**FAKE NEWS DETECTION USING NLP**

**BATCH NUMBER**

**AU950621104073**

Name: S.Rathika

Phase 3 submission document

**Project Title**: Fake news detection using NLP

**Phase 3**: Development Part 1

**Topic**: Fake news detection using NLP by loading and pre-processing the dataset.



**Fake news detection using NLP**

**Introduction:**

* We consume news through several mediums throughout the day in our daily routine, but sometimes it becomes difficult to decide which one is fake and which one is authentic.
* Every news that we consume is not real. If you listen to fake news it means you are collecting the wrong information from the world which can affect society because a person’s views or thoughts can change after consuming fake news which the user perceives to be true.
* We will focus on text-based news and try to build a model that will help us to identify if a piece of given news is fake or real.
* NLP is a branch of data science that consists of systematic processes for analyzing, understanding, and deriving information from the text data in a smart and efficient manner.

Terminologies:

**Fake news:**

A sort of sensationalist reporting, counterfeit news embodies bits of information that might be lies and is, for the most part, spread through web-based media and other online media.

This is regularly done to further or force certain kinds of thoughts or for false promotion of products and is frequently accomplished with political plans.

**Necessary step to follow:**

**Import Libraries:**

Start by importing the necessary libraries:

**Program:**

Import pandas as pd

Import numpy as np

from sklearn. model\_ selection import train\_ test\_ split

from sklearn. preprocessing import StandaredScaler

plt.style.use('Solarize\_Light2')

%matplotlib inline

**Load the Dataset:**

Load the dataset into a pandas DataFrame.

**Program:**

df=pd.read\_csv('fake-news/train.csv')

df.head()

**Output:**

| 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 Airstr... | 1 |
| 4 | 4 | Iranian woman jailed for fictional unpublished... | Howard Portnoy | Print \nAn Iranian woman has been sentenced to... | 1 |

**Program:**

df.shape()

**output:**

(20800, 5)

**Program:**

df.isnull().sum()

**Output:**

id 0

title 558

author 1957

text 39

label 0

dtype: int64

**Program:**

df.dropna(inplace = True)

df.isna().sum()

**Output:**

id 0

title 0

author 0

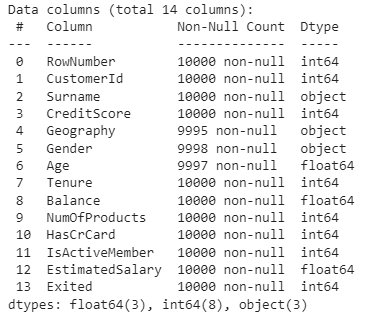
text 0

label 0

dtype: int64

dataset.info()

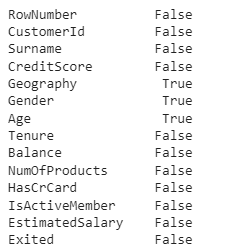
**Output :**



**Data Preprocessing:**

dataset.isnull().any()

**Output :**



**1.Text Cleaning:**

This involves removing any unnecessary characters, punctuation, or special symbols from the text. You can also convert the text to lowercase to ensure consistency.

**2. Tokenization:**

This step involves breaking down the text into individual words or tokens. This allows for easier analysis and processing of the text data.

**3. Stop Word Removal:**

Stop words are common words that don’t carry much meaning, such as “the”, ”is”, or “and”. Removing this words can help reduce noise in the data and improve the accuracy of the analysis.

**4. Lemmatization or Stemming:**

This step involves reducing words to their base or root form. For example, converting “running” to “run” to “cat”. This helps to consolidate similar words and improve the efficiency of the analysis.

**5. Feature Extraction:**

This transforming the text into numerical features that can be used for analysis. Technique such as TF-IDF(Term Frequency-Inverse Document Frequency).

**6. Vectorization:**

Vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which is used to find word predictions, word similarities/semantics.

**Program:**

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

count\_vectorizer = CountVectorizer()

count\_vectorizer.fit\_transform(x\_df)

freq\_term\_matrix = count\_vectorizer.transform(x\_df)

tfidf = TfidfTransformer(norm = "l2")

tfidf.fit(freq\_term\_matrix)

tf\_idf\_matrix = tfidf.fit\_transform(freq\_term\_matrix)

print(tf\_idf\_matrix)

**7. Modelling:**

After vectorization, we split the data into test and train data.

**Program:**

x\_train, x\_test, y\_train, y\_test = train\_test\_split(tf\_idf\_matrix,y\_df, random\_state=0)

**NLP Techniques :**

We area unit exploitation Pos-tagging and TF-IDF for word frequency scores that attempt to highlight words that area unit a lot of fascinating, e.g., frequent in a very document however not across documents.

Pos-tagging could be a method to price the words in text format for a selected a part of a speech supported its definition and context. it's chargeable for text reading in a very language and distribution some specific token (Parts of Speech) to every word.

The TfIdf-Vectorizer can tokenize documents, learn the vocabulary and inverse document frequency weightings, and permit you to cypher new documents. whereas coaching the dataset this is often important step.

Formula(Algorithm): TF-IDF = TF(t,d) \* IDF(t) ‘ TF’ stands for term frequency and ‘IDF’ stands for Inverse document frequency ,in this ‘t’ is a term frequency that is number of times t appears in a doc,’d’. IDF is calculated by : log[1+n/1+df(d,t)]+1

# Exploratory Data Analysis:

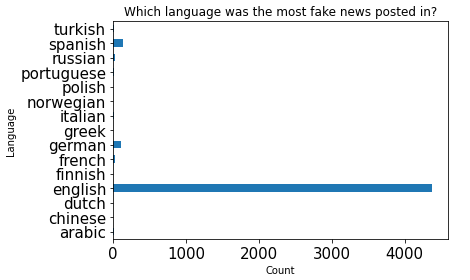


Exploratory Data Analysis(EDA) is the process of exploring data, generating insights, testing hypotheses, checking assumptions and revealing underlying hidden patterns in the data.

Performing thorough exploratory data analysis (EDA) and cleaning the dataset are not only essential steps, but also a great opportunity to lay the foundation for a strong machine learning model.

We have used a bar plot to answer this question from our dataset, plotting the bar plot consisted of making use of the df.plot() function from the pandas dataset in python.

The benefits of a bar plot are that it is an excellent tool to observe the values of a few number of data points but want to present the general trend.

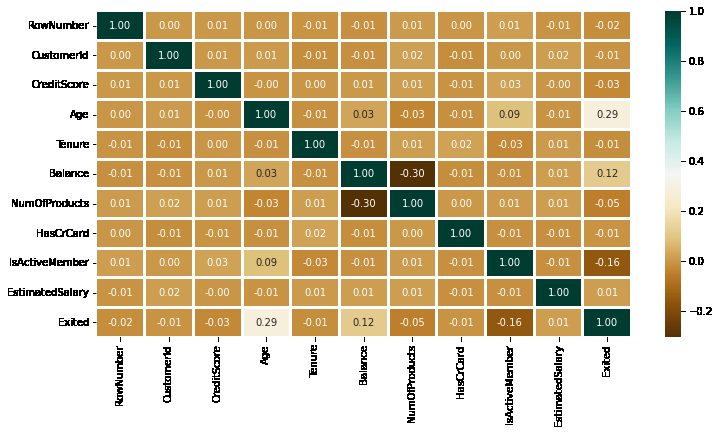


As the above plot shows, the main language represented in the dataset was English which was the main outlier with about 4507 data points.

The rest of the languages had a fairly even distribution among them.

|  |
| --- |
| Program:  plt.figure(figsize=(12,6))  sns.heatmap(dataset.corr(),              cmap='BrBG',              fmt='.2f',              linewidths=2,              annot=True) |

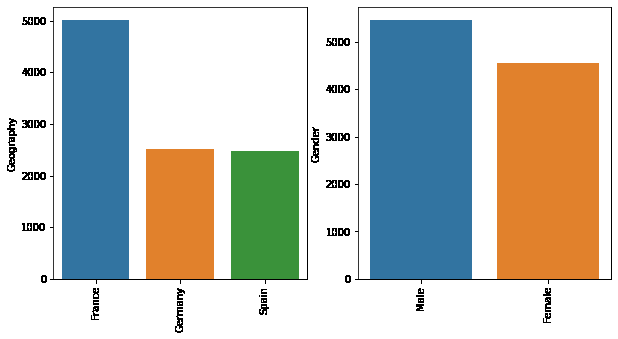
**Output :**



Program:

|  |
| --- |
| lis2 = ['Geography', 'Gender']  plt.subplots(figsize=(10, 5))  index = 1    for col in lis2:      y = dataset[col].value\_counts()      plt.subplot(1, 2, index)      plt.xticks(rotation=90)      sns.barplot(x=list(y.index), y=y)      index += 1 |

**Output :**



**Logistic Regression:**

**Logistic regression is** a data analysis technique that uses mathematics to find the relationships between two data factors.

**It then uses this relationship to predict the value of one of those factors based on the other. The prediction usually has a finite number of outcomes, like yes or no.**

Based on the analysis done, it was observed that logistic regression model achieved high accuracy, precision, recall, and F1 score in classifying news content as real or fake when compared with K-Nearest Neighbour (KNN), Passive Aggressive classi er and Naïve Bayes classi er.

**Program:**

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

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

Accuracy = logreg.score(x\_test, y\_test)

print(Accuracy\*100)

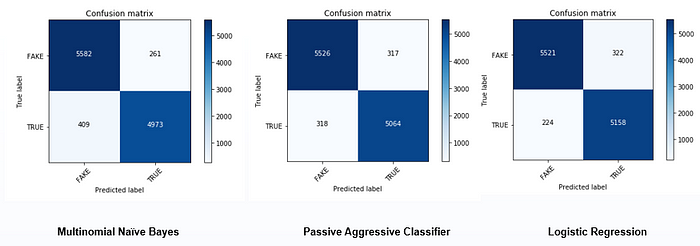
**Output:**

Accuracy: 91.73%

**Naïve baise:**

A probabilistic model known as the Naive Bayes classifier makes predictions based on the likelihood that particular occurrences will occur.

The events are the classes or categories to which news articles can be classified, and the features are the words or other characteristics of the articles.



**Program:**

from sklearn.naive\_bayes import MultinomialNB

classifier = MultinomialNB()

classifier.fit(X\_train, y\_train)

**Output:**

MultinomialNB()

**Program:**

y\_pred = classifier.predict(X\_test)

y\_pred

**Output:**

array(['0', '0', '0', ..., '0', '0', '0'], dtype='<U1')

**Program:**

from sklearn import metrics

metrics.accuracy\_score(y\_test, y\_pred)

**Output:**

0.9015748031496063

**Decision Tree:**

Decision trees are used to extract relevant features from the text data, and Ada boost is employed to enhance the performance of the decision trees.

The proposed method is evaluated on a dataset of news articles, and the results demonstrate its effectiveness in detecting fake news.

**Program:**

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

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

Accuracy = clf.score(x\_test, y\_test)

print(Accuracy\*100)

**Output:**

Accuracy: 80.49%

**Program:(Real values)**

feature\_names = cv.get\_feature\_names()

sorted(zip(classifier.coef\_[0],feature\_names),reverse=True)[0:20]

**Output:**

[(-3.9914839716490507, 'trump'),

(-4.279898913316484, 'hillari'),

(-4.348158915587783, 'clinton'),

(-4.88335015151613, 'elect'),

(-5.184527705442797, 'new'),

(-5.23658406739885, 'comment'),

(-5.3201316376692915, 'video'),

(-5.349607455802245, 'us'),

(-5.388828168955526, 'hillari clinton'),

(-5.406765869642194, 'war'),

(-5.429650163475781, 'fbi'),

(-5.45307043768388, 'email'),

(-5.4722097778945775, 'vote'),

(-5.579181897446745, 'world'),

(-5.579181897446745, 'obama'),

(-5.755072563910409, 'russia'),

(-5.828194828739372, 'donald'),

(-5.835115271583946, 'day'),

(-5.835115271583946, 'america'),

(-5.870454638029255, 'say')]

**Program:(Fake values)**

eature\_names = cv.get\_feature\_names()

sorted(zip(classifier.coef\_[0],feature\_names),reverse=True)[-20:]

**Output:**

[(-10.804928571159946, 'american new'),

(-10.804928571159946, 'american breitbart'),

(-10.804928571159946, 'america new york'),

(-10.804928571159946, 'america new'),

(-10.804928571159946, 'america breitbart'),

(-10.804928571159946, 'amazon'),

(-10.804928571159946, 'ali'),

(-10.804928571159946, 'aleppo new york'),

(-10.804928571159946, 'aleppo new'),

(-10.804928571159946, 'agenda breitbart'),

(-10.804928571159946, 'age new york'),

(-10.804928571159946, 'age new'),

(-10.804928571159946, 'advis new york'),

(-10.804928571159946, 'advis new'),

(-10.804928571159946, 'advic'),

(-10.804928571159946, 'act new york'),

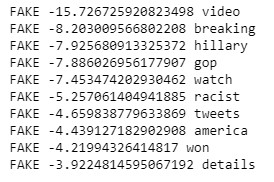
(-10.804928571159946, 'act new'),

(-10.804928571159946, 'abus new york'),

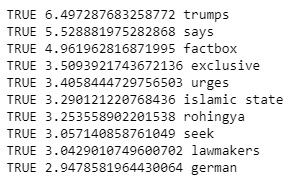
(-10.804928571159946, 'abus new'),

(-10.804928571159946, 'abroad')]

The top 10 key features in predicting fake news are shown below.



The top 10 key features in predicting true news are shown below.



In most of the top fake and true news, only unigrams are present while in case of true news one bigram feature is also present.

**Conclusion:**

Fake news detection techniques can be divided into those based on style and those based on content, or fact-checking. Too often it is assumed that bad style (bad spelling, bad punctuation, limited vocabulary, using terms of abuse, ungrammaticality, etc.) is a safe indicator of fake news.

More than ever, this is a case where the machine’s opinion must be backed up by clear and fully verifiable indications for the basis of its decision, in terms of the facts checked and the authority by which the truth of each fact was determined.