PHASE 5: Submission Document

FAKE NEWS DETECTION USING NLP

Introduction:

[Fake news detection using Natural Language Processing (NLP) is a practical problem that can help reduce the time and effort needed to detect and prevent the spread of fake news](https://aclanthology.org/2020.lrec-1.747.pdf" \t "https://edgeservices.bing.com/edgesvc/_blank)

Problem statement:

Fake news can spread misinformation, which can lead to incorrect beliefs or assumptions about important topics. In some cases, fake news can pose a threat to public safety. For example, during a health crisis, false information about treatments or preventative measures can lead to harmful behaviour.

Therefore, detecting and combating fake news is crucial to maintain the integrity of information that we consume and share, and to ensure informed decision-making in various aspects of our lives.

Data set:

Here I will explain the data set.

In this python project, we have used the CSV dateset. True data set contains 21418 rows and 4 columns and False data set contains 23503 rows and 4 columns.

Data Collection:

This data set has four columns:

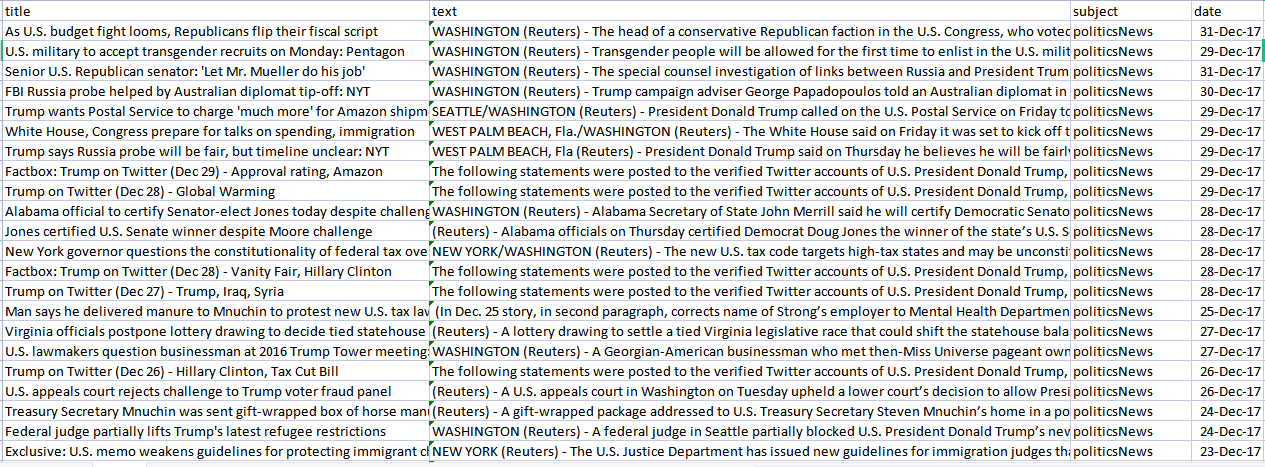
1. **title**: this represents the title of the news.

2. **text**: this column has the news itself.

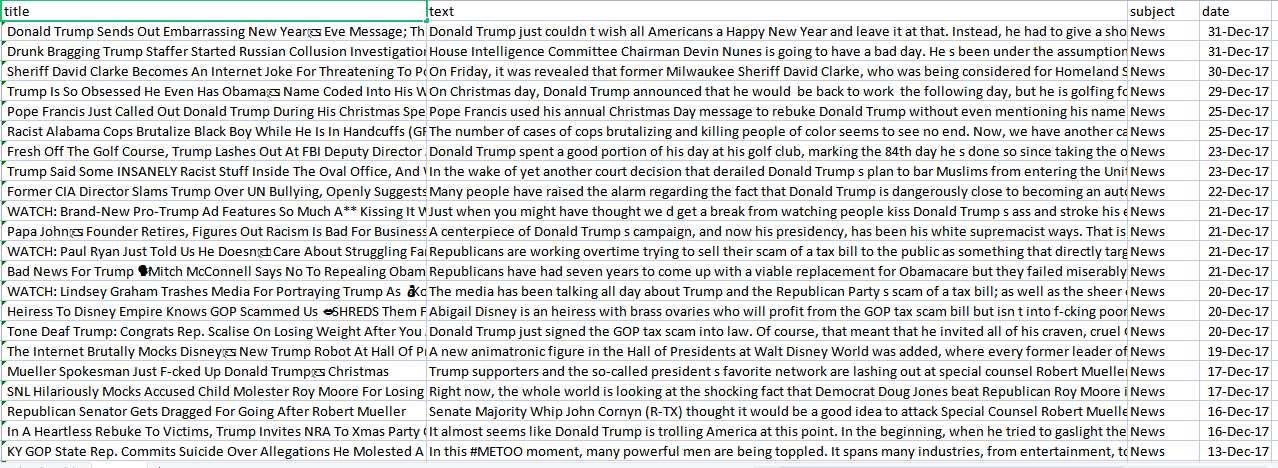
3.**subject:** this column shows subject of the news

4.**date:** this column shows when news has created

**TRUE NEWS:**



**FAKE NEWS:**

****

**Design  thinking process:**

Design thinking is a problem-solving and innovation methodology that focuses on understanding the needs and perspectives of end users. It is a human-centered approach to creating solutions and products.

**Data Preprocessing:**

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

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

Importing fake and true database:

>>dataframe\_fake.head()

dataframe\_true.head()

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

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

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.

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

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

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

**Feature Extraction:**

The cleaned text is converted into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer.

****TF (Term Frequency):****

In the document, words are present so many times that is called term frequency. In this section, if you get the largest values it means that word is present so many times with respect to other words. when you get word is parts of speech word that means the document is a very nice match.

Code:

>>from sklearn.feature\_extraction.text import TfidfVectorizer

vectorization = TfidfVectorizer()

xv\_train = vectorization.fit\_transform(x\_train)

xv\_test = vectorization.transform(x\_test)

Model Training:

A machine learning model is trained on the training data. [This involves feeding the data through the model, and using an optimization algorithm to adjust the model’s internal parameters to minimize the difference between the model’s predictions and the actual values](https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/" \t "https://edgeservices.bing.com/edgesvc/_blank).

* Logistic Regression (LR)
* Decision Tree Classifier (DT)
* Gradient Boosting Classifier (GB)
* Random Forest Classifier (RF)

Logistic Regression (LR):

[Logistic Regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance belongs to a given class](https://www.geeksforgeeks.org/understanding-logistic-regression/" \t "https://edgeservices.bing.com/edgesvc/_blank). [It is a kind of statistical algorithm, which analyzes the relationship between a set of independent variables and the dependent binary variables](https://www.geeksforgeeks.org/understanding-logistic-regression/" \t "https://edgeservices.bing.com/edgesvc/_blank).

Logistic Regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. [But instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1](https://www.geeksforgeeks.org/understanding-logistic-regression/).

Code:

>>from sklearn.linear\_model import LogisticRegression

LR = LogisticRegression()

LR.fit(xv\_train,y\_train)

pred\_lr=LR.predict(xv\_test)

LR.score(xv\_test, y\_test)

Decision Tree Classifier:

[A Decision Tree Classifier is a supervised machine learning algorithm used for both classification and regression tasks](https://www.geeksforgeeks.org/decision-tree/" \t "https://edgeservices.bing.com/edgesvc/_blank). [It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label](https://www.geeksforgeeks.org/decision-tree/" \t "https://edgeservices.bing.com/edgesvc/_blank)

[The Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets](https://www.geeksforgeeks.org/decision-tree/" \t "https://edgeservices.bing.com/edgesvc/_blank). [The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split](https://www.geeksforgeeks.org/decision-tree/" \t "https://edgeservices.bing.com/edgesvc/_blank)

Code:

>>from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()

DT.fit(xv\_train, y\_train)

pred\_dt = DT.predict(xv\_test)

DT.score(xv\_test, y\_test)

### Gradient Boost Classifier:

[Gradient Boosting Classifier is a powerful machine learning algorithm that combines several weak learners into strong learners](https://www.geeksforgeeks.org/ml-gradient-boosting/" \t "https://edgeservices.bing.com/edgesvc/_blank). [Each new model is trained to minimize the loss function, such as mean squared error or cross-entropy, of the previous model using gradient descent](https://www.geeksforgeeks.org/ml-gradient-boosting/" \t "https://edgeservices.bing.com/edgesvc/_blank).

[In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient](https://www.geeksforgeeks.org/ml-gradient-boosting/" \t "https://edgeservices.bing.com/edgesvc/_blank). [The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met](https://www.geeksforgeeks.org/ml-gradient-boosting/" \t "https://edgeservices.bing.com/edgesvc/_blank).

Code:

>>from sklearn.ensemble import GradientBoostingClassifier

GBC = GradientBoostingClassifier(random\_state=0)

GBC.fit(xv\_train, y\_train)

pred\_gbc = GBC.predict(xv\_test)

GBC.score(xv\_test, y\_test)

### Random Forest Classifier:

[Random Forest Classifier is a supervised machine learning algorithm that can be used for both classification and regression tasks](https://www.javatpoint.com/machine-learning-random-forest-algorithm" \t "https://edgeservices.bing.com/edgesvc/_blank). [It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model](https://www.javatpoint.com/machine-learning-random-forest-algorithm" \t "https://edgeservices.bing.com/edgesvc/_blank).

[The Random Forest Classifier creates a set of decision trees from a randomly selected subset of the training set](https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/" \t "https://edgeservices.bing.com/edgesvc/_blank). [It then collects the votes from different decision trees to decide the final prediction](https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/" \t "https://edgeservices.bing.com/edgesvc/_blank). [The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting](https://www.javatpoint.com/machine-learning-random-forest-algorithm" \t "https://edgeservices.bing.com/edgesvc/_blank).

Code:

>>from sklearn.ensemble import RandomForestClassifier

RFC = RandomForestClassifier(random\_state=0)

RFC.fit(xv\_train, y\_train)

pred\_rfc = RFC.predict(xv\_test)

RFC.score(xv\_test, y\_test)

**Predicting fake news :**

Here we are going to use all four models to check whether they are capable of detecting fake news. We have to check manually.

Code:

>>def output\_lable(n):

if n == 0:

 return "Fake News"

elif n == 1:

return "Real News"

def manual\_testing(news):

testing\_news = {"text":[news]}

new\_def\_test = pd.DataFrame(testing\_news)

new\_def\_test["text"] = new\_def\_test["text"].apply(wordopt)

new\_x\_test = new\_def\_test["text"]

new\_xv\_test = vectorization.transform(new\_x\_test)

  pred\_LR = LR.predict(new\_xv\_test)

pred\_DT = DT.predict(new\_xv\_test)

pred\_GBC = GBC.predict(new\_xv\_test)

 pred\_RFC = RFC.predict(new\_xv\_test)

return print("\n\nLR Prediction: {} \nDT Prediction: {} \nGBC Prediction: {} \nRFC Prediction: {}".format(output\_lable(pred\_LR[0]),output\_lable(pred\_GB[0]),output\_lable(pred\_RF[0]),output\_lable(pred\_DT[0])))

# Display the dialog box to enter news

news = str(input())

manual\_testing(news)

by entering the news text into the program, it uses trained machine learning models to predict whether the news is real or fake.

Conclusion:

The program preprocesses the text, extracts features, and feeds these features into the models for prediction. The models used in this program are Logistic Regression, Decision Tree Classifier, Gradient Boosting Classifier, and Random Forest Classifier. Each of these models makes a prediction, and the results are displayed.