

Fake News Analysis In Social Media Using IBM Watson

A MINI PROJECT REPORT Submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,
HYDERABAD**

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

JANGA SAHITH REDDY (18UK1A05D8)

KASULA VISHWASRI (18UK1A05E3)

CHIRRA SAI VYSHNAVI (18UK1A05C8)

KASULA ANILKUMAR (18UK1A0580)

Under the esteemed guidance of

Ms.B. AMBIKA

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTU, HYDERABAD
BOLLIKUNTA, WARANGAL-506005.
2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTU, HYDERABAD

BOLLIKUNTA, WARANGAL-506005.

2018-2022



CERTIFICATE

This is to certify that the Mini Project Report entitled “Fake News Analysis In Social Media Using IBM Watson” is being submitted by **JANGA SAHITH REDDY (18UK1A05D8) KASULA VISHWASRI (18UK1A05E3) CHIRRA SAI VYSHNAVI(18UK1A05C8) KASULA ANILKUMAR (18UK1A0580)** in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology in computer science and engineering during the academic year 2018-2022.

Ms.B. AMBIKA

project guide

Dr. R. NAVEEN KUMAR

head of the department

External Examiner

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this mini project in the institute.

We extend our heartfelt thanks to **Dr. R. Naveen Kumar**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the mini project.

We express heartfelt thanks to the Mini Project Coordinator, **Ms. G. Aruna kranthi**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion of this mini project.

We express heartfelt thanks to the guide, **Ms.B.AMBIKA**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion of this mini project.

We express our sincere thanks and gratitude to The Smart Bridge, for providing internship.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

CONTENTS

LIST OF CHAPTERS	PAGE NO
2 ABSTRACTION	5
3 INTRODUCTION	6
4 OBJECTIVE OF THE PROJECT	7
5 PROBLEM DEFINITIONS	8
6 MOTIVATION OF THE PROJECT	8
7 LITERATURE SURVEY	9
8 DESIGN OF THE PROJECT	10
9 IMPLEMENTATION AND TESTING	11-27
10 RESULTS	28
11 CONCLUSION	29
12 BIBILOGRAPHY	30-3

ABSTRACT

Social media for news consumption is a double-edged sword. On the one hand, its low cost, easy access, and rapid dissemination of information lead people to seek out and consume news from social media. On the other hand, it enables the wide spread of "fake news", i.e., low quality news with intentionally false information. The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention. Fake news detection on social media presents unique characteristics and challenges that make existing detection algorithms from traditional news media ineffective or not applicable. First, fake news is intentionally written to mislead readers to believe false information, which makes it difficult and nontrivial to detect based on news content; therefore, we need to include auxiliary information, such as user social engagements on social media, to help make a determination. Second, exploiting this auxiliary information is challenging in and of itself as users' social engagements with fake news produce data that is big, incomplete, unstructured, and noisy. Because the issue of fake news detection on social media is both challenging and relevant, we conducted this survey to further facilitate research on the problem. In this survey, we present a comprehensive review of detecting fake news on social media, including fake news characterizations on psychology and social theories, existing algorithms from a data mining perspective, evaluation metrics and representative datasets. We also discuss related research areas, open problems, and future research directions for fake news detection on social media

1 INTRODUCTION

Nowadays, fake news has become a common trend. Even trusted media houses are known to spread fake news and are losing their credibility. So, how can we trust any news to be real or fake? There should be a system which can analyse whether a given news post is fake or not . so the main of this project is to build an application that can analyse fake news.

In this project, we have built a classifier model that can identify news as real or fake. For this purpose, we have used data from Kaggle, but you can use any data to build this model following the same methods.

With the help of this project you can create an NLP classifier to detect whether the news is real or fake.

There is a paradigm shift in how people consume news today. They mostly look for a summarized version of news over the social media platforms to quickly gather more information . This change is due to easy access to news readily available over social media platforms like Twitter and Facebook. The study's findings concluded that social media, mostly microblogging platforms, were the most significant contributor to spread false information.

The extensive spread of fake news has the potential for extremely negative impacts on individuals and society. Therefore, fake news detection on social media has recently become an emerging research that is attracting tremendous attention. Fake news detection on social media presents unique characteristics and challenges that make existing detection algorithms from traditional news media.

1.1 OBJECTIVES OF THE PROJECT

By the end of this project

- Knowledge on Machine Learning Algorithms.
- Knowledge on Python Language with Machine Learning
- Knowledge on Statistics and Graphs and their relations
- Knowledge on Natural Language Processing (NLP).
- Real Time Analysis of Project
- Building an ease of User Interface (UI)
- Navigation of ideas towards other projects(creativity)
- Knowledge on building ML Model..
- You will be able to know how to find the accuracy of the model.
- How to Build web applications using the Flask framework.

2. PROBLEM DEFINITION

Fake news has existed for a very long time, nearly the same amount of time as news began to circulate widely after the printing press was invented in 1439. However, there is no agreed definition of the term “fake news”. Therefore, we first discuss and compare some widely used definitions of fake news in the existing literature, and provide our definition of fake news that will be used for the remainder of this survey .A narrow definition of fake news is news articles that are intentionally and verifiably false and could mislead readers .There are two key features of this definition: authenticity and intent. First, fake news includes false information that can be verified as such. Second, fake news is created with dishonest intention to mislead consumers. This definition has been widely adopted in recent studies .Broader definitions of fake news focus on the either authenticity or intent of the news content. Some papers regard satire news as fake news since the contents are false even though satire is often entertainment-oriented and reveals its own deceptiveness to the consumers . Other literature directly treats deceptive news as fake news, which includes serious fabrications, hoaxes, and satires.

2.1 MOTIVATION

The project aims to predict fake news analysis in social media using IBM Watson.

In this project we predict whether the news is real or fake. we present the details of mathematical formulation of fake news detection on social media.

3. LITERATURE SURVEY

We discuss how to assess the performance of algorithms for fake news detection

We focus on availability datasets and evaluation for this task.

Online news can be collected from different sources such as news agency home pages , search engines and social media websites. However manually determining veracity of news is a challenging news usually requiring annotators with domain expertise who performs careful analysis of claims and additional evidence, context, and reports from authoritative sources. Generally, news data with annotations can be gathered in the following ways: Expert journalists Fact-checking websites, Industry detectors, and Crowd-sourced workers. However, there are no agreed upon bench-mark datasets for the fake news detection problem. Some publicly available datasets are listed below:

- Buzz Feed News: This dataset comprises a complete sample of news published in Facebook from news agencies over a week close to the 2016 U.S. election from September 19 to 23 and September 26 and 27. Every post and the linked article were fact checked claim by claim by 5 Buzz Feed journalists. This dataset is further enriched in by adding the linked articles, attached media, and relevant metadata. It contains, 627 articles—826 mainstream, 356 left-wing, and 545 right-wing articles.

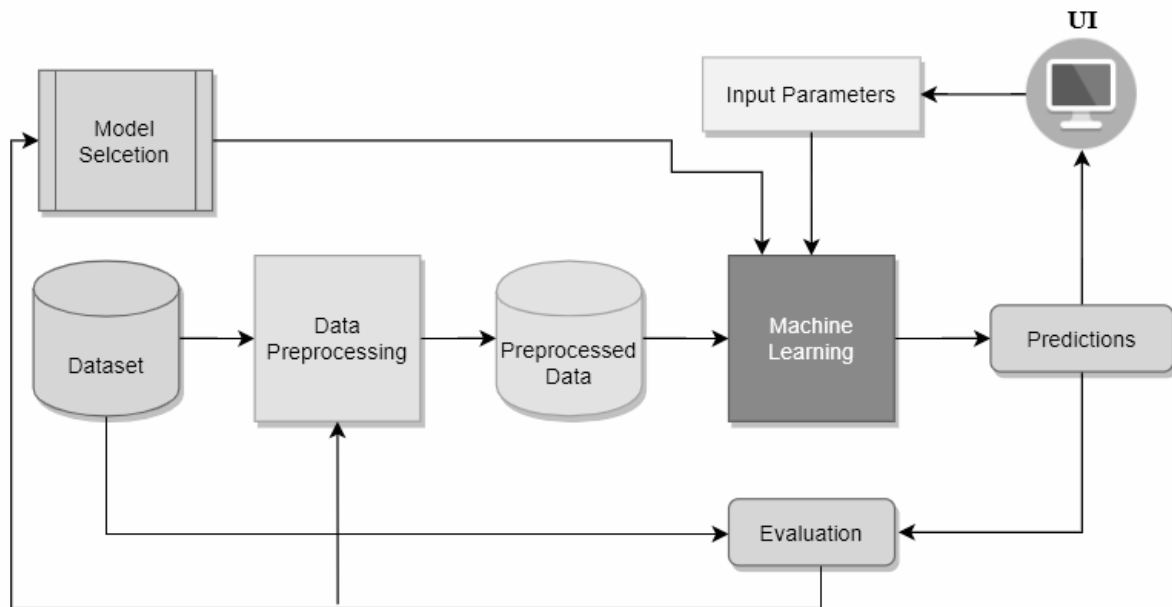
- LIAR: This dataset is collected from fact-checking website Politi Fact through its API .It includes 12,836 human-labeled short statements, which are sampled from various contexts, such as news releases, TV/radio interviews, campaign speeches, etc. The labels for news truthfulness are fine-grained multiple classes: pants-fire, false, barely-true, half-true, mostly true, and true.

- BS Detector : This dataset is collected from a browser extension called BS detector developed for checking news veracity. It searches all links on a given web-page for references to unreliable sources by checking against a manually compiled list of domains. The labels are the outputs of BS detector, rather than human annotators.

CREDBANK: CREDBANK is a large-scale crowd-sourced dataset of approximately 60 million tweets that cover 96 days starting from October 2015. All the tweets are broken down to be related to over 1,000 news event.

DESIGN OF THE PROJECT

ARCHITECTURE:



5.1 INTRODUCTION & INSTALLATIONS:

The first step is usually importing the libraries that will be needed for model building.

Import the libraries given in the below image

```
#Importing Libraries

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import pickle
```

Fig :Importing all the necessary libraries

Numpy- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines. Pandas objects are very much dependent on NumPy objects.

Pandas- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Seaborn- Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

Count-Vectorizer- Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

TF-IDF Vectorizer - TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. Next step is to load the dataset.

Project Flow

- User interacts with the UI (User Interface) to give the review as input
- Given Review is analyzed by the model which is integrated to UI build
- Once model analyses the review the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

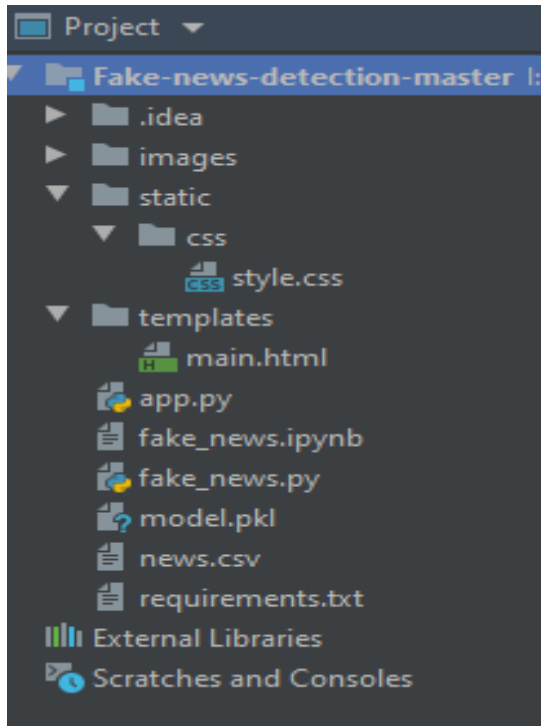
- Install required packages and libraries.
 - Install packages and libraries.
 - Install Anaconda software.
 - Run Jupyter
- Understanding the data.
 - Download the dataset.
 - Importing the required libraries.
 - Loading the dataset.
 - Countvectorizer for text classification.
 - TF-IDF Vectorizer for text classification.
 - Inspecting the vectors.
- Model Building
 - Splitting the data into train and test.
 - Training and testing the model with Countvectorizer & Predicting the result.
 - Training and testing the model with TF-IDF Vectorizer & Predicting the result.
 - Improving the model.
 - Inspecting the model.
 - Saving the model
- Application Building
 - Flask Structure
 - Importing libraries
 - Load Flask and Assign the model.
 - Routing to the Html page.
 - Run the app in local browser.
- Final UI
 - Input the URL & get the result.

Project Structure

You can download the files from this section

Or

Create a Project folder which contains files as shown below



- We are building a Flask Application which needs HTML pages “main.html” stored in the templates folder and a python script app.py for server side scripting
- The model is built in the notebook fake_news.ipynb
- We need the model which is saved and the saved model in this content is model.pkl.
- The static folder will contain a css file which is used in the html file.
- The templates mainly used here are “main.html” for showcasing the UI
- The flask app is denoted as app.py

Now that you have all the required packages and necessary files lets start building the project

Loading The Dataset

Here, we are reading the dataset(.csv) from the system using pandas and storing it in a variable 'df'. It's time to begin building your text classifier! The data has been loaded into a DataFrame called df. The .head() method is particularly informative.

```
df = pd.read_csv('news.csv')
#print the first several row of data
df.head()
```

	Unnamed: 0		title	text	label
0	8476		You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...		Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy		U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...		FAKE
4	875	The Battle of New York: Why This Primary Matters		It's primary day in New York and front-runners...	REAL

Fig: Required Data set .

You can see your data set has three columns: title, text and label. We make use of Text and label columns to build the classifier.

After loading dataset next step is Count vectorizer.

CountVectorizer

Building word count vectors with scikit-learn CountVectorizer for text classification. Now, we'll use pandas alongside scikit-learn to create a sparse text vectorizer. We can use it to train and test a simple supervised model. To begin, you'll set up a CountVectorizer and learn some of its features.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

```
X = df['text'] # independent variable
y = df['label'] # dependent variable
```

```
# Create training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33, random_state=53)
```

```
# Initialize a CountVectorizer object: count_vectorizer
count_vectorizer = CountVectorizer(stop_words='english')
```

```
# Transform the training data using only the 'text' column values: count_train
count_train = count_vectorizer.fit_transform(X_train)

# Transform the test data using only the 'text' column values: count_test
count_test = count_vectorizer.transform(X_test)
```

```
# Print the first 10 features of the count_vectorizer
print(count_vectorizer.get_feature_names()[:10])
```

```
['00', '000', '0000', '00000031', '000035', '00006', '0001', '0001pt', '000ft', '000km']
```


TfidfVectorizer For Text Classification

Similar to the sparse CountVectorizer created in the previous step, we'll work on creating tf-idf vectors for your documents. You'll set up a TfidfVectorizer and learn some of its features.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize a TfidfVectorizer object: tfidf_vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Transform the training data: tfidf_train
tfidf_train = tfidf_vectorizer.fit_transform(X_train)

# transform the test data: tfidf_test
tfidf_test = tfidf_vectorizer.transform(X_test)

# Print the first 10 features
print(tfidf_vectorizer.get_feature_names()[:10])

# Print the first 5 vectors of the tfidf training data
print(tfidf_train.A[:5])
```

['00', '000', '0000', '00000031', '000035', '00006', '0001', '0001pt', '000ft', '000km']

[[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]

To get a better idea of how the vectors work, we'll investigate them by converting them into pandas DataFrames.

```
count_df = pd.DataFrame(count_train.A, columns=count_vectorizer.get_feature_names())

# Create the TfidfVectorizer DataFrame: tfidf_df
tfidf_df = pd.DataFrame(tfidf_train.A, columns=tfidf_vectorizer.get_feature_names())

# Print the head of count_df
print(count_df.head())

# Print the head of tfidf_df
print(tfidf_df.head())

# Calculate the difference in columns: difference
difference = set(count_df.columns) - set(tfidf_df.columns)
print(difference)

# Check whether the DataFrame are equal
print(count_df.equals(tfidf_df))
```

[illegible]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523</
--	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-------

```
[5 rows x 56922 columns]
```

[illegible][illegible]

```
[5 rows x 56922 columns]
```

```
set()  
False
```

Model Building

Splitting The Data Into Train And Test

Divide the model into Train and Test data by Dependent and Independent Columns. Here in the dataset we need to separate the dependent and independent variables.

1. The independent variable in the dataset would be considered as 'x'
2. The dependent variable in the dataset would be considered as 'y'

Then we will split the data of independent and dependent variables.

Basically, as we know splitting of data is done to separate training and testing values to train and test the model. The percentage of splitting would be ideal when we take 67% of train data and 33% of test data to give it to the model.

Let's split the data into train and test by using Scikit learn as a package and then split the data as shown below.

[illegible]

Training And Testing A Classification Model With Scikit-Learn

Training and testing the "fake news" model with CountVectorizer Now we need to train the "fake news" model using the features you identified and extracted. Firstly we'll train and test a Naive Bayes model using the CountVectorizer data.

```
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, confusion_matrix

# Instantiate a Multinomial Naive Bayes classifier: nb_classifier
nb_classifier = MultinomialNB()

# Fit the classifier to the training data
nb_classifier.fit(count_train, y_train)

# Create the predicted tags: pred
pred = nb_classifier.predict(count_test)

# Calculate the accuracy score: score
score = accuracy_score(y_test, pred)
print(score)

# Calculate the confusion matrix: cm
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print(cm)
```

```
0.893352462936394
```

```
[[ 865  143]
 [   80 1003]]
```

Training And Testing The "Fake News" Model With TfidfVectorizer

The TfidfVectorizer Now that we have evaluated the model using the CountVectorizer, we'll do the same using with a Naive Bayes model.

```
nb_classifier = MultinomialNB()

# Fit the classifier to the training data
nb_classifier.fit(tfidf_train, y_train)

# Create the predicted tags: pred
pred = nb_classifier.predict(tfidf_test)

# Calculate the accuracy score: score
score = accuracy_score(y_test, pred)
print(score)

# Calculate the confusion matrix: cm
cm = confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
print(cm)
```

0.8565279770444764
[[739 269]
 [31 1052]]

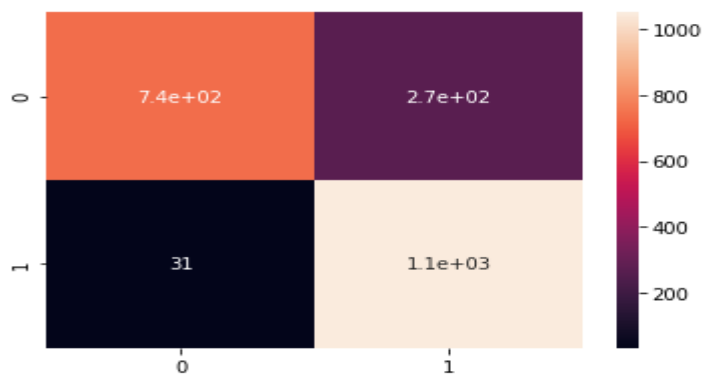
Show the confusion matrix and accuracy score for the model on the test data.

The output is meaningful, but looks like absolute garbage. So, we can make it beautiful with a heatmap from the Seaborn library for plotting the confusion matrix for count vectorizer.

```
0.8565279770444764  
[[ 739  269]  
 [  31 1052]]
```

```
#plot the confusion matrix for tf-idf vectorizer  
import seaborn as sns  
sns.heatmap(cm, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x286b09e21c8>
```



Simple NLP, Complex Problems

Improving the model. Our job is to test a few different alpha levels using the Tf-Idf vectors to determine if there is a better performing combination.

```
alphas = np.arange(0, 1, 0.1)

# Define train_and_predict()
def train_and_predict(alpha):
    # Instantiate the classifier: nb_classifier
    nb_classifier = MultinomialNB(alpha=alpha)

    # Fit to the training data
    nb_classifier.fit(tfidf_train, y_train)

    # Predict the labels: pred
    pred = nb_classifier.predict(tfidf_test)

    # Compute accuracy: score
    score = accuracy_score(y_test, pred)
    return score

# Iterate over the alphas and print the corresponding score
for alpha in alphas:
    print('Alpha: ', alpha)
    print('Score: ', train_and_predict(alpha))
    print()
```

Output of Improving model:

```
Alpha: 0.0
C:\Users\Shivam\anaconda3\lib\site-packages\sklearn\naive_bayes.py:512: UserWarning: alpha too small will result in numeric errors, setting alpha = 1.0e-10
'setting alpha = %.1e' % _ALPHA_MIN)
Score: 0.8813964610234337

Alpha: 0.1
Score: 0.8976566236250598

Alpha: 0.2
Score: 0.8938307030129125

Alpha: 0.30000000000000004
Score: 0.8900047824007652

Alpha: 0.4
Score: 0.8857006217120995

Alpha: 0.5
Score: 0.8842659014825442

Alpha: 0.6000000000000001
Score: 0.874701099952176

Alpha: 0.7000000000000001
Score: 0.8703969392635102

Alpha: 0.8
Score: 0.8660927785748446

Alpha: 0.9
Score: 0.8589191774270684
```

Inspecting Our Model

Now that we have built a "fake news" classifier, we'll investigate what it has learned. We can map the important vector weights back to actual words using some simple inspection techniques.

```
class_labels = nb_classifier.classes_

# Extract the features: feature_names
feature_names = tfidf_vectorizer.get_feature_names()

# Zip the feature names together with the coefficient array
# and sort by weights: feat_with_weights
feat_with_weights = sorted(zip(nb_classifier.coef_[0], feature_names))

# Print the first class label and the top 20 feat_with_weights entries
print(class_labels[0], feat_with_weights[:20])

# Print the second class label and the bottom 20 feat_with_weights entries
print(class_labels[1], feat_with_weights[-20:])

FAKE [(-11.316312804238807, '0000'), (-11.316312804238807, '000035'), (-11.316312804238807, '0001'), (-11.316312804238807, '0001pt'), (-11.316312804238807, '000km'), (-11.316312804238807, '0011'), (-11.316312804238807, '006s'), (-11.316312804238807, '007'), (-11.316312804238807, '007s'), (-11.316312804238807, '008s'), (-11.316312804238807, '0099'), (-11.316312804238807, '00am'), (-11.316312804238807, '00p'), (-11.316312804238807, '00pm'), (-11.316312804238807, '014'), (-11.316312804238807, '015'), (-11.316312804238807, '018'), (-11.316312804238807, '01am'), (-11.316312804238807, '020'), (-11.316312804238807, '023')]
REAL [(-7.742481952533027, 'states'), (-7.717550034444668, 'rubio'), (-7.703583809227384, 'voters'), (-7.654774992495461, 'house'), (-7.649398936153309, 'republicans'), (-7.6246184189367, 'bush'), (-7.616556675728881, 'percent'), (-7.545789237823644, 'people'), (-7.516447881078008, 'new'), (-7.448027933291952, 'party'), (-7.411148410203476, 'cruz'), (-7.410910239085596, 'state'), (-7.35748985914622, 'republican'), (-7.33649923948987, 'campaign'), (-7.2854057032685775, 'president'), (-7.2166878130917755, 'sanders'), (-7.108263114902301, 'obama'), (-6.724771332488041, 'clinton'), (-6.5653954389926845, 'said'), (-6.328486029596207, 'trump')]
```


Saving The Model

After building the model we have to save the model.

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. wb indicates write method and rd indicates read method.

This is done by the below code

```
: #saving the model
import pickle
pickle.dump(nb_classifier,open('model.pkl','wb'))
```

Build Flask Application

Flask Structure

- To build this flask application you should have basic knowledge of “HTML, CSS, Bootstrap, flask framework and python”
- For more information regarding [flask](#)
- Main Python Script
 - Let us build a flask file 'Fake_news.ipynb' which is a web framework written in python for server-side scripting. Let's see the step by step procedure for building the backend application.
 - You can also run this on spyder, given as app.py
 - App starts running when the “__name__” constructor is called in main.
 - Render_template is used to return html files.
 - “GET” method is used to take input from the user.

“POST” method is used to display the output to the user, Main .html pages are given

Libraries required for the app to run are to be imported.

```
#Importing the Libraries

#flask is use for run the web application.
import flask
#request is use for accessing file which was uploaded by the user on our application.
from flask import Flask, request,render_template
from flask_cors import CORS

#Python pickle module is used for serializing
# and de-serializing a Python object structure.
import pickle

#OS module in python provides functions for interacting with the operating system
import os

#Newspaper is used for extracting and parsing newspaper articles.
#For extracting all the useful text from a website.
from newspaper import Article

#URLlib is use for the urlopen function and is able to fetch URLs.
#This module helps to define functions and classes to open URLs
import urllib
```

Loading Flask and assigning the model variable

```
#Loading Flask and assigning the model variable
app = Flask(__name__)
CORS(app)
app=flask.Flask(__name__,template_folder='templates')

with open('model.pkl', 'rb') as handle:
    model = pickle.load(handle)
```

- Routing to the html Page

Basically we give routes of our html pages in order to showcase the UI. By giving the routes the built code in the html page is connected to our flask app. This is how a UI can be built and showcased.

```
@app.route('/') #default route
def main():
    return render_template('main.html')

#Receiving the input url from the user and using Web Scrapping to extract the news content
@app.route('/predict',methods=['GET','POST'])
```

We are routing the app to the html templates which we want to render. Firstly we are rendering the “main.html” template and from there we are navigating to our prediction page.

```
def predict():
    #Contains the incoming request data as string in case.
    url =request.get_data(as_text=True)[5:]

    #The URL parsing functions focus on splitting a URL string into its components,
    #or on combining URL components into a URL string.
    url = urllib.parse.unquote(url)

    #A new article come from Url and convert onto string
    article = Article(str(url))

    #To download the article
    article.download()

    #To parse the article
    article.parse()

    #To perform natural language processing ie..nlp
    article.nlp()
    #To extract summary
    news = article.summary
    print(type(news))

    #Passing the news article to the model and returning whether it is Fake or Real
    pred = model.predict([news])
    print(pred)
    return render_template('main.html', prediction_text='The news is "{}".format(pred[0]))
```

Here the route for prediction is given and necessary steps are performed in order to get the predicted output.

```
if __name__=="__main__":
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
```

Lastly, we run our app on the local host. Here we are running it on localhost:5000

- Run The app in local browser

```
(base) C:\Users\Shivam>cd I:\SmartBridge Projects\Fake-news-detection-master  
(base) C:\Users\Shivam>I:  
(base) I:\SmartBridge Projects\Fake-news-detection-master>python app.py_
```

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.

Now type “python app.py” command

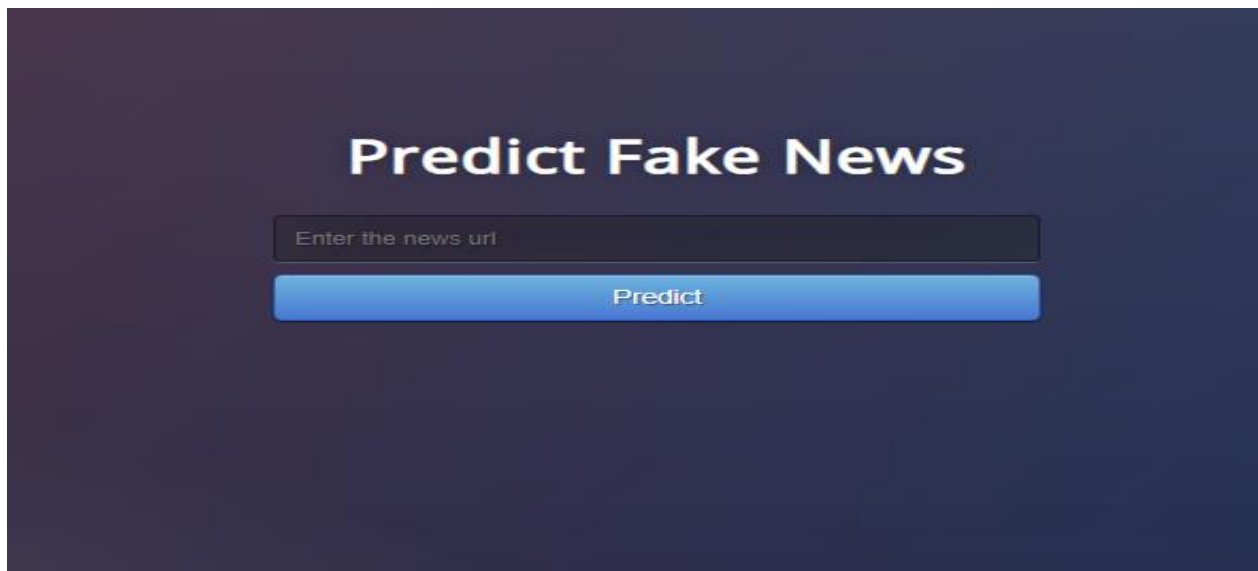
Navigate to the localhost where you can view your web page

Final UI

This is the main page of Fake News Detection where we give the URL as input and predict the output.

- Input the URL & predict the result.

Paste the URL of the article and click to predict the output whether it is Fake or Real.

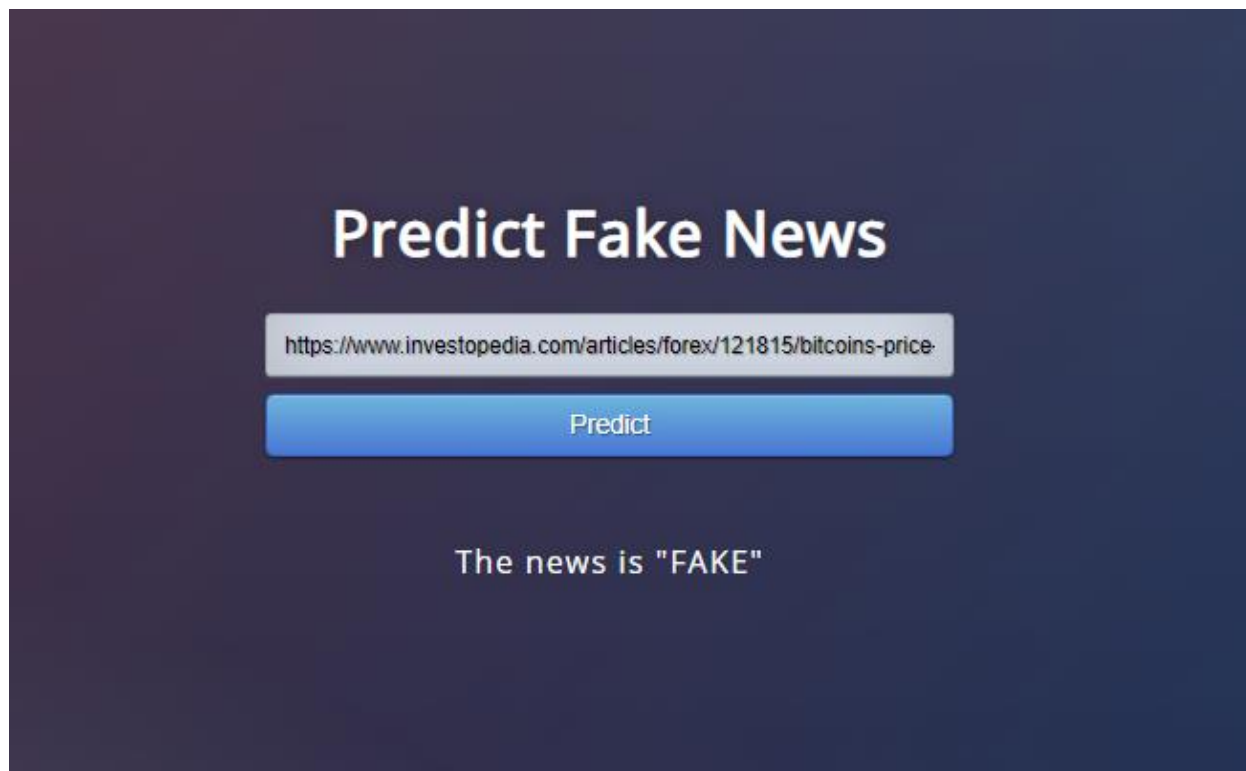


The image shows a web application interface with a dark blue background. At the top, the title "Predict Fake News" is displayed in a large, white, sans-serif font. Below the title, there is a text input field with a light gray border and the placeholder text "Enter the news url" in a small, light gray font. Directly beneath the input field is a prominent blue button with rounded corners and the word "Predict" written in white, centered text.

This article is Real based on our model.



This article is Fake based on our model.



CONCLUSION

The task of classifying news manually requires in-depth knowledge of the domain and expertise to identify anomalies in the text. In this research, we discussed the problem of classifying fake news articles using machine learning models and ensemble techniques. The data we used in our work is collected from the World Wide Web and contains news articles from various domains to cover most of the news rather than specifically classifying political news. The primary aim of the research is to identify patterns in text that differentiate fake articles from true news. We extracted different textual features from the articles using an LIWC tool and used the feature set as an input to the models. The learning models were trained and parameter-tuned to obtain optimal accuracy. Some models have achieved comparatively higher accuracy than others. We used multiple performance metrics to compare the results for each algorithm. The ensemble learners have shown an overall better score on all performance metrics as compared to the individual learners.

Fake news detection has many open issues that require attention of researchers. For instance, in order to reduce the spread of fake news, identifying key elements involved in the spread of news is an important step. Graph theory and machine learning techniques can be employed to identify the key sources involved in spread of fake news. Likewise, real time fake news identification in videos can be another possible future direction.

BIBLIOGRAPHY

References

1. A. Douglas, “News consumption and the new electronic media,” *The International Journal of Press/Politics*, vol. 11, no. 1, pp. 29–52, 2006. View at: [Publisher Site](#) | [Google Scholar](#)
2. J. Wong, “Almost all the traffic to fake news sites is from facebook, new data show,” 2016. View at: [Google Scholar](#)
3. D. M. J. Lazer, M. A. Baum, Y. Benkler et al., “The science of fake news,” *Science*, vol. 359, no. 6380, pp. 1094–1096, 2018. View at: [Publisher Site](#) | [Google Scholar](#)
4. S. A. García, G. G. García, M. S. Prieto, A. J. M. Guerrero, and C. R. Jiménez, “The impact of term fake news on the scientific community scientific performance and mapping in web of science,” *Social Sciences*, vol. 9, no. 5, 2020. View at: [Google Scholar](#)
5. A. D. Holan, *2016 Lie of the Year: Fake News*, Politifact, Washington, DC, USA, 2016.
6. S. Kogan, T. J. Moskowitz, and M. Niessner, “Fake News: Evidence from Financial Markets,” 2019, <https://ssrn.com/abstract=3237763>. View at: [Google Scholar](#)

A. Robb, “Anatomy of a fake news scandal,” *Rolling Stone*, vol. 1301, pp. 28–33, 2017. View at: [Google Scholar](#)
7. J. Soll, “The long and brutal history of fake news,” *Politico Magazine*, vol. 18, no. 12, 2016. View at: [Google Scholar](#)
8. J. Hua and R. Shaw, “Corona virus (covid-19) “infodemic” and emerging issues through a data lens: the case of China,” *International Journal of Environmental Research and Public Health*, vol. 17, no. 7, p. 2309, 2020. View at: [Publisher Site](#) | [Google Scholar](#)
9. N. K. Conroy, V. L. Rubin, and Y. Chen, “Automatic deception detection: methods for finding fake news,” *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015. View at: [Publisher Site](#) | [Google Scholar](#)
10. F. T. Asr and M. Taboada, “Misinfotext: a collection of news articles, with false and true labels,” 2019. View at: [Google Scholar](#)

11. K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, “Fake news detection on social media,” *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017. View at: [Publisher Site](#) | [Google Scholar](#)
12. S. Vosoughi, D. Roy, and S. Aral, “The spread of true and false news online,” *Science*, vol. 359, no. 6380, pp. 1146–1151, 2018. View at: [Publisher Site](#) | [Google Scholar](#)
13. H. Allcott and M. Gentzkow, “Social media and fake news in the 2016 election,” *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–236, 2017. View at: [Publisher Site](#) | [Google Scholar](#)
1. V. L. Rubin, N. Conroy, Y. Chen, and S. Cornwell, “Fake news or truth? using satirical cues to detect potentially misleading news,” in *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pp. 7–17, San Diego, CA, USA, 2016. View at: [Google Scholar](#)
2. H. Jwa, D. Oh, K. Park, J. M. Kang, and H. Lim, “exBAKE: automatic fake news detection model based on bidirectional encoder representations from transformers (bert),” *Applied Sciences*, vol. 9, no. 19, 2019. View at: [Publisher Site](#) | [Google Scholar](#)
3. H. Ahmed, I. Traore, and S. Saad, “Detection of online fake news using n-gram analysis and machine learning techniques,” in *Proceedings of the International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pp. 127–138, Springer, Vancouver, Canada, 2017. View at: [Publisher Site](#) | [Google Scholar](#)
4. W. Y. Wang, *Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News Detection*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2017.