Assignment - 2

We will begin with our second major assignment and the deadline for building the solution to the same is given lenient for the first time so that you all get enough time to digest whatever we have learnt till date and go through all the links and resources we have shared till now.

Deadline: 28th June 2025
Submission form link:

https://docs.google.com/forms/

The task is as follows:
Fake News Detection on Twitter using NLP and Machine Learning
Develop a Fake News Detection System to analyze tweets and classify them as Real or Fake using Natural Language Processing (NLP) and Machine Learning models.
The dataset to be used is the one I will provide here:

1. What files do you need?
You'll need train.csv, test.csv and sample_submission.csv.
assignment1_Multimodal_sentimental_analysis
all the files are in this folder, you may download them in your local and work further.

2. What should you expect the data format to be?
Each sample in the train and test set has the following information:

- The text of a tweet
- A keyword from that tweet (although this may be blank!)
- The location the tweet was sent from (may also be blank)

3. What are you predicting?
You are predicting whether a given tweet is about a real disaster or not. If so, predict a 1. If not, predict a 0.

4. Files
train.csv - the training set
test.csv - the test set
sample_submission.csv - a sample submission file in the correct format

5. Columns
id - a unique identifier for each tweet
text - the text of the tweet
location - the location the tweet was sent from (may be blank)
keyword - a particular keyword from the tweet (may be blank)
target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not (0)

Modules and Libraries Required:

1. **pandas** – Data loading and manipulation
2. **numpy** – Numerical computations
3. **matplotlib**, **seaborn** – Data visualization
4. **nltk**, **spacy**, **re** – Text preprocessing (tokenization, stopwords, stemming)
5. **sklearn** – Machine learning algorithms (Logistic Regression, SVM)
6. **wordcloud** – Generating visual representations of text data
7. **CountVectorizer**, **TF-IDF Vectorizer** – Text vectorization
8. **train_test_split**, **classification_report** – Model evaluation

9. **transformers** – Pre-trained BERT embeddings (**optional** for advanced modeling)

Assignment Tasks include:

Data Exploration and Cleaning

1. Load the dataset using `pandas` and inspect its structure.
2. Check for null values and handle missing data appropriately.
3. Perform basic exploratory data analysis (EDA) using visualizations like bar plots, histograms, and word clouds to understand the distribution of fake and real tweets.

Text Preprocessing

1. Remove punctuation, special characters, and numbers from tweets using **regex**.
2. Convert text to lowercase and remove stopwords using **nltk**.
3. Apply stemming and lemmatization to normalize text using **nltk** and **spacy**.
4. Visualize the most frequently used words with **wordcloud** and **matplotlib**.

Feature Extraction

1. Transform the preprocessed text into numerical features using:
    - **Bag of Words (CountVectorizer)**
    - **TF-IDF (Term Frequency-Inverse Document Frequency)**
2. Explore word embeddings (e.g., Word2Vec or pre-trained BERT embeddings using **transformers**) for better feature extraction.

Model Building

1. Split the data into training and testing sets using **train_test_split**.
2. Train different machine learning models, including:
    - **Logistic Regression**
    - **Support Vector Machine (SVM)**
    - **Naive Bayes Classifier**
3. Evaluate model performance using accuracy, precision, recall, F1-score, and confusion matrix.

Hyperparameter Tuning

1. Use **GridSearchCV** or **RandomizedSearchCV** from **sklearn** to optimize hyperparameters for the models.
2. Compare performance metrics after tuning.

Visualization and Analysis

1. Plot ROC curves and Precision-Recall curves for all models.
2. Display misclassified examples and analyze reasons for failure.
3. Visualize confusion matrices to understand classification errors.

Task 7: Deployment as a Web App *(Optional for Bonus marks!)*

1. Create a simple web interface using **Flask** or **Streamlit**.
2. Allow users to input text and predict whether it is **Real** or **Fake** news.
3. Integrate the trained model into the app for real-time classification.

Deliverables:

- Submit a Jupyter Notebook (`fake_news_detection.ipynb`) with the following:
  - Clear explanations and comments.
  - Visualizations and insights from EDA.
  - Model training, evaluation, and comparisons.

- If the web app is implemented, submit the app code in a separate folder with documentation.