Assignment: Sentiment Analysis using NLP and Machine Learning

# Problem Statement

To implement a system that analyzes text data and predicts its sentiment (positive, negative, or neutral). The system should preprocess textual data, convert it into numerical features, train a machine learning model, and evaluate its performance. Additionally, visualization techniques such as word clouds and distribution plots should be used for better insights.

# Objectives

* To understand the basics of Natural Language Processing (NLP).
* To preprocess text data (tokenization, stopword removal, feature extraction).
* To implement sentiment classification using a machine learning model.
* To visualize sentiment distribution and most frequent words.
* To provide interactive prediction for unseen text.

# Software Used

Libraries:  
- Pandas, NumPy → Data handling  
- Scikit-learn → Feature extraction (TF-IDF) & model training  
- Matplotlib, Seaborn → Visualization  
- WordCloud → Word cloud generation  
- NLTK (optional) → Text preprocessing

# Theory

Sentiment Analysis is an application of NLP that classifies opinions expressed in text into categories such as positive, negative, or neutral. It is widely used in product reviews, social media monitoring, and customer feedback analysis.  
  
- Text Preprocessing: Includes cleaning, tokenization, and removal of stopwords.  
- Feature Extraction: TF-IDF (Term Frequency–Inverse Document Frequency) converts text into numerical vectors.  
- Classification Model: Naive Bayes, a probabilistic model, is efficient for text classification tasks.  
- Visualization: Word clouds and distribution plots help understand frequent terms and class balance.

# Methodology

1. Dataset Acquisition: Load sentiment dataset containing text and corresponding sentiment labels.
2. Preprocessing: Convert text to lowercase, remove stopwords, and vectorize using TF-IDF.
3. Visualization: Generate distribution plots of sentiments and word clouds for each sentiment.
4. Model Training: Train a Multinomial Naive Bayes classifier on the TF-IDF vectors.
5. Evaluation: Use accuracy, precision, recall, F1-score, and confusion matrix.
6. Interactive Prediction: Allow user to input a sentence and predict its sentiment in real-time.

# Working / Algorithm

Step 1: Import necessary libraries (pandas, sklearn, seaborn, matplotlib, wordcloud).

Step 2: Load the dataset (CSV containing `text` and `sentiment` columns).

Step 3: Preprocess text data (cleaning, stopword removal).

Step 4: Visualize sentiment distribution and generate word clouds.

Step 5: Split dataset into training and testing sets.

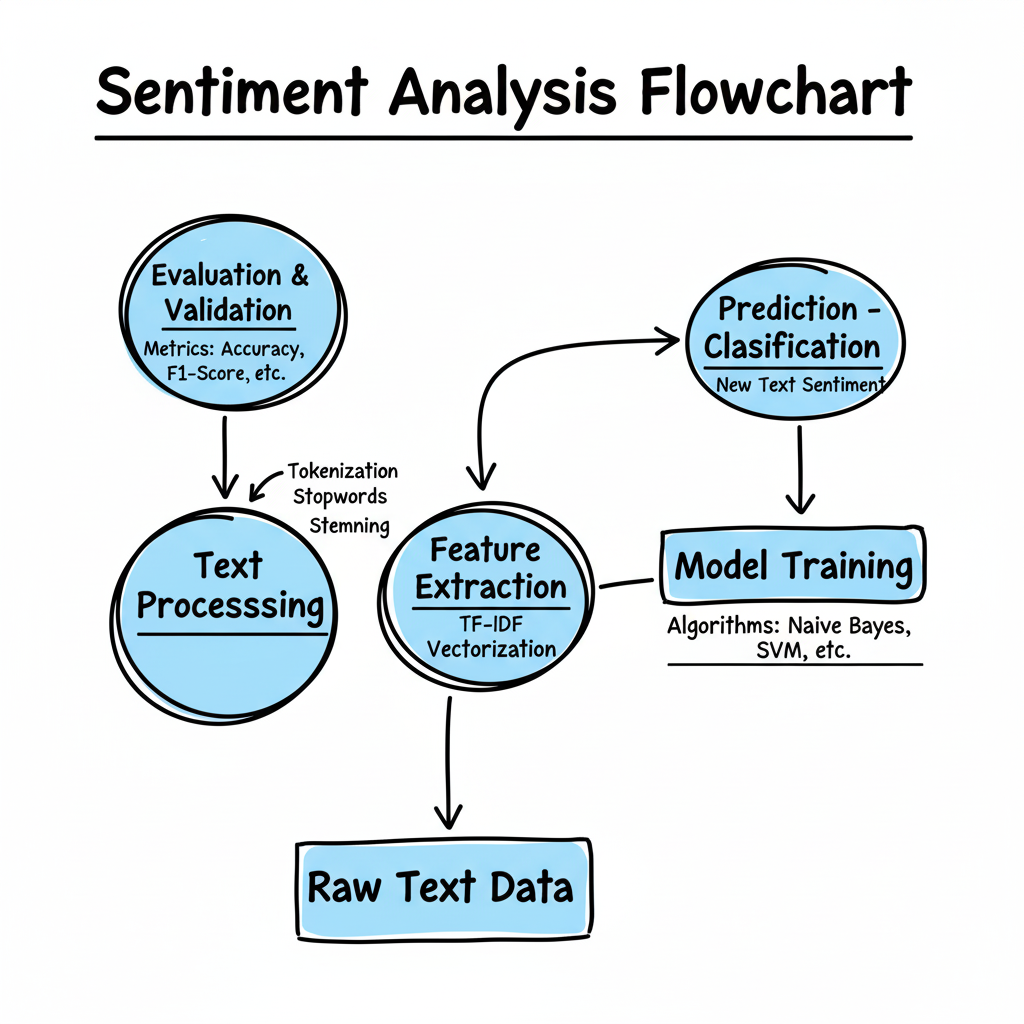
Step 6: Convert text into numerical features using TF-IDF Vectorizer.

Step 7: Train a Naive Bayes model on training data.

Step 8: Evaluate using classification report, accuracy, and confusion matrix.

Step 9: Implement an interactive function for custom sentiment prediction.

# Diagram



# Advantages

* Simple and efficient model for sentiment classification.
* Provides insights through visualization (word clouds, distribution).
* Interactive prediction allows real-world testing.

# Limitations

* Model performance depends on dataset quality.
* TF-IDF ignores word order and context.
* May misclassify sarcastic or complex sentences.

# Applications

* Customer feedback analysis
* Social media monitoring
* Product and service reviews
* Market research and opinion mining

# Conclusion

This project demonstrates the implementation of sentiment analysis using Natural Language Processing and Machine Learning techniques. With proper preprocessing and feature extraction using TF-IDF, a Naive Bayes model can effectively classify sentiments into positive, negative, and neutral. The inclusion of visualizations (distribution plots, word clouds) enhances understanding, while interactive prediction makes the system practical for real-time applications.