**FAKE NEWS DETECTION**

**USING**

**NATURAL LANGUAGE PROCESSING**

**PROJECT REPORT**

**SUBMITTED BY**

**ASWIN OK**

**INTRODUCTION**

Fake News Detection aims to categories the daily news to classify into real news or fake news using Natural Language Processing. It helps to prevent the spreading of fake news. This dataset contains over 40 thousand data. It makes the mode more accurate and precise so the prediction will be more accurate.

**GENERAL BACKGROUND**

1**. Data cleaning**: it is the process of removal of unwanted and noisy data from

the huge volume of data, cleansing is the other name of this process.

2. **Data integration**: it is the process of combining certain data which are

showing the same property.

3. **Data selection**: In this phase the data needed for the work is identified and

fetched out from the huge volume of data.

4**. Data transformation**: The process of translating data from one format to

another format, which is useful for data analysis purpose, data consolidation is

another name of data transformation.

5. **Data mining**: In the data mining step the whole dataset is traversed deeply to

collect useful information for the work.

6. **Pattern evaluation**: It is the process of finding out the pattern and relations

associated with each attribute and its instances.

7. **Knowledge representation**: The mining data is visually represented in this

step and, the interpretation of results is being done.

**CODE**

**IMPORTING NECCESARY LIBRARIES**

import pandas as pd

import nltk

from matplotlib import pyplot as plt

import seaborn as snc

from wordcloud import WordCloud,STOPWORDS

from PIL import Image

from nltk.corpus import wordnet,stopwords

from nltk.stem import WordNetLemmatizer

lemmatize=WordNetLemmatizer()

import re

from sklearn.feature\_extraction.text import CountVectorizer,TfidfVectorizer

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

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import MultinomialNB

from xgboost import XGBClassifier

from sklearn.model\_selection import GridSearchCV

**READING FILES**

fake=pd.read\_csv('Fake.csv')

fake['label']='fake'

real=pd.read\_csv(' True.csv')

real['label']='real'

df=pd.concat([fake,real],ignore\_index=True)

df.head()

**ANALYSING DATA**

df.info()

df.isnull().sum()

**PLOTTING**

label\_count=df['label'].value\_counts()

label\_count.plot(kind='bar',color=['blue','orange'])

plt.title('News Count')

plt.xlabel('Fake or Real')

plt.ylabel('Count')

plt.show()

plt.figure(figsize=(12,6))

snc.countplot(data=df,x='subject',hue='label')

plt.title('Subject Wise Clasiification')

plt.show()

**WORD CLOUD**

title=df['title']

title=str(title)

wordcloud=WordCloud(width=1000,height=1000).generate(title)

plt.imshow(wordcloud)

plt.axis('off')

plt.title('WordClod of Title')

plt.show()

text=df['text']

text=str(text)

wordcloud=WordCloud(width=1000,height=1000).generate(text)

plt.imshow(wordcloud)

plt.axis('off')

plt.title('WordClod of Text')

plt.show()

fake\_text=fake['text']

text=str(fake\_text)

wordcloud\_fake=WordCloud(width=1000,height=1000).generate(text)

plt.imshow(wordcloud\_fake)

plt.axis('off')

plt.title('WordClod of Fake News Text')

plt.show()

real\_text=real['text']

text=str(real\_text)

wordcloud\_real=WordCloud(width=1000,height=1000).generate(text)

plt.imshow(wordcloud\_real)

plt.axis('off')

plt.title('WordClod of Real News Text')

plt.show()

**WORD COUNT GRAPH**

def get\_top\_ngram(corpus, n=None):

  vec=CountVectorizer(ngram\_range=(n,n), stop\_words='english').fit(corpus )

  bag\_of\_words=vec.transform(corpus )

  sum\_words=bag\_of\_words.sum(axis=0)

  words\_freq=[(word,sum\_words[0,idx]) for word, idx in vec.vocabulary\_.items()]

  words\_freq=sorted(words\_freq,key=lambda x: x[1],reverse=True)

  return words\_freq[:10]

top\_n\_bigrams=get\_top\_ngram(df[df['label']=='fake']['text'],2)[:10]

x,y=map(list,zip(\*top\_n\_bigrams))

snc.barplot(x=y,y=x)

plt.title('Top 10 Frequent Words In Fake News')

top\_n\_bigrams=get\_top\_ngram(df[df['label']=='real']['text'],2)[:10]

x,y=map(list,zip(\*top\_n\_bigrams))

snc.barplot(x=y,y=x)

plt.title('Top 10 Frequent Words In Real News')

**DATA GROUPING**

fake1=fake[:2000]

real1=real[:2000]

df1=pd.concat([fake1,real1],ignore\_index=True)

df1.head()

y=df1.label

**PARTS OF SPEECH TAGGER**

text=[]

for i in df1.text:

    tk=nltk.sent\_tokenize(i)

    text.append(tk)

a1=[]

for i in range(len(text)):

  a=str(text[i])

  a=re.sub('[^a-zA-Z]',' ',a)

  pos\_tagged\_text=nltk.pos\_tag(nltk.word\_tokenize(a))

  a1.append(pos\_tagged\_text)

def pos\_tagger(nltk\_tag):

  nltk\_tag=str(nltk\_tag)

  if nltk\_tag.startswith('J'):

    return wordnet.ADJ

  elif nltk\_tag.startswith('V'):

    return wordnet.VERB

  elif nltk\_tag.startswith('N'):

    return wordnet.NOUN

  elif nltk\_tag.startswith('R'):

    return wordnet.ADV

  else:

    return None

wordnet\_tagged=list(map(lambda x: (x[0],pos\_tagger(x[1])),a1))

**LEMMATIZATION**

lemmatized\_sentence=[]

for word,tag in wordnet\_tagged:

  if word not in set(stopwords.words('english')):

    if tag is None:

      lemmatized\_sentence.append(word)

    else:

      lemmatized\_sentence.append(lemmatize.lemmatize(word, tag))

  else:

    pass

**COUNT VECTORIZER**

count\_vectorizer=CountVectorizer()

vector=count\_vectorizer.fit\_transform([str.join(' ', x) for x in lemmatized\_sentence])

feature\_name\_count=count\_vectorizer.get\_feature\_names\_out()

array\_count=vector.toarray()

x\_count=pd.DataFrame(array\_count)

**TFIDF VECTORIZER**

tfidf\_vectorizer=TfidfVectorizer()

vector=tfidf\_vectorizer.fit\_transform([str.join(' ', x) for x in lemmatized\_sentence])

feature\_name\_tfidf=tfidf\_vectorizer.get\_feature\_names\_out()

array\_tfidf=vector.toarray()

x\_tfidf=pd.DataFrame(array\_tfidf)

**TRAIN TEST SPLIT**

x\_train\_c,x\_test\_c,y\_train\_c,y\_test\_c=train\_test\_split(x\_count,y,test\_size=.2,random\_state=0)

x\_train\_tf,x\_test\_tf,y\_train\_tf,y\_test\_tf=train\_test\_split(x\_tfidf,y,test\_size=.2,random\_state=0)

**MODEL SELECTION**

# USING COUNT VECTORIZER

models={

    'Logistic':{'model':LogisticRegression(),'params':{}},

    'Random Forest':{'model':RandomForestClassifier(),'params':{'n\_estimators':[5,9,41]}},

    'Decision Tree':{'model':DecisionTreeClassifier(),'params':{}},

    'Multinomial':{'model':MultinomialNB(),'params':{}},

    'SVM':{'model':SVC(),'params':{'gamma':[1,2,100],'C':[1,2,5,100],'kernel':['linear','poly','sigmoid','rbf']}}

}

result=[]

for i in models:

  a=models.get(i)

  model=a.get('model')

  params=a.get('params')

  cv=GridSearchCV(model,params,cv=5,return\_train\_score=True)

  cv.fit(x\_count,y)

  bp=cv.best\_params\_

  bs=cv.best\_score\_

  t\_res={'model':i,'best params':bp,'best score':bs,'vectorizer':'CountVectorizer'}

  result.append(t\_res)

# USING TFIDF VECTORIZER

models={

    'Logistic':{'model':LogisticRegression(),'params':{}},

    'Random Forest':{'model':RandomForestClassifier(),'params':{'n\_estimators':[5,9,41]}},

    'Decision Tree':{'model':DecisionTreeClassifier(),'params':{}},

    'Multinomial':{'model':MultinomialNB(),'params':{}},

    'SVM':{'model':SVC(),'params':{'gamma':[1,2,100],'C':[1,2,5,100],'kernel':['linear','poly','sigmoid','rbf']}}

}

for i in models:

  a=models.get(i)

  model=a.get('model')

  params=a.get('params')

  cv=GridSearchCV(model,params,cv=5,return\_train\_score=True)

  cv.fit(x\_tfidf,y)

  bp=cv.best\_params\_

  bs=cv.best\_score\_

  t\_res={'model':i,'best params':bp,'best score':bs,'vectorizer':'TfidfVectorizer'}

  result.append(t\_res)

f=pd.DataFrame(result)

f

**SORTING BEST SCORE**

f.sort\_values('best score').tail(1)

**MODEL FITTING**

model = DecisionTreeClassifier()

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

**PREDITING**

text=["""Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that.

 Instead, he had to give a shout out to his enemies, haters and  the very dishonest fake news media.

 The former reality show star had just one job to do and he couldn t do it. As our Country rapidly grows

 stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest

 Fake News Media, a Happy and Healthy New Year,  President Angry Pants tweeted.  2018 will be a great year for America!

 As our Country rapidly grows stronger and smarter, I want to wish all of my friends, supporters, enemies, haters,

 and even the very dishonest Fake News Media, a Happy and Healthy New Year. 2018 will be a great year for America!  Donald J.

 Trump (@realDonaldTrump) December 31, 2017Trump s tweet went down about as welll as you d expect.What kind of president

  sends a New Year s greeting like this despicable, petty, infantile gibberish? Only Trump! His lack of decency won t even

  llow him to rise above the gutter long enough to wish the American citizens a happy new year!  Bishop Talbert Swan

  (@TalbertSwan) December 31, 2017no one likes you  Calvin (@calvinstowell) December 31, 2017Your impeachment would make

  2018 a great year for America, but I ll also accept regaining control of Congress.  Miranda Yaver (@mirandayaver) December 31,

   2017Do you hear yourself talk? When you have to include that many people that hate you you have to wonder? Why do the they

    all hate me?  Alan Sandoval (@AlanSandoval13) December 31, 2017Who uses the word Haters in a New Years wish??  Marlene

    (@marlene399) December 31, 2017You can t just say happy new year?  Koren pollitt (@Korencarpenter) December 31, 2017Here s

    Trump s New Year s Eve tweet from 2016.Happy New Year to all, including to my many enemies and those who have fought me and

    lost so badly they just don t know what to do. Love!  Donald J. Trump (@realDonaldTrump) December 31, 2016This is nothing

    new for Trump. He s been doing this for years.Trump has directed messages to his  enemies  and  haters  for New Year s,

    Easter, Thanksgiving, and the anniversary of 9/11. pic.twitter.com/4FPAe2KypA  Daniel Dale (@ddale8) December 31, 2017Trump

     s holiday tweets are clearly not presidential.How long did he work at Hallmark before becoming President?  Steven Goodine

      (@SGoodine) December 31, 2017He s always been like this . . . the only difference is that in the last few years,

       his filter has been breaking down.  Roy Schulze (@thbthttt) December 31, 2017Who, apart from a teenager uses the term

       haters?  Wendy (@WendyWhistles) December 31, 2017he s a fucking 5 year old  Who Knows (@rainyday80) December 31, 2017So,

        to all the people who voted for this a hole thinking he would change once he got into power, you were wrong! 70-year-old

         men don t change and now he s a year older.Photo by Andrew Burton/Getty Images."""]

a1=[]

for i in range(len(text)):

  a=str(text[i])

  a=re.sub('[^a-zA-Z]',' ',a)

  pos\_tagged\_text=nltk.pos\_tag(nltk.word\_tokenize(a))

  a1.append(pos\_tagged\_text)

wordnet\_tagged=list(map(lambda x: (x[0],pos\_tagger(x[1])),a1))

lemmatized\_sentence=[]

for word,tag in wordnet\_tagged:

  if word not in set(stopwords.words('english')):

    if tag is None:

      lemmatized\_sentence.append(word)

    else:

      lemmatized\_sentence.append(lemmatize.lemmatize(word,tag))

  else:

    pass

vector=tfidf\_vectorizer.transform([str.join(' ', x) for x in lemmatized\_sentence])

array\_pred=vector.toarray()

x\_pred=pd.DataFrame(array\_pred)

model.predict(x\_pred)

**FLASK CODE**

**Main.py**

from flask import Flask, request, render\_template  
import pandas as pd  
import nltk  
from nltk.corpus import wordnet,stopwords  
from nltk.stem import WordNetLemmatizer  
lemmatize=WordNetLemmatizer()  
import re  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.model\_selection import train\_test\_split  
from sklearn.tree import DecisionTreeClassifier  
tfidf\_vectorizer = TfidfVectorizer()  
  
app = Flask(\_\_name\_\_)  
fake=pd.read\_csv('Fake.csv')  
fake['label']='fake'  
real=pd.read\_csv('True.csv')  
real['label']='real'  
fake1=fake[:2000]  
real1=real[:2000]  
df1=pd.concat([fake1,real1],ignore\_index=True)  
  
x=df1['text']  
y=df1['label']  
  
  
def pos\_tagger(nltk\_tag):  
 nltk\_tag = str(nltk\_tag)  
 if len(nltk\_tag) > 1:  
 if nltk\_tag.startswith('J'):  
 return wordnet.ADJ  
 elif nltk\_tag.startswith('V'):  
 return wordnet.VERB  
 elif nltk\_tag.startswith('N'):  
 return wordnet.NOUN  
 elif nltk\_tag.startswith('R'):  
 return wordnet.ADV  
 return None  
  
def vectorize(x):  
 text = []  
 for i in df1.text:  
 tk = nltk.sent\_tokenize(i)  
 text.append(tk)  
 a1=[]  
 for i in range(len(text)):  
 a=str(text[i])  
 a=re.sub('[^a-zA-Z]',' ',a)  
 pos\_tagged\_text=nltk.pos\_tag(nltk.word\_tokenize(a))  
 a1.append(pos\_tagged\_text)  
  
 wordnet\_tagged = list(map(lambda x: (x[0], pos\_tagger(x[1]) if len(x) > 1 else (x[0], None)), a1))  
  
 lemmatized\_sentence=[]  
 for word,tag in wordnet\_tagged:  
 if word not in set(stopwords.words('english')):  
 if tag is None:  
 lemmatized\_sentence.append(word)  
 else:  
 lemmatized\_sentence.append(lemmatize.lemmatize(word,tag))  
 else:  
 pass  
  
  
  
 vector = tfidf\_vectorizer.fit\_transform([str.join(' ', x) for x in lemmatized\_sentence])  
 feature\_name\_tfidf = tfidf\_vectorizer.get\_feature\_names\_out()  
 array\_tfidf = vector.toarray()  
 x\_tfidf = pd.DataFrame(array\_tfidf)  
 return x\_tfidf  
  
x\_=vectorize(x)  
x\_train\_tf,x\_test\_tf,y\_train\_tf,y\_test\_tf=train\_test\_split(x\_,y,test\_size=.2,random\_state=0)  
model = DecisionTreeClassifier()  
model.fit(x\_train\_tf,y\_train\_tf)  
  
def fake\_new\_detector(text):  
 a1 = []  
 a = str(text) # Assuming text is a single string  
 a = re.sub('[^a-zA-Z]', ' ', a)  
 pos\_tagged\_text = nltk.pos\_tag(nltk.word\_tokenize(a))  
 a1.append(pos\_tagged\_text)  
  
 wordnet\_tagged = list(map(lambda x: (x[0], pos\_tagger(x[1]) if len(x) > 1 else (x[0], None)), a1))  
  
 lemmatized\_sentence = []  
 for word,tag in wordnet\_tagged:  
 if word not in set(stopwords.words('english')):  
 if tag is None:  
 lemmatized\_sentence.append(word)  
 else:  
 lemmatized\_sentence.append(lemmatize.lemmatize(word,tag))  
 else:  
 pass  
  
 if not lemmatized\_sentence:  
 # Handle case where lemmatized\_sentence is empty  
 return ["unknown"]  
  
 vector = tfidf\_vectorizer.transform([str.join(' ', x) for x in lemmatized\_sentence])  
 array\_pred = vector.toarray()  
 x\_pred = pd.DataFrame(array\_pred)  
  
 prediction = model.predict(x\_pred)  
 return prediction  
  
  
  
@app.route('/')  
def home():  
 return render\_template('index.html')  
  
  
@app.route('/predict', methods=['POST'])  
def predict():  
 if request.method == 'POST':  
 text = request.form['message']  
 pred=fake\_new\_detector(text)  
 print(pred)  
 return render\_template('index.html', prediction=pred)  
  
 else:  
 return render\_template('index.html', prediction="something went wrong")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run()

**templates/index.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Fake News Detection</title>  
</head>  
<body>  
 <div class="login">  
 <h1>Fake News Detection</h1>  
 <form action="{{url\_for('predict')}}" method="POST">  
 <textarea name="message" rows="6" cols="50" required="required"></textarea>  
 <br> </br>  
 <button type="submit" class="btn btn-primary btn-block btn-large">predict</button>  
 <div class="results">  
 {% if prediction == ["fake"]%}  
 <h2 style="color:red;">Looking Fake</h2>  
 {% elif prediction == ["real"]%}  
 <h2 style="color:green;">Looking Real</h2>  
 {% endif %}  
 </div>  
 </form>  
 </div>  
</body>  
</html>