

# Fake news detection using nlp







# Fake News Detection:

- Fake News Detection is a natural language processing task that involves identifying and classifying news articles or other types of text as real or fake. The goal of fake news detection is to develop algorithms that can automatically identify and flag fake news articles, which can be used to combat misinformation and promote the dissemination of accurate information.



# Advanced Python – Detecting Fake News:

- By practicing this advanced python project of detecting fake news, you will easily make a difference between real and fake news. Before moving ahead in this advanced Python project, get aware of the terms related to it like fake news, tfidfvectorizer, PassiveAggressive Classifier.
- TF:(Term Frequency): The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.
- IDF:(Inverse Document Frequency): Words that occur many times a document, but also occur many times in many others, may be irrelevant. IDF is a measure of how significant a term is in the entire corpus.The TfidfVectorizer converts a collection of raw documents into a matrix of TF-IDF features.
- Passive Aggressive algorithms are online learning algorithms. Such an algorithm remains passive for a correct classification outcome, and turns aggressive in the event of a miscalculation, updating and adjusting. Unlike most other algorithms, it does not converge. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector.



## The Dataset:

- The dataset we'll use for this python project- we'll call it news.csv. This dataset has a shape of  $7796 \times 4$ . The first column identifies the news, the second and third are the title and text, and the fourth column has labels denoting whether the news is REAL or FAKE.





# Steps 1 for detecting fake news with Python:

- `import numpy as np`  
`import pandas as pd`  
`import matplotlib.pyplot as plt`  
`import matplotlib`  
`import seaborn as sns`  
`import itertools`  
`from sklearn.model_selection`  
`import train_test_split`  
`from sklearn.feature_extraction.text`  
`import TfidfVectorizer`  
`from sklearn.linear_model`  
`import PassiveAggressiveClassifier`  
`from sklearn.metrics`  
`import accuracy_score,`  
`confusion_matrix`  
`accuracy_score,`  
`confusion_matrix`


## 2) Now, let's read the data into a DataFrame, and get the shape of the data and the first 5 records:

```
• Unnamed: 0      title      text      label0
      8476      You Can Smell Hillary's Fear
      Daniel Greenfield, a Shillman Journalism
      Fello...  FAKE1      10294      Watch The Exact
      Moment Paul Ryan Committed Pol...      Google
      Pinterest Digg Linkedin Reddit Stumbleu...  FAKE2
      3608      Kerry to go to Paris in gesture of
      sympathy U.S. Secretary of State John F. Kerry said
      Mon...  REAL3      10142      Bernie supporters on
      Twitter erupt in anger ag...      — Kaydee King
      (@KaydeeKing) November 9, 2016 T...      FAKE4
      875      The Battle of New York: Why This
      Primary Matters      It's primary day in New York and
      front-runners...  REAL 5      The Battle of New
      York: Why This Primary Matters      It's primary day in
      New York and front-runners...  REAL
```



### 3) get the labels from the DataFrame:

- `Labels=df.labellabels.head()0`  
FAKE1 FAKE2 REAL3 FAKE4  
REALName: label, dtype: object
- `Target=df.label.value_counts()target`  
REAL 3171FAKE 3164Name: label,  
dtype: int64

A yellow pencil is positioned diagonally across the frame, resting on a blue-lined notebook page. The page features multiple-choice bubbles with letters A, B, C, and D. The text '4) Split the dataset into training and testing sets:' is overlaid in white, bold font.

## 4) Split the dataset into training and testing sets:

- `X_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.2, random_state=7)`



## 5) Initialize a TfidfVectorizer:

- from the English language and a maximum document frequency of 0.7 (terms with a higher document frequency will be discarded). Stop words are the most common words in a language that are to be filtered out before processing the natural language data. And a TfidfVectorizer turns a collection of raw documents into a matrix of TF-IDF features. Now, fit and transform the vectorizer on the train set, and transform the vectorizer on the test set.
- `Tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)`#DataFlair – Fit and transform train set, transform test  
`settfidf_train=tfidf_vectorizer.fit_transform(x_train)`  
`) tfidf_test=tfidf_vectorizer.transform(x_test)`



## 6) Initialize a PassiveAggressiveClassifier:

- `Pac=PassiveAggressiveClassifier(max_iter=50)`  
`pac.fit(tfidf_train,y_train)`  
#DataFlair – Predict on the test set and calculate accuracy  
`y_pred=pac.predict(tfidf_test)`  
`score=accuracy_score(y_test,y_pred)`  
`print(f'Accuracy: {round(score*100,2)}%')`  
Accuracy: 92.5%  
We got an accuracy of 92.82% with this model. Finally, let's print out a confusion matrix to gain insight into the number of false and true negatives and positives.



## 7) Confusion matrix:

- `Confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])`  
`array([[58, 50], [ 45, 584]])`

