



# FAKE NEWS CLASSIFIER PROJECT

Submitted by:  
CHETHANA M

# ACKNOWLEDGMENT

I express my sincere gratitude to **Flip Robo Technologies** for giving me this opportunity to carry out the project work.

A special thanks to my mentor **Mohd Kashif** for guiding me in completing this project and being available to resolve my doubts whenever I raise any tickets.

I also would love to take this moment to thank **DataTrained** for giving me all the knowledge about how to build an effective machine learning model and providing this opportunity to work as intern in Flip Robo Technologies.

## INTRODUCTION



Fake news has become one of the biggest problems of our age. It has serious impact on our online as well as offline discourse. One can even go as far as saying that, to date, fake news poses a clear and present danger to western democracy and stability of the society.

What is a Fake News?

Fake news's simple meaning is to incorporate information that leads people to the wrong path. Nowadays fake news spreading like water and people share this information without verifying it. This is often done to further or impose certain ideas and is often achieved with political agendas.

Fake news on different platforms is spreading widely and is a matter of serious concern, as it causes social wars and permanent breakage of the bonds established among people. A lot of research is already going on focused on the classification of fake news.

Here we will try to solve this issue with the help of machine learning in Python.

### **Technical goals :**

As time flows, the amount of data, especially text data increases exponentially. Along with the data, our understanding of AI also increases and the computing power enables us to train very complex and large models faster. Fake news has been gathering a lot of attention worldwide recently. The effects can be political, economic, organizational, or even personal. This project discusses the approach of natural language processing and machine learning in order to solve this problem. Use of bag-of-words, count vectorizer has been made, TF-IDF, and trained the data on five classifiers to investigate which of them works well for this specific dataset of labelled news statements. The precision, recall and f1 scores help us determine which model works best.

### **Dataset :**

You can find many datasets for fake news detection on Kaggle or many other sites. I download these datasets from Kaggle. There are two datasets one for fake news and one for true news. In true news, there is 21417 news, and in fake news, there is 23481 news. You have to insert one label column zero for fake news and one for true news. We are combined both datasets using pandas built-in function.

Once we concat both True news data and Fake news data and add the labels, the shuffle the dataset it looks like below :

In [7]: df

Out[7]:

		title	text	subject	date	label
0	A New Low: Trump Exploits Las Vegas Shooting ...	Donald Trump doesn't exactly hide the fact tha...	News	October 8, 2017	0	
1	DODGY DOSSIER: The Trump-Russia Dossier Was Fu...	21st Century Wire says This week an unverified...	US_News	January 13, 2017	0	
2	FLINT RESIDENTS TOLD TO PAY BILLS FOR POISON W...	You seriously can't make this up. This Democra...	politics	Jan 25, 2016	0	
3	LOL! DEMOCRATS Express Concerns Over Possible ...	A progressive group charged Saturday that the ...	politics	Feb 26, 2017	0	
4	Vehemently Anti-Gay Pastor Arrested On 70 Cou...	On May 9, Pastor David Reynolds, formerly of C...	News	May 15, 2016	0	
...	...	...	...	...	...	
44893	COMEDY GENIUS: [Video] "Bob Ross" Paints Sick ...	Steven Crowder knocks it out of the park with ...	politics	Sep 17, 2016	0	
44894	Medicaid, pension costs create budget complica...	NEW YORK (Reuters) - A sluggish forecast for U...	politicsNews	July 24, 2017	1	
44895	Warren to keep up assault on White House hopef...	WASHINGTON (Reuters) - Democratic U.S. Senator...	politicsNews	June 9, 2016	1	
44896	FLASHBACK [VIDEO]: Libertarian Gary Johnson Di...	Libertarian Presidential candidate Gary Johnso...	left-news	Sep 8, 2016	0	
44897	Senate starts debate on broad energy bill	WASHINGTON (Reuters) - The U.S. Senate on Wedn...	politicsNews	January 27, 2016	1	

44898 rows x 5 columns

The final dataset has 44898 rows, 5 columns which includes news title, description, subject, published date and label.

### Process included :

1. Calling the required libraries
2. Import necessary files
3. Adding labels, concating the datasets and shuffling
4. Separating input and target variables
5. Applying Countvectorizer
6. Splitting data for training and testing
7. Building the model
8. Cross validation
9. Parameter tuning
10. Saving the final model

## Importing the required libraries

```
In [1]: import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, HashingVectorizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn import metrics
import itertools
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

Firstly we will remove all the stopwords, punctuations and any irrelevant spaces from the text. For that NLTK Library is required and some of its module need to be downloaded. So, for that run the above code.

## Import necessary files and Adding labels, concating the datasets and shuffling

```
In [2]: fakedata=pd.read_csv('Fakenews_Fake_data.csv')
truedata=pd.read_csv('Fakenews_True_data.csv')

In [3]: df1=pd.DataFrame(fakedata)
df2=pd.DataFrame(truedata)

In [4]: #adding labels to the datasets
df1['label'] = 0
df2['label'] = 1

In [5]: #concating both the dataframes
df=pd.concat([df1, df2])
df
```

The data was available in 2 datasets, one for fake news and one for true news. In true news, there are 21417 news, and in fake news, there are 23481 news. We have to insert one label column, zero for fake news and one for true news. Then we have combined both datasets using pandas built-in function.

## Stemmatize news titles

```
In [10]: #stemmatize news titles

ps = PorterStemmer()
corpus = []
for i in range(0, len(messages)):
    review = re.sub('[^a-zA-Z]', ' ', messages['title'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

Text normalization is the process of simplifying multiple variations or tenses of the same word. Stemming and lemmatization are two methods of text normalization, the former being the simpler of the two. To stem a word, we simply remove the suffix of a word to reduce it to its root.

As an example, “jumping”, “jumps”, and “jumped” all are stemmed to “jump.”

Stemming is not without its faults, however. We can run into the issue of overstemming. Overstemming is when words with different meanings are stemmed to the same root - a false positive.

**vectorize all preprocessed titles in corpus, Applying Countvectorizer and Creating the Bag of Words model.**

```
In [12]: #vectorize all preprocessed titles in corpus
#Applying Countvectorizer
# Creating the Bag of Words model
cv = CountVectorizer(max_features=5000,ngram_range=(1,3))
x = cv.fit_transform(corpus).toarray()
```

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

We can use it as follows:

- Create an instance of the CountVectorizer class.
- Call the fit() function in order to learn a vocabulary from one or more documents.
- Call the transform() function on one or more documents as needed to encode each as a vector.

An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

In our project we are using count vectorizer to vectorize all the titles in corpus by applying CountVectorizer then create a bag of words model.

## Algorithm & Techniques :

The algorithms that are used in the classifier model are Logistic Regression, Decision Tree Classifier, Random forest classifier, KNeighbors Classifier and Multinomial Naive Bayes.

## Splitting train-test data :

```
In [14]: # Divide the dataset into Train and Test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

To split the data into our two datasets, we'll use scikit-learn's `train_test_split()` method.

Let's say we have 100 records in the loaded dataset. If we specify the test dataset is 25 percent, we'll split 75 records for training and use the remaining 25 records for testing.

## Building the models :

```
In [18]: mn = MultinomialNB()
lg = LogisticRegression()
dtc = DecisionTreeClassifier()
knc = KNeighborsClassifier()
rfc = RandomForestClassifier()

In [19]: model=[mn,lg,dtc,knc,rfc]
for m in model:
    m.fit(x_train,y_train)
    m.score(x_train,y_train)
    pred=m.predict(x_test)
    print("Accuracy score of ",m,"is : ")
    print(accuracy_score(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print(classification_report(y_test,pred))
    print('\n')
```

Firstly we will be calling all the algorithms required and the start building the models for the same train and test datasets. We are here using `MultinomialNB`, `LogisticRegression`, `DecisionTreeClassifier`, `KNeighborsClassifier` and `RandomForestClassifier`.

The reason behind why we are building 5 models simultaneously for the same train and test data is that to check for the performance of the models and conclude the final model which gives the best score. The same is don by running the code as shown above.

By running the above codes we were able see the performance of each model and by comparing the score and matrix we can tell that `LogisticRegression` is giving the best result whose accuracy score and matrix is as below.

```
Accuracy score of  LogisticRegression() is :
0.9496659242761692
[[5556  338]
 [ 227 5104]]
      precision    recall  f1-score   support

     0       0.96       0.94       0.95       5894
     1       0.94       0.96       0.95       5331

 accuracy                   0.95       11225
 macro avg       0.95       0.95       0.95       11225
 weighted avg    0.95       0.95       0.95       11225
```

Going further we will be performing cross validation on the algorithm which are giving us the better scores comparatively. This can be performed by running the codes as shown below.

```
In [20]: from sklearn.model_selection import cross_val_score

#Cross validation

m1=[mn,lg,rfc]
for m in m1:
    score=cross_val_score(m,x_train,y_train,cv=5)
    print("Score of : ",m )
    print(score)
    print(score.mean()
          )
    print(score.std())
    print('\n')
```

While performing cross validation we could observe that mean score was best for the Logistic Regression model again.

## Parameter tuning

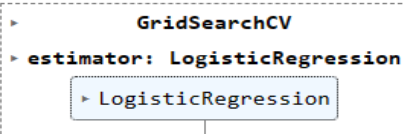
As we can observe that Logistic regression is the best model so far, to check if we can further improve the performance of the model by using parameter tuning method.

```
In [22]: #Parameter tuning for LogisticRegression

from sklearn.model_selection import GridSearchCV

params1={'penalty':['l1','l2','elasticnet']}
grd1=GridSearchCV(estimator=lg,param_grid=params1,cv=5)
grd1.fit(x_train,y_train)
```

```
Out[22]:
```



```
In [23]: grd1.best_score_
```

```
Out[23]: 0.9461883131816317
```

```
In [24]: grd1.best_params_
```

```
Out[24]: {'penalty': 'l2'}
```

So now we can observe that Logistic regression model is working better with the parameter where penalty is 'l2' by giving us the accuracy score of 94.61%.

We can now finalize LogisticRegression with penalty as 'l2' model as our final classifier.



```
In [27]: # Finalizing the best model

final_classifier=LogisticRegression(penalty='l2')
final_classifier.fit(x_train,y_train)
pred=final_classifier.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

0.9496659242761692
[[5556  338]
 [ 227 5104]]
      precision    recall  f1-score   support

      0       0.96      0.94      0.95      5894
      1       0.94      0.96      0.95      5331

   accuracy          0.95
  macro avg       0.95      0.95      0.95
 weighted avg     0.95      0.95      0.95
```

## Saving the final model and checking for the predicted values

```
In [28]: # We can save the final model now

import joblib
joblib.dump(final_classifier,'Fake_news_classifier.obj')
```

Out[28]: ['Fake\_news\_classifier.obj']

Loading the saved model and predicting the values

```
In [30]: Fake_news=joblib.load('Fake_news_classifier.obj')
```

```
In [31]: pred=Fake_news.predict(x_test)
print("Predicted values :",pred)
```

Predicted values : [0 1 1 ... 1 0 0]

Now we can create a dataframe with the predicted values and original values and do the comparisons between both values.

```
In [33]: # Making a DataFrame of Predicted values and Original values

df_predicted=pd.DataFrame({'Predicted values':pred,'Original values':y_test})
df_predicted
```

Out[33]:

	Predicted values	Original values
22216	0	0
27917	1	1
25007	1	1
1377	0	0
32476	1	0
...	...	...
15578	1	1
29394	0	0
3120	1	1
25388	0	0
14337	0	0

11225 rows × 2 columns

## CONCLUSION

To summarize this project, the first step involved studying the dataset.

The second major step was to separate input and target variables, stemmatizing the titles, applying Countvectorizer, Splitting data for training and testing.

The third step was to build models MultinomialNB, LogisticRegression, DecisionTreeClassifier, KneighborsClassifier and RandomForestClassifier and check for the best performing model comparing their accuracy scores, precision, recall and f1 scores.

Finally parameter tuning is done on **LogisticRegression** model as it was performing better compared to other 4 models and chose the best model with best parameter where **penalty = l2**, which was giving the best score of **94.96%**.

Although we have built a model which is giving us accuracy of 94.9%, it is always good to build maximum number of models for the same test and train data to check if other model perform better than the current one.