**FAKE NEWS DETECTION USING NLP**

**TEAM MEMBERS:**

1. ARUN SURIYA CN - (2021504302)
2. SANTHANAKRISHNAN T - (2021504303)
3. MANOJKUMAR M - (2021504306)
4. SUDHARSAN K - (2021504707)
5. UMA N - (2021504717)

**PHASE 1**

**PROBLEM DEFINITION:**

In this part you will need to understand the problem statement and create a document on what have you understood and how will you proceed ahead with solving the problem. Please think on a design and present in form of a document.

**AIM:**

Develop an NLP-based system to detect and debunk fabricated news by analysing linguistic patterns and cross-referencing with credible sources, ensuring information accuracy and fostering media literacy.

**DATABASE LINK:**

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

**Step 1 - Data Collection:**

Gather a dataset of labelled news articles, where each article is labelled as either "real" or "fake". Websites like Kaggle often have datasets available for this purpose.

**Step 2 - Data Pre-processing:**

Clean and pre-process the text data. Steps may include:

Removing irrelevant information (e.g., special characters, HTML tags, punctuation).

**Tokenization:** Splitting text into words or phrases (tokens).

**Stop word removal:** Removing common words that don't carry much meaning.

**Lemmatization or stemming:** Reducing words to their base or root form.

**Step 3 - Feature Extraction:**

Use techniques like TF-IDF (Term Frequency-Inverse Document Frequency) or word embedding (e.g., Word2Vec, GloVe) to convert the pre-processed text into numerical features.

**Step 4 - Model Selection:**

Choose an appropriate machine learning model for classification. Common choices include:

**Naive Bayes:** Suitable for simple and fast classification.

**Support Vector Machines (SVM):** Effective for binary classification tasks.

**Random Forest:** Suitable for more complex classification tasks.

**Step 5 - Model Training:**

Train the chosen model using the pre-processed and feature-extracted data. Split the dataset into training and testing sets for evaluation.

**Step 6 - Model Evaluation:**

Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score. Fine-tune the model and experiment with different hyper parameters to improve performance.

**Step 7 - Integration and Deployment:**

Integrate the trained model into an application or platform where users can input news articles for classification. The model will output whether the article is real or fake.

**Step 8 - Continual Improvement:**

Continuously update and retrain the model with new data to adapt to evolving trends in fake news.

**Step 9 - Post-Deployment Monitoring:**

Monitor the model's performance and gather feedback to make further improvements.

**Step 10 - Incorporate External Signals:**

Consider incorporating additional features or external signals, such as the source credibility, author reputation, or social media reactions, to enhance the fake news detection system.

**PHASE 2**

In Phase 1, we identified a design solution to address a specific problem. Now, in Phase 2, we will outline the comprehensive steps required to transform this design into a practical innovation. Here are the steps to achieve this:

**PROJECT OBJECTIVES:**

**Primary of NLP:**

The primary objective of utilizing Natural Language Processing (NLP) for fake news detection is to develop robust algorithms and models that can effectively identify and combat misinformation and deceptive content in text. develop conversational agents or chatbots that can engage in natural and coherent conversations with users, providing assistance, information, or support.

**Identification and Classification**:

Develop models to automatically detect and classify news articles or text as either genuine or fake, utilizing features like language patterns, misinformation indicators, and credibility assessment

**Semantic Analysis:**

Analyse the semantic structure of news articles to determine if the content contradicts established facts, contains misleading information, or exhibits biased language**.**

**Contextual Understanding:**

Incorporate contextual understanding to distinguish between satirical content, opinion pieces, and deliberately misleading information, ensuring a nuanced assessment.

**Public Awareness and Education:**

Develop educational tools and resources to raise awareness among the public about the prevalence of fake news, its impact, and how to critically evaluate the information they encounter**.**

**Text Classification for Fake News Detection:**

**Problem Statement:**

Design a classification model that can accurately classify a given news article as either real or fake based on its text content.

**Approach:**

Utilize NLP techniques to preprocess and represent the text data. Develop a robust classification model (e.g., using neural networks, SVM, or ensemble methods) that can effectively differentiate between genuine and fake news based on linguistic patterns, semantic analysis, and other relevant features.

**Metrics for Evaluation:**

Accuracy, precision, recall, F1 score, and area under the ROC curve (AUC-ROC) to measure the model's performance in distinguishing between real and fake news.

**DESIGN THINKING AND APPROACH:**

**Data Collection and Preprocessing:**

* Gathering a diverse dataset of news articles, labeled as real or fake.
* Preprocessing the text data by tokenization, removing stopwords, and transforming words into numerical representations.

**Feature Engineering and Representation:**

* Extracting features from the pre-processed text using techniques like TF-IDF, word embeddings (Word2Vec, GloVe), or contextual embeddings (BERT, GPT).

**Model Selection and Training:**

* Choosing appropriate machine learning or deep learning models such as SVM, Naive Bayes, LSTM, or transformers.

**Linguistic and Semantic Analysis:**

* Analyzing linguistic patterns, sentence structures, and grammar to detect inconsistencies or anomalies in the text.
* Conducting semantic analysis to understand the meaning and context of the content, identifying potential misinformation.

**Classification and Verification:**

* Applying the trained model to classify news articles into real or fake categories based on learned patterns and features.
* Incorporating additional verification steps, like cross-referencing with trusted sources or fact-checking databases, to validate the classification.

**2. Data Collection:**

Text Lowercasing:

* Convert all text to lowercase to ensure consistency in the text and avoid duplicate words based on case differences.
* Remove any non-alphanumeric characters, including punctuation, special symbols, and any irrelevant characters that don't contribute to the analysis.

**Tokenization:**

* Break the text into smaller units called tokens, which could be words, subworlds, or characters. This facilitates further analysis at a granular level.

**Stop word Removal:**

* Remove common words (e.g., "and," "the," "is") that don't carry significant meaning and are unlikely to contribute to the analysis.

**Stemming and Lemmatization:**

* Reduce words to their base or root form using stemming or lemmatization, respectively, to handle variations of words (e.g., "running" to "run").

**Handling Numbers:**

* Decide whether to keep, remove, or replace numbers based on the specific analysis requirements. For certain tasks, it may be useful to retain numerical information.

**Handling HTML Tags and URLs:**

* If working with web data, remove HTML tags and URLs as they may not contribute to the analysis and could add noise.

**Handling Whitespaces:**

* Remove excess whitespaces or normalize them to a single space to ensure consistent spacing between words.

**Custom Cleaning based on Data Characteristics:**

* Depending on the nature of the data and the analysis, specific cleaning steps may be needed. For instance, handling mentions, hashtags, or emojis in social media data.

**Encoding and Vectorization:**

* Convert the cleaned text data into numerical vectors using techniques like TF-IDF, word embeddings (e.g., Word2Vec, GloVe), or transformer-based embeddings (e.g., BERT).

**DATA COLLECTION**

**Naive Bayes:**

* Naive Bayes is simple, efficient, and often used as a baseline model. It works well for text classification tasks like fake news detection, especially when the dataset is not very large.

**Support Vector Machines (SVM):**

* SVMs are effective for high-dimensional data and provide good accuracy. They can handle non-linear decision boundaries and are suitable for binary and multiclass classification.

**Logistic Regression:**

* Logistic Regression is a straightforward and interpretable model. It works well for binary classification tasks, making it suitable for distinguishing between real and fake news.

**Random Forest:**

* Random Forest is an ensemble learning method that provides high accuracy and can handle a mix of numerical and categorical features. It's robust and less prone to overfitting.

**Gradient Boosting Machines (e.g., XGBoost, LightGBM):**

* Gradient boosting algorithms often provide high accuracy by constructing an ensemble of weak learners. XGBoost and LightGBM are popular choices known for their efficiency and performance.
* Neural Networks (e.g., Feedforward Neural Networks, Convolutional Neural Networks).

**Long Short-Term Memory (LSTM) Networks:**

* LSTM networks, a type of recurrent neural network (RNN), are effective for sequential data and can capture long-term dependencies, making them suitable for text classification tasks.

**BERT and Transformer-based Models:**

* BERT and transformer-based models have shown state-of-the-art performance in NLP tasks. Fine-tuning pre-trained models like BERT for fake news detection can yield impressive results.

**PHASE 3**

**FAKE NEWS DETECTION**

A sort of sensationalist reporting, counterfeit news embodies bits of information that might be lies and is, for the most part, spread through web-based media and other online media. This is regularly done to further or force certain kinds of thoughts or for false promotion of products and is frequently accomplished with political plans. Such news things may contain bogus and additionally misrepresented cases and may wind up being virtualized by calculations, and clients may wind up in a channel bubble.

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.

2.IDF (Inverse Document Frequency)

In a single document, words are present so many times, but also available so many times in another document also which is not relevant. IDF is a proportion of how critical a term is in the whole corpus. collection of word Documents will convert into the matrix which contains TF-IDF features using   Tfidf Vectorizer.

3. PROJECT:

To get the accurately classified collection of news as real or fake we have to build a machine learning model. To deals with the detection of fake or real news, we will develop the project in python with the help of ‘sklearn’, we will use ‘Tfidf Vectorizer’ in our news data which we will gather from online media. After the first step is done, we will initialize the classifier, transform and fit the model. In the end, we will calculate the performance of the model using the appropriate performance matrix/matrices. Once will calculate the performance matrices we will be able to see how well our model performs.

4.DATA ANALYSIS:

In this python project, we have used the CSV dataset. The dataset contains 7796 rows and 4 columns.This dataset has four columns,

1. **title**: this represents the title of the news.
2. **author**: this represents the name of the author who has written the news.
3. **text**: this column has the news itself.
4. **label**: this is a binary column representing if the news is fake (1) or real (0).The dataset is open-sourced and can be found.

5.LIBRARIES:

The very basic data science libraries are sk learn, pandas, NumPy etc. and some specific libraries such as transformers.

6.DATA PROCESSING:

In data processing, we will focus on the text column on this data which actually contains the news part. We will modify this text column to extract more information to make the model more predictable. To extract information from the text column, we will use a library, which we know by the name of **‘NLTK’.**

7.REMOVING STOPWORDS:

These are the words that are used in any language used to connect words or used to declare the tense of sentences. This means that if we use these words in any sentence, they do not add much meaning to the context of the sentence so even after removing the stop words we can understand the context.

8.TOKENIZATION:

Tokenization is the process of breaking text into smaller pieces.Each word, special character, or number in a sentence can be depicted as a token in NLP.Tokenization is the process of breaking down a piece of code into smaller units called tokens.

9.CONVERTING LABEL:

The dataset has a Label column whose datatype is Text Category. The Label column in the dataset is classified into two parts, which are denoted as Fake and Real. To train the model, we need to convert the label column to a numerical one.

10.VECTORIZATION:

Vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which is used to find word predictions, word similarities/Sem Here, we are using vectorizer objects provided by Scikit-Learn which are quite reliable right out of the box.

Here, with ‘Tfidftransformer’ we are computing word counts using ‘Count Vectorizer’ and then computing the IDF values and after that the Tf-IDF scores. With ‘Tfidfvectorizer’ we can do all three steps at once. The code written above will provide with you a matrix representing your text.

11.MODELING:

After Vectorization, we split the data into test and train data.

I fit four ML models to the data, Logistic Regression, Naive-Bayes,

Decision Tree, and Passive-Aggressive Classifier. After that, predicted on the test set from the Tfidf Vectorizer and calculated the accuracy with accuracy score from scalar. metrics.

The very basic data science libraries are sklearn, pandas, NumPy e.t.c and some specific libraries such as transformers.

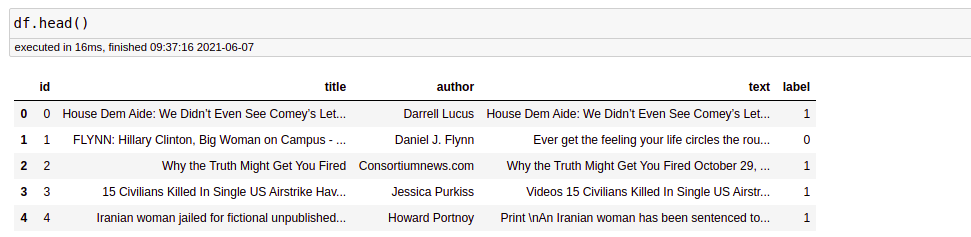
**PHASE 4**

**READ DATA FROM THE CSV FILE:**

df=pd.read\_csv('fake-news/train.csv')

df.head()

**output:**



Before proceeding, we need to check whether a null value is present in our dataset or not.

**df.isnull().sum()**

There is no null value in this dataset. But if you have null values present in your dataset then you can fill it. In the code given below, I will tell you how you can replace the null values.

df = df.fillna(' ')

**DATA PROCESSING:**

In data processing, we will focus on the text column on this data which actually contains the news part. We will modify this text column to extract more information to make the model more predictable. To extract information from the text column, we will use a library, which we know by the name of **‘NLTK’.**

**REMOVING STOPWORDS:**

These are the words that are used in any language used to connect words or used to declare the tense of sentences. This means that if we use these words in any sentence they do not add much meaning to the context of the sentence so even after removing the stop words we can understand the context.

**TOKENIZATION:**

Tokenization is the process of breaking text into smaller pieces.Each word, special character, or number in a sentence can be depicted as a token in NLP.Tokenization is the process of breaking down a piece of code into smaller units called tokens.

**CONVERTING LABEL:**

The dataset has a Label column whose datatype is Text Category. The Label column in the dataset is classified into two parts, which are denoted as Fake and Real. To train the model, we need to convert the label column to a numerical one.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

df.label = df.label.astype(str)

df.label = df.label.str.strip()

dict = { 'REAL' : '1' , 'FAKE' : '0'}

df['label'] = df['label'].map(dict)df.head()

To proceed further, we separate our dataset into features(x\_df) and targets(y\_df).

x\_df = df['total']

y\_df = df['label']

**VECTORIZATION:**

Vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which is used to find word predictions, word similarities/sem Here, we are using vectorizer objects provided by Scikit-Learn which are quite reliable right out of the box.

 from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.feature extraction. Text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

count\_vectorizer = CountVectorizer()

count\_vectorizer.fit\_transform(x\_df)

freq\_term\_matrix = count\_vectorizer.transform(x\_df)

tfidf = TfidfTransformer(norm = "l2")

tfidf.fit(freq\_term\_matrix)

tf\_idf\_matrix = tfidf.fit\_transform(freq\_term\_matrix)

print(tf\_idf\_matrix)

Here, with ‘Tfidftransformer’ we are computing word counts using ‘CountVectorizer’ and then computing the IDF values and after that the Tf-IDF scores. With ‘Tfidfvectorizer’ we can do all three steps at once.The code written above will provide with you a matrix representing your text. It will be a sparse matrix with a large number of elements in a Compressed Sparse Row format.The most used vectorizers are

**Count Victories:**

The most straightforward one, it counts the number of times a token shows up in the document and uses this value as its weight.

**Hash Vectorizer:**

This one is designed to be as memory efficient as possible. Instead of storing the tokens as strings, the vectorizer applies the hashing trick to encode them as numerical indexes. The downside of this method is that once vectorized, the features’ names can no longer be retrieved.  
**TF-IDF Victories:**

TF-IDF stands for “term frequency-inverse document frequency”, meaning the weight assigned to each token not only depends on its frequency in a document but also how recurrent that term is in the entire corpora. More on that here.

**MODELING:**

After Vectorization, we split the data into test and train data.

# Splitting the data into test data and train data

x\_train, x\_test, y\_train, y\_test = train\_test\_split(tf\_idf\_matrix,y\_df, random\_state=0)

I fit four ML models to the data, Logistic Regression, Naive-Bayes, Decision Tree, and Passive-Aggressive Classifier. After that, predicted on the test set from the Tfidf Vectorizer and calculated the accuracy with accuracy\_score() from sklearn. metrics. This one is designed to be as memory efficient as possible. Instead of storing the tokens as strings, the vectorizer applies the hashing trick to encode them as numerical indexes. The downside of this method is that once vectorized, the features’ names can no longer be retrieved.

**13. LOGISTIC REGRESSION:**

#LOGISTIC REGRESSION

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(x\_train, y\_train)

Accuracy = logreg.score(x\_test, y\_test)

print(Accuracy\*100)

Accuracy: 91.73%

**NAIVE BAYES:**

#NAIVE BAYES

from sklearn.naive\_bayes import MultinomialNB

NB = MultinomialNB()

NB.fit(x\_train, y\_train)

Accuracy = NB.score(x\_test, y\_test)

print(Accuracy\*100)

Accuracy: 82.32 %

**DECISION TREE:**

# DECISION TREE

from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()

clf.fit(x\_train, y\_train)

Accuracy = clf.score(x\_test, y\_test)

print(Accuracy\*100)

Accuracy: 80.49%

**PASSIVE AGGRESSIVE CLARIFIER:**

Their characteristic is that they remain passive when dealing with an outcome that has been correctly classified, and become aggressive when a miscalculation takes place, thus constantly self-updating and adjusting.

# PASSIVE-AGGRESSIVE CLASSIFIER

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import PassiveAggressiveClassifier

pac=PassiveAggressiveClassifier(max\_iter=50)

pac.fit(x\_train,y\_train)

#Predict on the test set and calculate accuracy

y\_pred=pac.predict(x\_test)

score=accuracy\_score(y\_test,y\_pred)

print(f'Accuracy: {round(score\*100,2)}%')

Output:Accuracy: 93.12%.

**CONCLUSION:**

The passive-aggressive classifier performed the best here and gave an accuracy of 93.12%. We can print a confusion matrix to gain insight into the number of false and true negatives and positives Fake news detection techniques can be divided into those based on style and those based on content, or fact-checking. Too often it is assumed that bad style (bad spelling, bad punctuation, limited vocabulary, using terms of abuse, ungrammaticality, etc.) is a safe indicator of fake news.

More than ever, this is a case where the machine’s opinion must be backed up by clear and fully verifiable indications for the basis of its decision, in terms of the facts checked and the authority by which the truth of each fact was determined.