### A project report on

Fake News Detection

### submitted in partial fulfillment of the requirements for

the award of marks in subject **Data Analytics** of the degree of

## Masters in Computer Application

**Tushar Sachdeva (Roll No.- 52212215)**

**Shubham Kumar Sinha (Roll No.-52212207)**

**Mohit (Roll No.-52212106)**



## DEPARTMENT OF COMPUTER APPLICATION

**National Institute of Technology Kurukshetra**

April, 2024

**BONAFIDE CERTIFICATE**

This is to certify that the project titled “**Fake News Detection”** is a Boanfide record of the work done by

**Tushar Sachdeva (Roll No.- 52212215)**

**Shubham Kumar Sinha (Roll No.-52212207)**

**Mohit (Roll No.-52212106)**

in partial fulfillment of the requirements for the award of marks in subject **Data Analytics** subject code **MCA-206** of **Masters in Computer Application** of the **National Institute of Technology, Kurukshetra**, during the year 2024.

**PROJECT GUIDE**

Dr. Kapil Gupta

**ABSTRACT**

This project endeavors to harness the power of data analysis and machine learning to combat the proliferation of fake news. Through the amalgamation of natural language processing techniques and diverse machine learning algorithms, we seek to construct a robust predictive model capable of discerning between genuine and fabricated news articles with high accuracy. Our methodology involves comprehensive preprocessing of textual data, feature extraction, and model training, followed by rigorous evaluation and deployment of the final model as a user-friendly tool. By empowering individuals to verify news authenticity in real-time, this initiative contributes to fostering a more informed and discerning society amidst the challenges of misinformation.

**ABOUT DATASET**

This dataset comprises various attributes aimed at aiding in the prediction of fake news. Each entry in the dataset is identified by a unique ID and includes the following parameters:

* ID: A unique identifier assigned to each news article.
* Title: The title of the news article, providing a succinct representation of its content.
* Author: The individual or entity credited with creating the news article.
* Text: The textual content of the article itself. Please note that this text may sometimes be incomplete or truncated.
* Label: This attribute serves as the target variable for prediction, indicating whether the news article is real or fake:

1: Denotes fake news.

0: Denotes real news.

By utilizing this dataset, researchers and practitioners can develop and evaluate models for detecting and classifying fake news, contributing to efforts aimed at combating misinformation and promoting media literacy.



**STEPS TAKEN IN OUR ANALYSIS PROCESS**

1) Data preprocessing

a) Data cleaning - After collecting a dataset, data cleaning should be performed to ensure that the dataset remains consistent and suitable for further analysis. For example, in our news dataset, we encountered instances where certain fields, such as article content or author information, were missing. To address this, we replaced the null values with an empty string using the pandas fillna() function.

**# replacing the null values with empty string**

**news\_dataset = news\_dataset.fillna('')**

b) Stemming - Stemming is a crucial step in the text preprocessing phase of our analysis process. It involves reducing words to their root or base form, which helps in standardizing text data and improving the efficiency of subsequent analysis techniques. For example in our dataset of news articles, we encountered variations of words derived from the same root. For instance, words like "actor," "actress," and "acting" all stem from the root word "act." By applying stemming, we transformed these variations into their base form, simplifying the text and reducing redundancy. This process not only enhances the consistency of our data but also facilitates more accurate analysis and classification of news articles.

**port\_stem = PorterStemmer()**

**def stemming(content):**

**stemmed\_content = re.sub('[^a-zA-Z]',' ',content)**

**stemmed\_content = stemmed\_content.lower()**

**stemmed\_content = stemmed\_content.split()**

**stemmed\_content = [port\_stem.stem(word) for word in stemmed\_content if not word in stopwords.words('english')]**

**stemmed\_content = ' '.join(stemmed\_content)**

**return stemmed\_content**

**news\_dataset['content'] = news\_dataset['content'].apply(stemming)**

c) Train test split - The first step in building our predictive system is splitting our dataset into training and test data. This is done using the train\_test\_split function from the sklearn.model\_selection module. The function takes four arguments: X (the feature matrix), Y (the target vector), test\_size (the proportion of the dataset to include in the test split), and random\_state (the seed used by the random number generator). The stratify parameter ensures that the proportion of values in the sample produced will be the same as the proportion of values provided to parameter stratify. Here, it ensures the training set and test set have approximately the same percentage of samples of each target class as the complete set.

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, stratify=Y, random\_state=2)**

After splitting the data, we train our model using logistic regression, a machine learning algorithm used for binary classification problems. We create an instance of the LogisticRegression class and then call the fit method to train the model on our training data.

**model = LogisticRegression()**

**model.fit(X\_train, Y\_train)**

The fit method adjusts the weight of our input features to minimize the difference between our model's predictions and the actual values. After the model is trained, it can be used to predict whether a given news article is real or fake.

This process forms the core of our predictive system for fake news detection. It's important to note that while the system is highly accurate, it's not infallible and should be used as a part of a comprehensive approach to news verification.

2) **Evaluation** – Our evaluation process is divided into two main parts: training data evaluation and test data evaluation. In the training data evaluation, we use our model to predict the outcomes for our training dataset. The model.predict(X\_train) function is used to make these predictions. We then compare these predicted outcomes with the actual outcomes (Y\_train). The accuracy\_score function is used to calculate the accuracy of our model on the training data. This gives us an idea of how well our model is learning from the training data.

In the test data evaluation, we follow a similar process. We use our model to predict the outcomes for our test dataset using model.predict(X\_test). We then compare these predicted outcomes with the actual outcomes (Y\_test). The accuracy\_score function is again used to calculate the accuracy of our model on the test data. This gives us an idea of how well our model is likely to perform on unseen data.

The accuracy scores for both the training and test data are then printed for analysis. A high accuracy score on the training data indicates good learning, while a high accuracy score on the test data indicates good generalization.

This evaluation process helps us ensure that our model is effective at predicting fake news, both in terms of learning from the data it is trained on and generalizing to new, unseen data. It’s a crucial step in the development of any machine learning model.

**# accuracy score on the training data**

**X\_train\_prediction = model.predict(X\_train)**

**training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)**

**print('Accuracy score of the training data : ', training\_data\_accuracy)**

**# accuracy score on the test data**

**X\_test\_prediction = model.predict(X\_test)**

**test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)**

**print('Accuracy score of the test data : ', test\_data\_accuracy)**

**3) Making predictive system -** Our predictive system for fake news detection is built on the principles of machine learning. The system is trained on a dataset of news articles, which are labeled as either ‘Real’ or ‘Fake’. This dataset serves as the foundation for our model to learn the distinguishing features between real and fake news.

The code snippet provided is a part of the prediction phase, which occurs after the model has been trained. Here’s a step-by-step explanation of the code:

X\_new = X\_test[3]: This line selects the fourth news article from our test dataset. The test dataset is a subset of our original dataset that the model has not seen during training. We use this unseen data to evaluate the performance of our model.

prediction = model.predict(X\_new): Here, we use the predict method of our trained model to predict whether the selected news article is real or fake. The predict method returns an array of predictions for the input samples.

print(prediction): This line prints the prediction result. The result is in the form of 0 (indicating ‘Real’) or 1 (indicating ‘Fake’).

The if-else statement prints ‘The news is Real’ if the prediction is 0, and ‘The news is Fake’ otherwise.

print(Y\_test(3)): Finally, we print the actual label of the news article from the test dataset. This allows us to compare the model’s prediction with the actual label.

This predictive system is a powerful tool in the fight against misinformation, providing a fast and reliable way to distinguish between real and fake news. It’s important to note that while the system is highly accurate, it’s not infallible and should be used as a part of a comprehensive approach to news verification.

**Questions Being Answered**

Q5) Compare and contrast different evaluation metrics for classification (e.g., accuracy, precision, recall, F1-score) and regression (e.g., mean squared error, R-squared). Discuss the strengths and limitations of each metric and when to use them.

A5) Classification metrics, like accuracy, precision, recall, and F1-score, help assess how well a model classifies data into categories. Accuracy tells us how often the model is right overall. Precision focuses on how many selected items are relevant. Recall tells us how many relevant items were selected. F1-score balances precision and recall.

Fake News Detection Model (Logistic Regression) vs. Fake News Detection Model (Random Forest):

1. Accuracy:

- Logistic Regression: 0.952

- Random Forest: 0.962

- Both models achieve high accuracy, with Random Forest slightly outperforming Logistic Regression.

2. Precision:

- Logistic Regression: 0.952

- Random Forest: 0.970

- Random Forest exhibits slightly higher precision compared to Logistic Regression, indicating fewer false positives.

3. Recall:

- Logistic Regression: 0.937

- Random Forest: 0.940

- Both models demonstrate similar recall values, indicating their ability to capture a high proportion of actual positive cases.

4. F1 Score:

- Logistic Regression: 0.944

- Random Forest: 0.955

- Random Forest achieves a slightly higher F1 score compared to Logistic Regression, reflecting better balance between precision and recall.

5. ROC AUC Score:

- Logistic Regression: 0.950

- Random Forest: 0.959

- Both models exhibit high ROC AUC scores, with Random Forest again showing a slight advantage over Logistic Regression in terms of discriminating between positive and negative cases.

Author Verification Model (Logistic Regression) vs. Author Verification Model (Random Forest):

1. Accuracy:

- Logistic Regression: 0.932

- Random Forest: 0.935

- Both models achieve comparable accuracy scores in author verification.

2. Precision:

- Logistic Regression: 0.901

- Random Forest: 0.891

- Logistic Regression demonstrates slightly higher precision compared to Random Forest, indicating fewer false positives in identifying authors.

3. Recall:

- Logistic Regression: 0.947

- Random Forest: 0.967

- Random Forest exhibits higher recall compared to Logistic Regression, suggesting it can identify a higher proportion of actual positive cases.

4. F1 Score:

- Logistic Regression: 0.923

- Random Forest: 0.928

- Random Forest achieves a slightly higher F1 score compared to Logistic Regression, indicating better balance between precision and recall in author verification.

5. ROC AUC Score:

- Logistic Regression: 0.934

- Random Forest: 0.939

- Both models demonstrate high ROC AUC scores, with Random Forest showing a slight advantage over Logistic Regression in discriminating between positive and negative cases in author verification.

**CODE**

import numpy as np

import pandas as pd

import re

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

import nltk

nltk.download('stopwords')

# printing the stopwords in English

print(stopwords.words('english'))

# loading the dataset to a pandas DataFrame

news\_dataset = pd.read\_csv('/content/train.csv')

news\_dataset.shape

# print the first 5 rows of the dataframe

news\_dataset.head()

# counting the number of missing values in the dataset

news\_dataset.isnull().sum()

# replacing the null values with empty string

news\_dataset = news\_dataset.fillna('')

# merging the author name and news title

news\_dataset['content'] = news\_dataset['author']+' '+news\_dataset['title']

print(news\_dataset['content'])

# separating the data & label

X = news\_dataset.drop(columns='label', axis=1)

Y = news\_dataset['label']

print(X)

print(Y)

port\_stem = PorterStemmer()

def stemming(content):

stemmed\_content = re.sub('[^a-zA-Z]',' ',content)

stemmed\_content = stemmed\_content.lower()

stemmed\_content = stemmed\_content.split()

stemmed\_content = [port\_stem.stem(word) for word in stemmed\_content if not word in stopwords.words('english')]

stemmed\_content = ' '.join(stemmed\_content)

return stemmed\_content

news\_dataset['content'] = news\_dataset['content'].apply(stemming)

print(news\_dataset['content'])

#separating the data and label

X = news\_dataset['content'].values

Y = news\_dataset['label'].values

print(X)

print(Y)

Y.shape

# converting the textual data to numerical data

vectorizer = TfidfVectorizer()

vectorizer.fit(X)

X = vectorizer.transform(X)

print(X)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, stratify=Y, random\_state=2)

model = LogisticRegression()

model.fit(X\_train, Y\_train)

# accuracy score on the training data

X\_train\_prediction = model.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print('Accuracy score of the training data : ', training\_data\_accuracy)

# accuracy score on the test data

X\_test\_prediction = model.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print('Accuracy score of the test data : ', test\_data\_accuracy)

X\_new = X\_test[3]

prediction = model.predict(X\_new)

print(prediction)

if (prediction[0]==0):

print('The news is Real')

else:

print('The news is Fake')

print(Y\_test[3])