Text Classification with Scikit-Learn

Objective

Build and evaluate a machine learning model that classifies text documents into categories. You will apply text preprocessing, feature extraction, model training, and performance evaluation using Python and scikit-learn. Dataset Options

Students use the following datasets:
1. 20 Newsgroups dataset and classify news posts into topics like politics, science, or sports.

from sklearn.datasets import fetch_20newsgroups

Tasks

- 1. Data Loading
  - Load the dataset
  - Print sample texts and their labels.
- 2. Text Preprocessing
  - Convert text to lowercase.
  - Remove punctuation and stopwords.
  - Apply stemming and lemmatization.
- 3. Feature Extraction
  - Use TfidfVectorizer and CountVectorizer from scikit-learn.
  - Optional: how n-grams affect model performance (e.g., unigram vs bigram).
    vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,2))
    X = vectorizer.fit_transform(texts)
    ngram_range=(1,2) → **Unigrams + bigrams** (word pairs, e.g., “not good”).
- 4. Model Training
  - Train at least two classifiers (e.g., MultinomialNB, LinearSVC, LogisticRegression).
  - Split dataset into training/test sets.
- 5. Model Evaluation
  - Compute and display: Accuracy, Precision, Recall, F1-score.
  - Plot confusion matrix using matplotlib.
  - Compare model results and discuss findings.

- 6. Experimentation
  - Optional: Try dimensionality reduction (e.g., TruncatedSVD for LSA).
    svd = TruncatedSVD(n_components=100,…)
  - Use a Pipeline to combine preprocessing + model steps.
  - Perform hyperparameter tuning with GridSearchCV.

Expected Output

• Printed metrics and plots.
• Explanation of preprocessing decisions.
• Discussion on model performance and insights from misclassified examples.

Submission Requirements

• Submit a single Jupyter Notebook (.ipynb).

Lab_3-TextClass-FirstLastName.ipynb

• Include Markdown explanations and code comments.
• Provide plots and a short summary paragraph at the end.