**FAKE NEWS DETECTION USING NLP**

Project: Fake News Detection



**Introduction:**

* Fake news detection is a critical yet challenging problem in Natural Language Processing (NLP). The rapid rise of social networking platforms has not only yielded a vast increase in information accessibility but has also accelerated the spread of fake news. Thus, the effect of fake news has been growing, sometimes extending to the offline world and threatening public safety.
* Given the massive amount of Web content, automatic fake news detection is a practical NLP problem useful to all online content providers, in order to reduce the human time and effort to detect and prevent the spread of fake news.
* In this paper, we describe the challenges involved in fake news detection and also describe related tasks. We systematically review and compare the task formulations, datasets and NLP solutions that have been developed for this task, and also discuss the potentials and limitations of them.
* Based on our insights, we outline promising research directions, including more fine-grained, detailed, fair, and practical detection models. We also highlight the difference between fake news detection and other related tasks, and the importance of NLP solutions for fake news detection.

**Content for Project Phase 3:**

In this part you will begin building your project by loading and preprocessing the dataset.

Begin building the fake news detection model by loading and preprocessing the dataset.

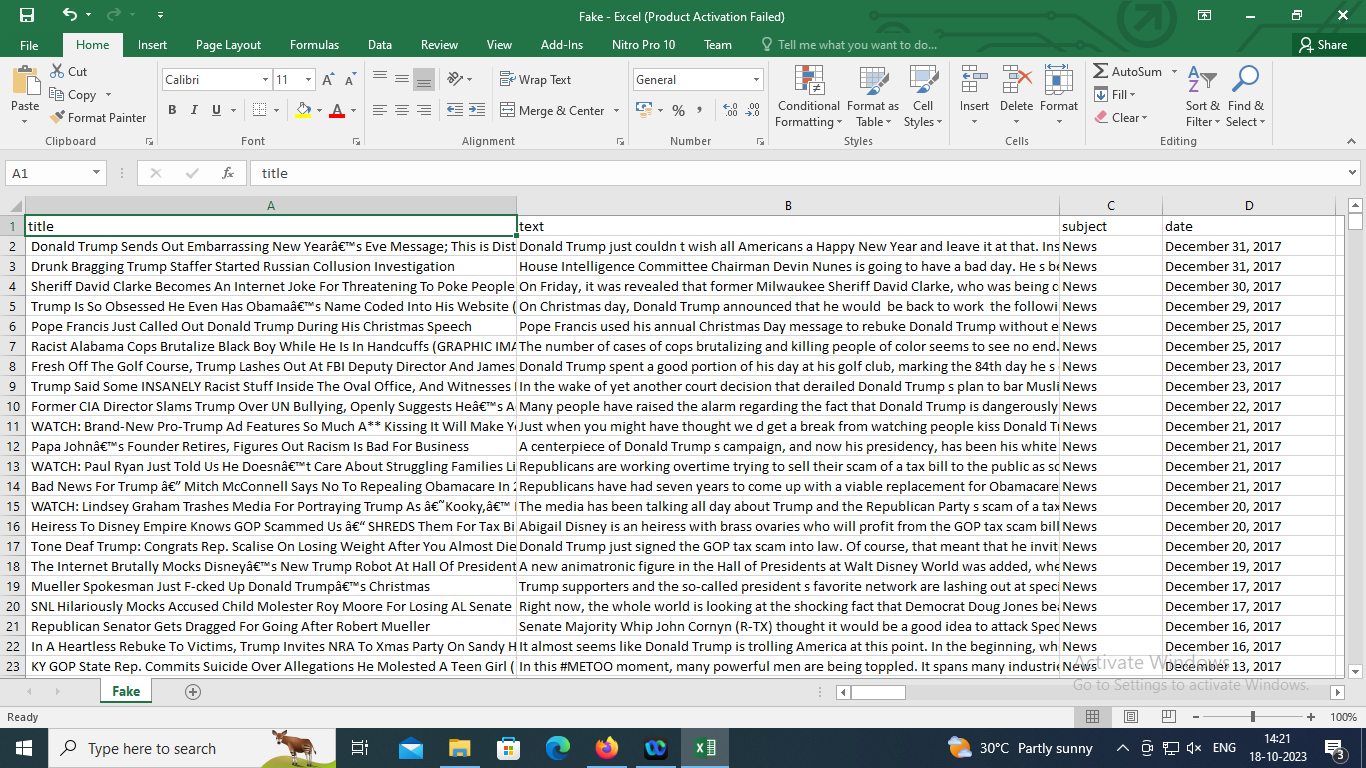
Load the fake news dataset and preprocess the textual data.

**Data Source:**

A good data source for fack news detection using nlp should be accurate,complex and accessible.

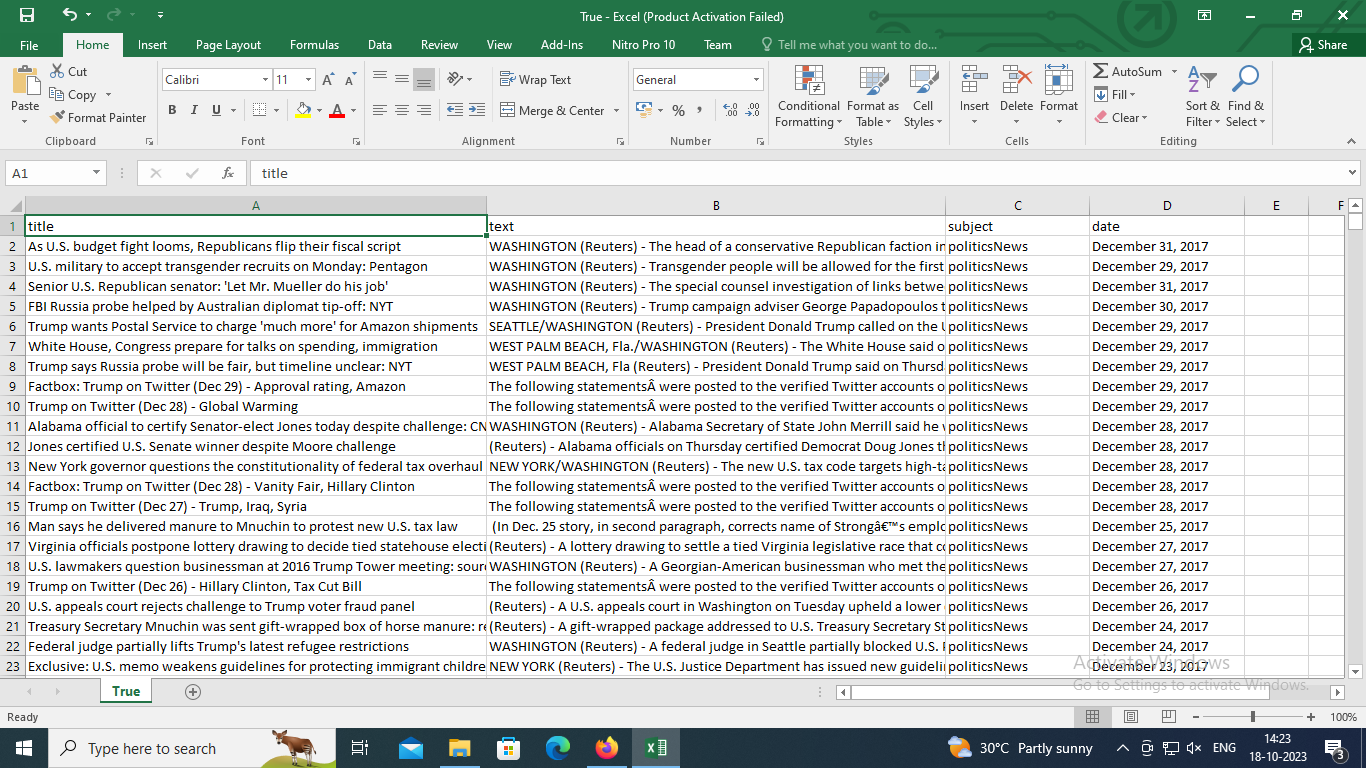
**Dataset Link:**[**https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset**](https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset)

Fake



Given dataset

True



Necessary steps to follow:

1.Import necessary libraries:

**Program:**

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import LabelEncoder

2.Load the dataset:

You can use a dataset in a CSV format. For example:

**Program:**

dataset = pd.read\_csv(‘fake\_news\_dataset.csv’)

3.Explore the dataset:

Check the structure of the dataset, including columns and labels.

4.Preprocess the text data:

* Handle missing values if any.
* Remove any irrelevant columns.
* Clean and preprocess the text data by removing punctuation, stopwords, and performing tokenization. You can use libraries like NLTK or spaCy for this.

5.Split the dataset into training and testing sets:

**Program:**

X = dataset[‘text’] # Text data

Y = dataset[‘label’] # Labels (e.g., ‘real’ or ‘fake’)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

6.Convert text data to numerical features:

Use techniques like TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert the text data into numerical format:

**Program:**

Tfidf\_vectorizer = TfidfVectorizer(max\_features=5000)

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

7.Encode labels:

Convert the categorical labels (e.g., ‘real’ and ‘fake’) into numerical values:

**Program:**

Label\_encoder = LabelEncoder()

Y\_train\_encoded = label\_encoder.fit\_transform(y\_train)

Y\_test\_encoded = label\_encoder.transform(y\_test)

Now, you have a preprocessed dataset with text data represented as numerical features. You can use this dataset to train a machine learning model, such as a classifier, for fake news detection using NLP techniques. Make sure to choose an appropriate NLP model (e.g., Naïve Bayes, Logistic Regression, or more advanced models like BERT) and evaluate its performance on the test set to assess its accuracy in detecting fake news.

**Code:**

Now, we will try to implement machine learning methods for the detection of fake news. Here we will have two datasets: "**Fake.csv"** and **"True.csv".**

One contains fake news, and the other contains true news.

**Importing Libraries:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

import re

import string

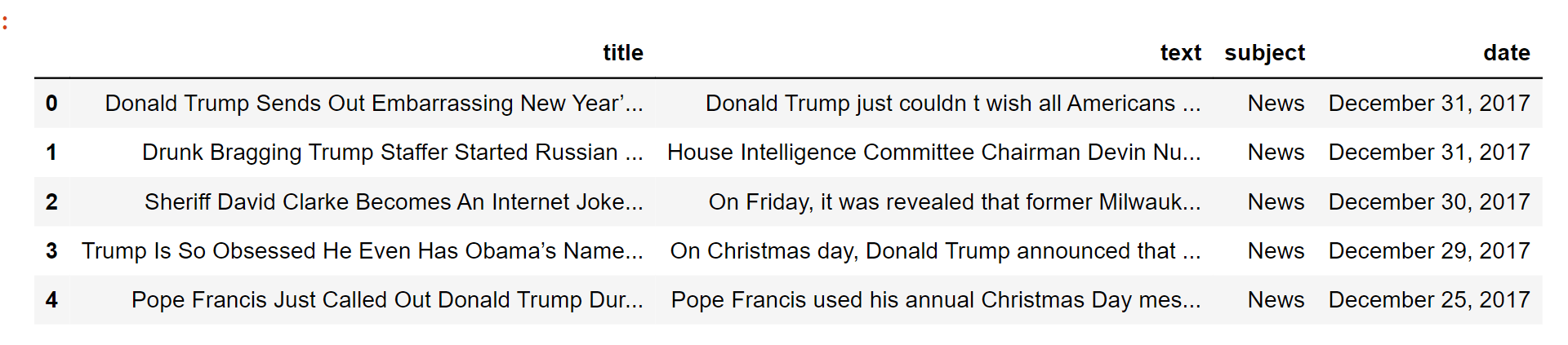
**Importing Dataset:**

dataframe\_fake = pd.read\_csv("Fake.csv")

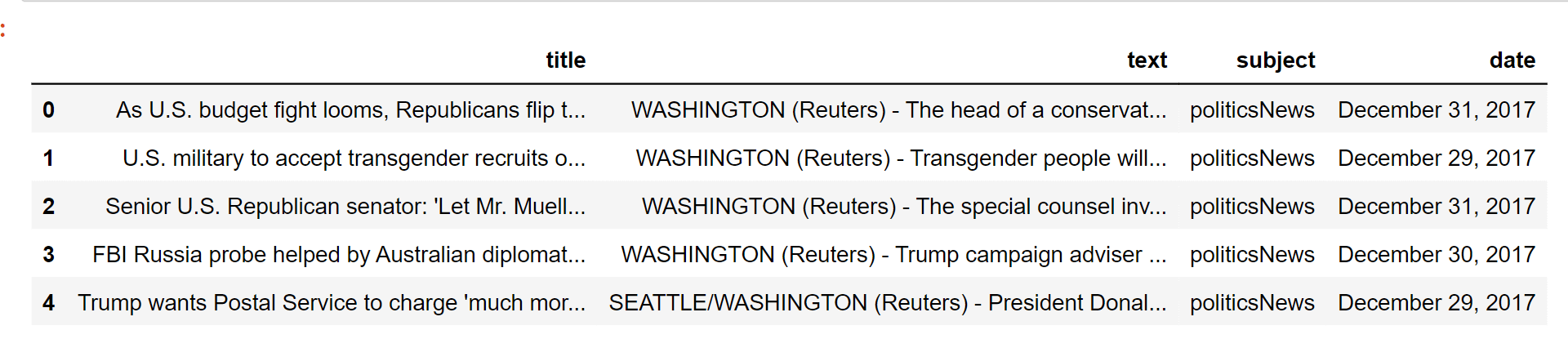
dataframe\_true = pd.read\_csv("True.csv")

dataframe\_fake.head()

**Output:**



dataframe\_true.head()

**Output:**

Now we will insert a column in both of the datasets named "class", which will be the target feature. In a fake dataframe, we will give a value of 1 to the class and on the other hand, with true, we will allocate 0.

dataframe\_true["class"] = 0

dataframe\_true["class"] = 1

# Now, we will look at the shape of both the dataset

dataframe\_fake.shape, dataframe\_true.shape

**Output:**

**( (23481,5 ) , (21417,5) )**

dataframe\_fake dataset contains 23481 rows and 5 columns.

dataframe\_true dataset contains 21417 rows and 5 columns.

Let's have some manual testing

# We will remove the last 10 rows for manual testing

dataframe\_fakedataframe\_fake\_manual\_testing = dataframe\_fake.tail(10)

for i in range(23480,23470,-1):

    dataframe\_fake.drop([i], axis = 0, inplace = True)

dataframe\_truedataframe\_true\_manual\_testing = dataframe\_true.tail(10)

for i in range(21416,21406,-1):

    dataframe\_true.drop([i], axis = 0, inplace = True)

# Let's have a look at the change in the shape of both the dataset

dataframe\_fake.shape, dataframe\_true.shape

**Output:**

**( (23471,5) , (21407,5) )**

If you look here, there is a decrease in the number of rows. It is because we took 10 rows from each dataset for manual testing.

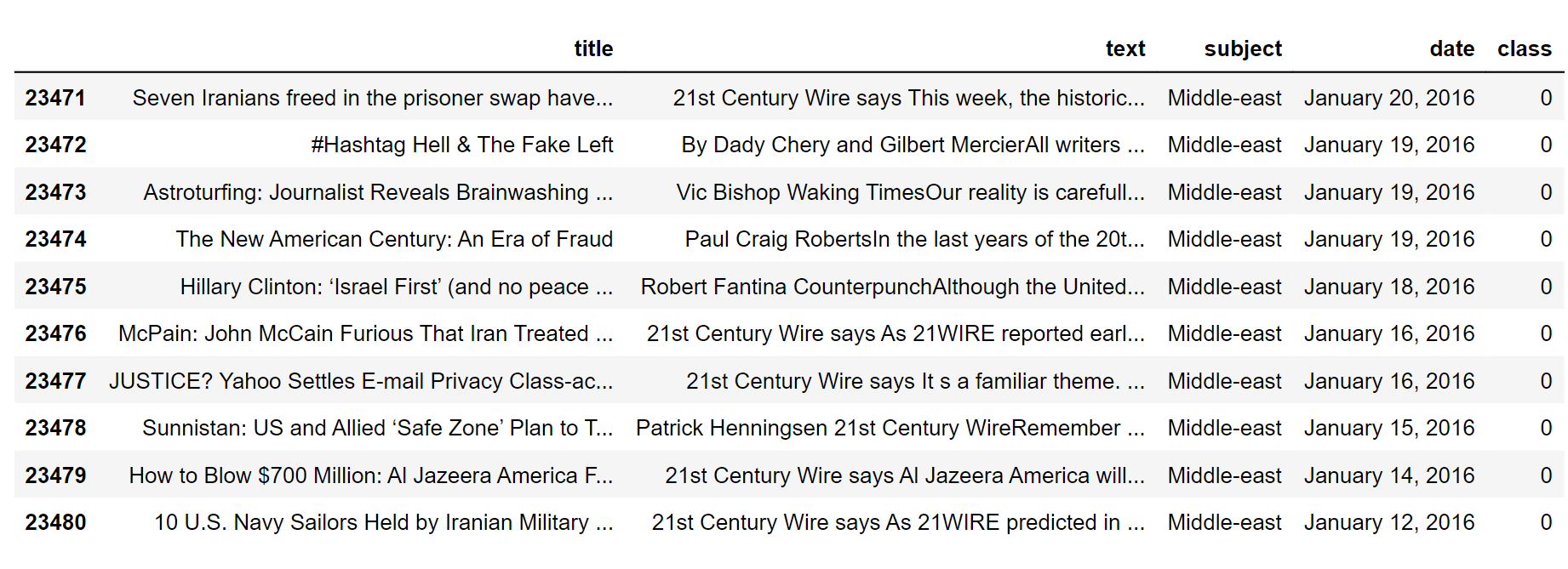
#Inserting the class column in both of the manual testing datasets

dataframe\_fake\_manual\_testing["class"] = 0

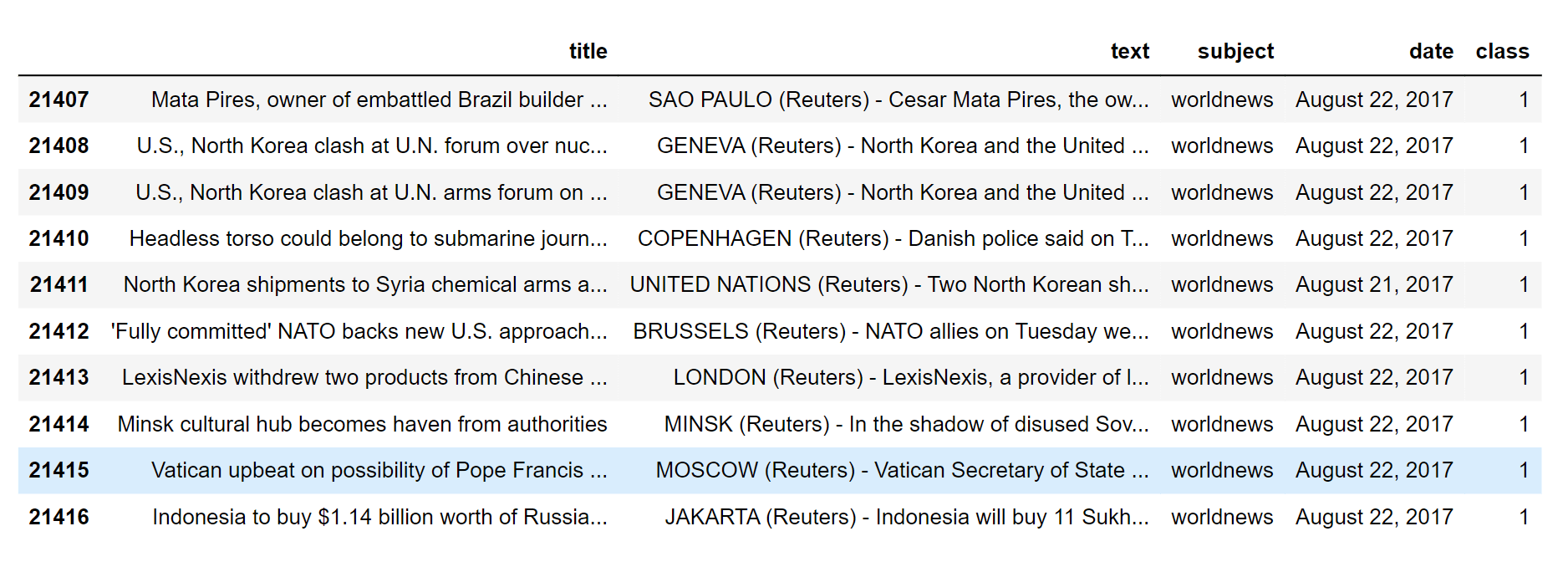
dataframe\_true\_manual\_testing["class"] = 1

dataframe\_fake\_manual\_testing.head(10)

**Output:**



dataframe\_true\_manual\_testing.head(10)

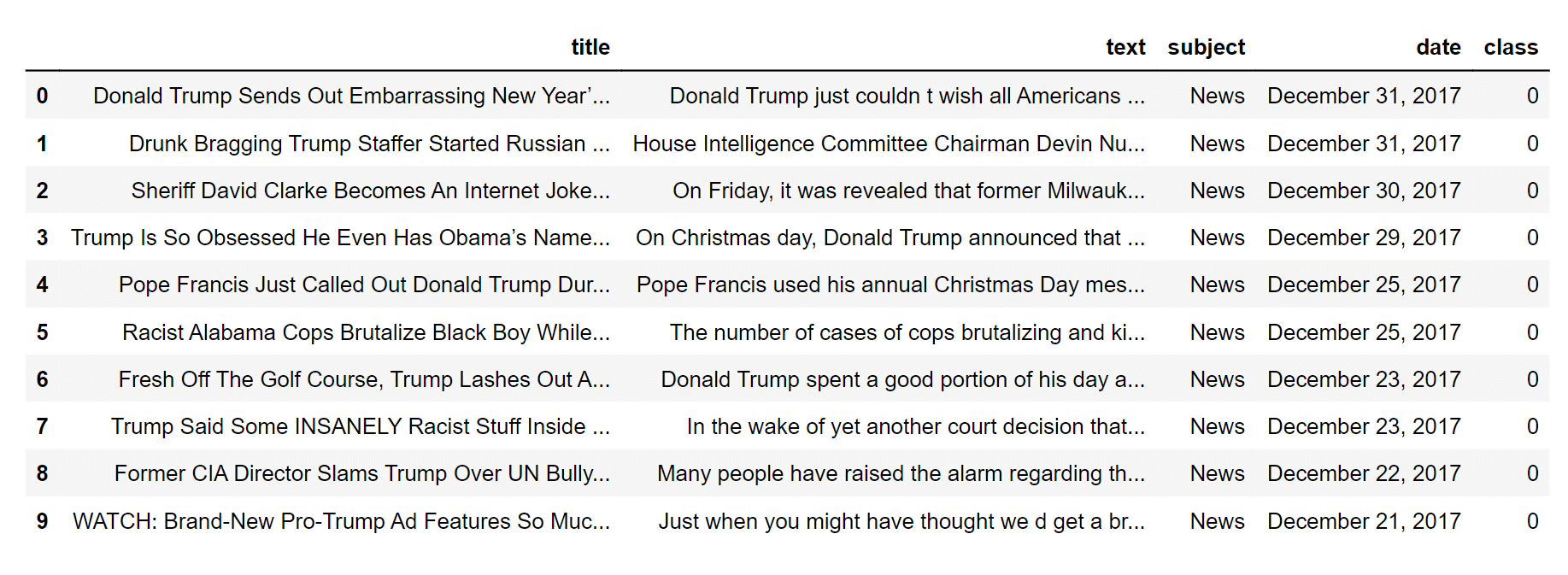
**Output:**

## Merging True and Fake Dataframes:

Here, we will merge '**dataframe\_fake**' and '**dataframe\_true**' to form a new dataset so that we perform the machine learning operations on it.

dataframe\_merge = pd.concat([dataframe\_fake, dataframe\_true], axis =0 )

dataframe\_merge.head(10)

**Output:**

When we have concat the datasets, the rows don't have randomness.

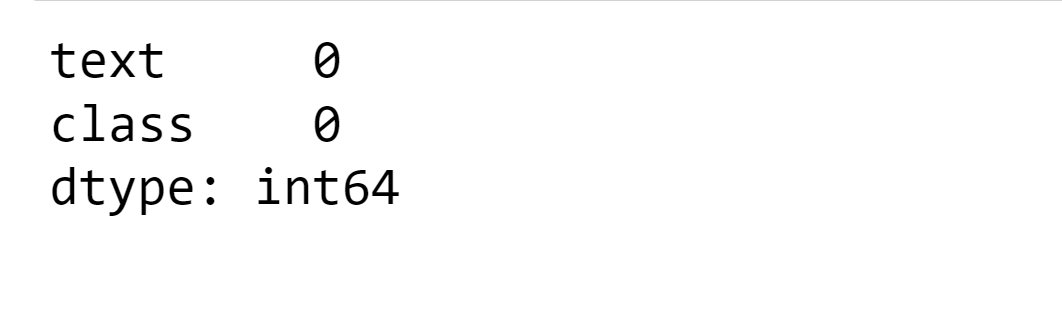
# We will remove the columns that are required for us

dataframe = dataframe\_merge.drop(["title", "subject","date"], axis = 1)

# Let's check if there are any null values in the dataset

dataframe.isnull().sum()

**Output:**



Luckily, we don't have any missing values in our dataset.

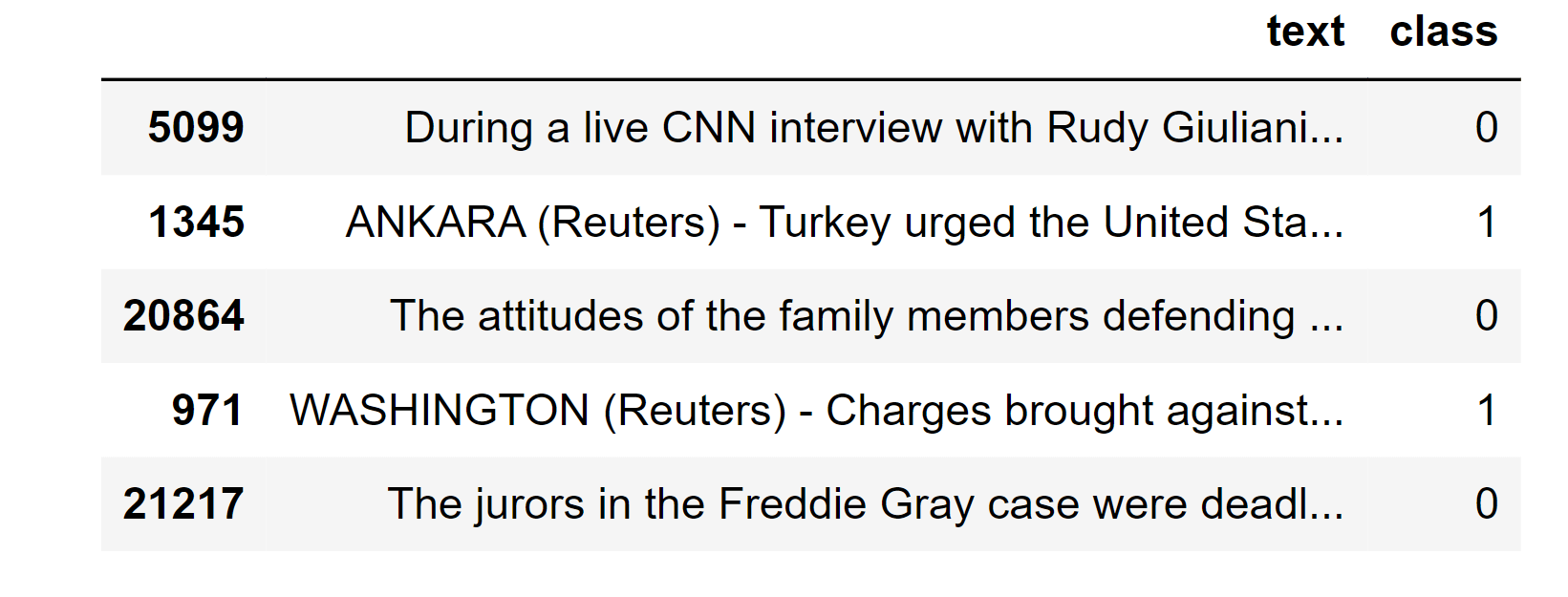
As we have only concat the two datasets so it will be true and fake datasets are arranged just after one another. So we need to create randomness in the dataset. We can shuffle the rows of the dataset.

# Here is the random shuffling of the rows in dataset

dataframedataframe = dataframe.sample(frac = 1)

dataframe.head()

**Output:**



**Conclusion:**

Fake news detection using natural language processing is a promising approach to combating fake news. Machine learning algorithms can analyze large datasets and identify patterns that are commonly found in fake news articles. By detecting fake news articles before they are widely disseminated, machine learning algorithms can prevent the harm caused by fake news. However, it is important to use diverse datasets and other techniques, such as fact-checking, to verify the authenticity of news articles.

.