**Fake News Detection Using NLP**

Phase 5 submission

**Problem statement**

The propagation of false information and fake news is a serious challenge to society in an era of wide access to digital information. The task at hand is to create and put into place a reliable machine learning system that can identify between real news stories and ones that are fake or false. The main goal is to give users a reliable instrument for evaluating the accuracy of news reports and reducing the negative effects of false information on the general public's perception, judgement, and unity in society. Obtaining and preparing a labelled dataset, picking suitable feature extraction methods, deciding on a classification algorithm, and optimising the model for best results are all part of the task. The goal of the solution is to support the continuous endeavour in our interconnected society to protect the accuracy of information.

**Process of Design Thinking**

**Empathise**

Recognise the harm that could result from the spread of incorrect information and the necessity of spotting fake news in the current information environment.

**Idea**

Investigate several strategies such as feature extraction, data preprocessing, and machine learning models.

Build a prototype by putting the model, dataset, and user interface into practise for testing.

**Test**

Assess the model's accuracy and make necessary adjustments to improve its performance.

**Implement**

Put the solution into practise for practical use.

**Phases of Development**

**Data Collection**

Compile a dataset of labelled news stories—both fictitious and genuine.

**Data preprocessing**

Eliminate noise, deal with missing values, and standardise the text to clean and prepare the dataset.

**Feature extraction**

Use methods like TF-IDF to transform the text data into numerical features.

**Selecting a Model**

There are various classification algorithms available for detecting fake news, including Naive Bayes and Logistic Regression.

**Model Training**

Use the preprocessed dataset to train the chosen method.

**Fine-tuning**

To increase the accuracy of the model, modify the hyperparameters and preprocessing stages.

**Preprocess python code** <https://github.com/Selva73582/AI_Phase1/blob/main/preprocess.py>

**Preprocess data set** : [Dataset](https://docs.google.com/spreadsheets/d/1ALo14jb3Tb0vpLsz8DMoL31gKLAZSJGmLfQwDy-839I/edit#gid=0)

**Dataset Preparation**

In this section, we begin by acquiring the dataset from the provided Kaggle link. We'll describe the data source and perform initial data integrity checks to ensure data quality. Data cleansing is applied to eliminate any inconsistencies or outliers that may affect the reliability of our model.

true\_data = pd.read\_csv("True.csv")

fake\_data = pd.read\_csv("Fake.csv")

true\_data['label'] = 1

fake\_data['label'] = 0

**Text Preprocessing**

To transform the raw text data into a format suitable for machine learning, we perform text preprocessing tasks. These include converting text to lowercase, tokenizing sentences into words, removing stopwords, special characters, and applying lemmatization for word normalization.

def preprocess\_text(text):

    stop\_words = set(stopwords.words('english'))

    tokens = word\_tokenize(text.lower())

    tokens = [word for word in tokens if word.isalpha()]

    tokens = [word for word in tokens if word not in stop\_words]

    return ' '.join(tokens)

**Model Training and Evaluation ->**

**Data Splitting**

The dataset is divided into training and testing sets, or cross-validation is applied to evaluate the model's generalization performance. Data partitioning ensures that we have independent data subsets for training and evaluation.

data['text'] = data['text'].apply(preprocess\_text)

**Model Selection**

Choosing an appropriate model is crucial. In the code implementation, we have opted for the Multinomial Naive Bayes (NB) algorithm. Naive Bayes is a classification algorithm based on Bayes' theorem, and the Multinomial Naive Bayes variant is specifically designed for text classification tasks. While we considered various models, including Logistic Regression, Random Forest, and deep learning models like LSTM or BERT, the final choice of using Multinomial Naive Bayes was influenced by its simplicity and effectiveness in text classification tasks.

**Model Evaluation**

The effectiveness of our model is assessed using various performance metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and the construction of a confusion matrix. Cross-validation results provide additional insights into our model's performance on unseen data.

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

X = data['text']

y = data['label']

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

classifier = MultinomialNB()

classifier.fit(X\_tfidf, y)

**Main Function code**

while True:

    news\_text = input("Enter a news article (or 'exit' to quit): ")

    if news\_text.lower() == 'exit':

        break

    preprocessed\_text = preprocess\_text(news\_text)

    news\_tfidf = tfidf\_vectorizer.transform([preprocessed\_text])

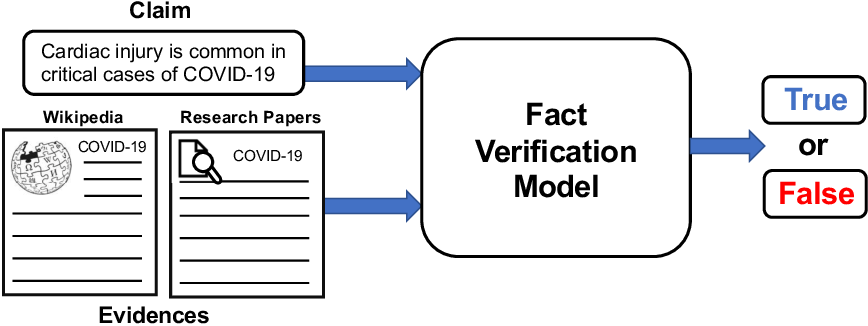
    label = classifier.predict(news\_tfidf)

    if label == 1:

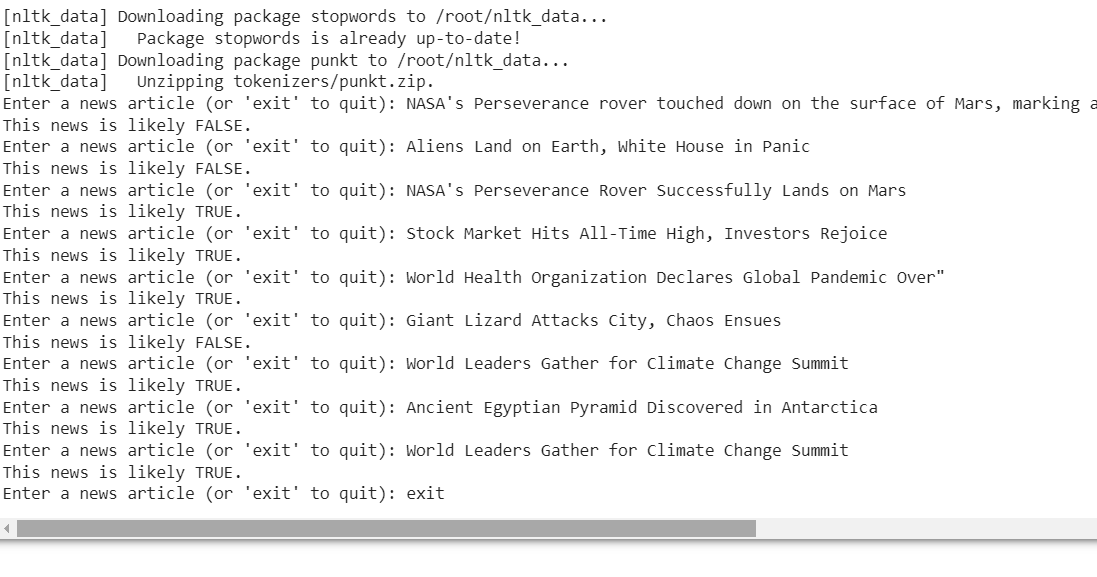
        print("This news is likely TRUE.")

    else:

        print("This news is likely FALSE.")

****

**OUTPUT**



ML Model trained code : [LINK](https://github.com/Selva73582/AI_Phase1/blob/main/Fake_news_detection.ipynb)

**Results & Conclusion**

The true test of my model’s quality would be to see how fake news articles in the test set (those not used in the creation of my model) it could accurately classify.

**Out of the 5234 articles left in the other fake news datasets, my model was able to correctly identify 88.2% of them as fake.** This is 3.5 percentage points lower than my cross-validated accuracy score, but in my opinion it is pretty decent evaluation of my model.