. We have analysed the performance of the models using accuracy and confusion matrix. But this is only a beginning point for the problem. There are advanced techniques like BERT, GloVe and ELMo which are popularly used in the field of NLP. If you are interested in NLP, you can work forward with these techniques.