

# Machine Learning Aided Differentiation of Real and Fake News

[BEGINNER](#)[MACHINE LEARNING](#)[NLP](#)[PYTHON](#)

This article was published as a part of the [Data Science Blogathon](#).

## Introduction

Since the dawn of this millennium, technology has seen rapid advancement. This has led to the introduction of many news channels across different media viz. electronic including online and television, and print media. An increase in the number of platforms and channels has also paved the way for ever-increasing competition. Sensationalization has become a new way of attracting the attention of the audience, particularly for electronic media which at times is driven by [fake news](#). Machine learning holds a promise in differentiating between real news and fake news. In this article, we shall be making use of a dataset to understand the technicalities of machine learning in detecting fake news.

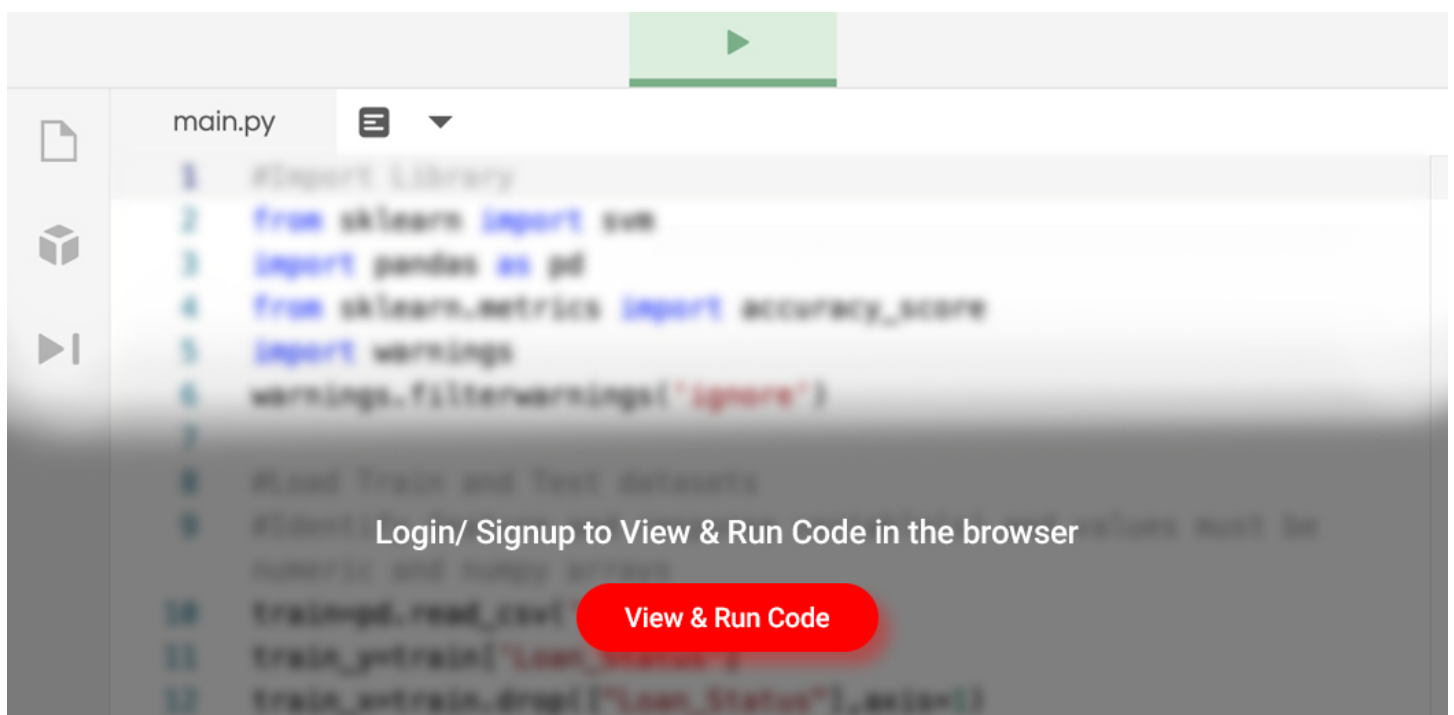
## Dataset

The dataset of fake and real news that has been used in this article can be downloaded here–

<https://www.kaggle.com/clmentbisailon/fake-and-real-news-dataset>

## Importing the Dependencies

At the outset, we shall be importing the dependencies with the help of the following lines of code–



```
1 #Import Library
2 from sklearn import svm
3 import pandas as pd
4 from sklearn.metrics import accuracy_score
5 import warnings
6 warnings.filterwarnings("ignore")
7
8 #Load Train and Test datasets
9 #Load
10
11 #Load Train and Test datasets
12 train=pd.read_csv(
13 train_path["train_data.csv"],
14 train_path["train_data.csv"],
15 train_path["train_data.csv"],
16 train_path["train_data.csv"],
17 train_path["train_data.csv"],
18 train_path["train_data.csv"],
19 train_path["train_data.csv"],
20 train_path["train_data.csv"],
21 train_path["train_data.csv"],
22 train_path["train_data.csv"],
23 train_path["train_data.csv"],
24 train_path["train_data.csv"],
25 train_path["train_data.csv"],
26 train_path["train_data.csv"],
27 train_path["train_data.csv"],
28 train_path["train_data.csv"],
29 train_path["train_data.csv"],
30 train_path["train_data.csv"],
31 train_path["train_data.csv"],
32 train_path["train_data.csv"],
33 train_path["train_data.csv"],
34 train_path["train_data.csv"],
35 train_path["train_data.csv"],
36 train_path["train_data.csv"],
37 train_path["train_data.csv"],
38 train_path["train_data.csv"],
39 train_path["train_data.csv"],
40 train_path["train_data.csv"],
41 train_path["train_data.csv"],
42 train_path["train_data.csv"],
43 train_path["train_data.csv"],
44 train_path["train_data.csv"],
45 train_path["train_data.csv"],
46 train_path["train_data.csv"],
47 train_path["train_data.csv"],
48 train_path["train_data.csv"],
49 train_path["train_data.csv"],
50 train_path["train_data.csv"],
51 train_path["train_data.csv"],
52 train_path["train_data.csv"],
53 train_path["train_data.csv"],
54 train_path["train_data.csv"],
55 train_path["train_data.csv"],
56 train_path["train_data.csv"],
57 train_path["train_data.csv"],
58 train_path["train_data.csv"],
59 train_path["train_data.csv"],
60 train_path["train_data.csv"],
61 train_path["train_data.csv"],
62 train_path["train_data.csv"],
63 train_path["train_data.csv"],
64 train_path["train_data.csv"],
65 train_path["train_data.csv"],
66 train_path["train_data.csv"],
67 train_path["train_data.csv"],
68 train_path["train_data.csv"],
69 train_path["train_data.csv"],
70 train_path["train_data.csv"],
71 train_path["train_data.csv"],
72 train_path["train_data.csv"],
73 train_path["train_data.csv"],
74 train_path["train_data.csv"],
75 train_path["train_data.csv"],
76 train_path["train_data.csv"],
77 train_path["train_data.csv"],
78 train_path["train_data.csv"],
79 train_path["train_data.csv"],
80 train_path["train_data.csv"],
81 train_path["train_data.csv"],
82 train_path["train_data.csv"],
83 train_path["train_data.csv"],
84 train_path["train_data.csv"],
85 train_path["train_data.csv"],
86 train_path["train_data.csv"],
87 train_path["train_data.csv"],
88 train_path["train_data.csv"],
89 train_path["train_data.csv"],
90 train_path["train_data.csv"],
91 train_path["train_data.csv"],
92 train_path["train_data.csv"],
93 train_path["train_data.csv"],
94 train_path["train_data.csv"],
95 train_path["train_data.csv"],
96 train_path["train_data.csv"],
97 train_path["train_data.csv"],
98 train_path["train_data.csv"],
99 train_path["train_data.csv"],
100 train_path["train_data.csv"]
```

We start by importing `numpy`, `pandas`, and `re`. 're' is an in-built package that stands for the regular expression. A search pattern is formed with the help of a sequence of characters. Then, we shall be importing stopwords. Stopwords are words that are not very significant like a, an, etc. We import stopwords from `nltk.corpus` where 'nltk' is the *natural language tool kit* and 'corpus' is a *repository of stopwords*.

*Lemmatization* is done to convert the word to its base form. There is also a process called stemming which also converts the word to its base form but lemmatization is more contextual compared to stemming. For this, we shall import `WordNetLemmatizer` from `nltk.stem.wordnet`. Lemmatization is done by using wordnets inherent function known as `morph`. `nltk.stem` is the library for python Lemmatization. Next, we shall import string to use any constant and classes.

Now, we import `TfidfVectorizer` from `sklearn.feature_extraction.text`. `TfidfVectorizer` is Term Frequency Inverse Document Frequency that transforms text into a meaningful collection of numbers. The numbers are used to fit the machine algorithm for prediction. This makes use of the package `sklearn.feature_extraction` which extracts features in a format supported by machine learning. As this is a binary classification problem, we shall be using *logistic regression* to classify real and fake news. Next, we shall import `nltk` and download *stopwords*.

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\91863\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!

Out[7]: True
```

Then, we shall print the stopwords in english with the help of print function.

```
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "yo
u've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'h
is', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
lf', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'who
m', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were',
'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing',
'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of',
'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'durin
g', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'o
n', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'wh
en', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'ot
her', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'to
o', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've",
'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "cou
ldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'h
aven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'n
eedn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'were
n', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Next, we shall read the files by creating a dataframe with the help of the following lines of code

```
true_data = pd.read_csv('True.csv')
fake_data = pd.read_csv('Fake.csv')
```

Next, we shall be creating target columns for both true and fake data

```
true_data['Target'] = 0
fake_data['Target'] = 1
```

Then, we concatenate both the true and the fake data into a common dataframe 'data'

```
data = pd.concat([true_data,fake_data]).sample(frac=1, random_state = 1).reset_index(drop=True) data
```

Out[11]:

		title	text	subject	date	Target
0		EPA chief says Paris climate agreement 'bad de...	WASHINGTON (Reuters) - The United States shoul...	politicsNews	April 2, 2017	0
1		BREAKING NEWS: President Trump Announces Major...	President Trump just tweeted out a new policy ...	politics	Jul 26, 2017	1
2		Trump says New Hampshire win not necessary to ...	WASHINGTON (Reuters) - U.S. Republican preside...	politicsNews	February 7, 2016	0
3		Kremlin: U.S. sanctions aimed at turning busin...	MOSCOW (Reuters) - The Kremlin said on Thursda...	worldnews	November 30, 2017	0
4		MUST WATCH: Kellyanne Conway PUNCHES BACK Afte...	Kellyanne Conway s response to Williams criti...	left-news	Dec 27, 2016	1
...		...	...	...	...	...
44893		THIS YEAR: Let's Make Christmas Great Again...	This year, let s try something a little differ...	US_News	December 25, 2016	1
44894		DEMOCRATS SELL Promo T-Shirt: "Democrats give ...	Yes, the Democrats think it s a good thing to ...	politics	Apr 20, 2017	1
44895		White House aides told to ...	WASHINGTON (Reuters) - The White House counsel	politicsNews	March 2, 2017	0

## Data Pre-processing

Before developing the data model, we shall be pre-processing the data. With the help of the *shape* function, we will be able to identify a number of rows and columns.

```
data.shape
```

There are 44898 rows and 5 columns. Then, we shall be printing the first 5 rows of the dataframe by using the *head()* function.

```
data.head()
```

Then, missing values of the dataset are found by using *isnull()* function.

The above output indicates that there are no missing or null values in the dataset. Next, we shall merge columns into a common pandas dataframe 'content' with the help of the following lines of code

```
data['content'] = data['title'] + ' ' + data['text']
```

Next, we shall use the print function to generate the output of content.

```
print(data['content'])
```

Now, we shall be separating the data and target.

```
X = data.drop(columns='Target', axis=1) Y = data['Target']
```

Then, we shall print the data and target which are stored in variables X and Y respectively.

```
print(X) print(Y)
```

We shall be doing lemmatization to convert the word to its base form.

```
def clean(doc): stop = stopwords.words('english') punct = string.punctuation wn1 = WordNetLemmatizer()
stopwords_free = " ".join([i for i in doc.lower().split() if i not in stop]) punctuations_free = "".join(ch
for ch in stopwords_free if ch not in punct) normalized = " ".join(wn1.lemmatize(word) for word in
punctuations_free.split()) return normalized
```

Then, we shall print the content dataframe.

We shall be doing lemmatization to convert the word to its base form.

```
def clean(doc): stop = stopwords.words('english') punct = string.punctuation wnl = WordNetLemmatizer()
stopwords_free = " ".join([i for i in doc.lower().split() if i not in stop]) punctuations_free = "".join(ch
for ch in stopwords_free if ch not in punct) normalized = " ".join(wnl.lemmatize(word) for word in
punctuations_free.split()) return normalized
```

Then, we shall print the content dataframe.

```
print(data['content'])
```

Next, we shall be using .values method to view the values of the dictionary as a list. We shall also be separating the data and the target stored in X and Y respectively. Also, we would print both data and target.

```
X = data['content'].values Y = data['Target'].values
```

```
print(X)
```

```
print(Y)
```

```
[0 1 0 ... 0 0 1]
```

Now, we shall be converting the textual data to numerical data. This shall be done using *TfidfVectorizer* which shall be imported from *sklearn.feature\_extraction.text*. *TfidfVectorizer* is Term Frequency Inverse Document Frequency that transforms text into a meaningful collection of numbers.

```
vectorizer = TfidfVectorizer() vectorizer.fit(X) X = vectorizer.transform(X)
```

We have fit X for transformation. Now, we shall be printing X to see the transformation.

```
print(X)
```

```
(0, 138133) 0.102554837755071
(0, 137512) 0.04772005423601248
(0, 137401) 0.026182155381637828
(0, 136877) 0.0285569324171666
(0, 136835) 0.0471422611627891
(0, 136021) 0.04598393034275061
(0, 135935) 0.01681910178733555
(0, 135930) 0.05605123178012753
(0, 135858) 0.018125179970683067
(0, 135333) 0.07224411306410877
(0, 135133) 0.03202960492626489
(0, 134773) 0.02650309229795233
(0, 132619) 0.024516811040229826
(0, 130517) 0.08817556702451246
(0, 129921) 0.04797926044745436
(0, 127655) 0.03201869259696583
(0, 125912) 0.01982276464410226
(0, 125760) 0.14855137125976772
(0, 125371) 0.029980522534522085
(0, 124693) 0.04079837400670336
(0, 124283) 0.03338315209459717
(0, 124148) 0.24296728402952922
(0, 124110) 0.06555492573298116
```

So, from the above output, we can draw the inference that 'X' has successfully transformed. This will be followed by splitting of the dataset into training and testing data.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify= Y, random_state=5)
```

In the above line of code, the test data would be 20% of the dataset.

## Model Development

The first step of model development is to train the model. Here, we shall be using a logistic regression classification algorithm to differentiate between fake and real news. This will be done by fitting the data.

```
model = LogisticRegression() model.fit(X_train, Y_train)
```

```
Out[29]: LogisticRegression()
```

The model has been developed, now we shall be evaluating the model.

## Evaluation of Model

The model is evaluated with the help of an accuracy score and confusion matrix. First of all, we would obtain the accuracy score of the training dataset by using the following lines of code

```
X_train_prediction= model.predict(X_train) training_data_accuracy = accuracy_score(X_train_prediction,
Y_train)
```

Then, we shall find the accuracy score of the training dataset

```
print('Accuracy score of the training data : ', training_data_accuracy)
```

Accuracy score of the training data : 0.9919817361768473

From the output generated above, the accuracy of the training dataset is 99.19% indicating high accuracy of almost a perfect 100%.

Next, we shall find the accuracy of the testing dataset on a code very similar to the one we did for the training dataset

```
X_test_prediction = model.predict(X_test) test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

Then, we shall find the accuracy score of the testing dataset

```
print('Accuracy score of the test data : ', test_data_accuracy)
```

Accuracy score of the test data : 0.9899777282850779

From the output generated above, the accuracy of the testing dataset is 98.99%, almost 99% indicating high accuracy.

The last step would be to generate a confusion matrix which would identify the percentage of the dataset that has been classified as true positive, true negative, false positive, and false negative. The confusion matrix is depicted below

Predicted: Yes	Predicted: No	Total	
Actual: Yes	a (T.P)	b (F.N)	a+b (Actual yes)
Actual: No	c (F.P)	d (T.N)	c+d (Actual No)
Total	a+c (Predicted yes)	b+d (Predicted No)	a+b+c+d

Here, T.P = True Positive, F.N = False Negative, F.P = False Positive, T.N = True Negative

```
print(confusion_matrix(X_test_prediction, Y_test))
```

```
[[4243  49]
 [ 41 4647]]
```

In this dataset, the model could classify 4243 as T.P and 4647 as F.P indicating a small percentage of misclassification.

## Conclusion to Machine Learning

The key takeaways of this article are-

1. We comprehended how to import dependencies particularly with respect to natural language processing.
2. We developed an understanding of the application of lemmatization and vectorization for data pre-processing.
3. Usefulness of logistic regression as a potent tool in binary classification as it could accurately distinguish between fake and real news to the tune of 99%.
4. A very negligible misclassification percentage as only 49 got classified as false negative and 41 as false positive.



5. There is another classification tool PassiveAggressiveClassifier which is an online classification tool very useful for data flowing from Twitter, etc. every second which we shall be discussing very soon.

Thank you for taking the time to go through this article on detecting fake and real news using machine learning. Stay safe!

**The media shown in this article is not owned by Analytics Vidhya and is used at the Author's discretion.**

---

Article Url - <https://www.analyticsvidhya.com/blog/2022/07/machine-learning-aided-differentiation-of-real-and-fake-news/>



**Saptarshi Dutta**