Lab Assignment No. 4- NLP

Deep Learning

Team members:

Sachin Jadhav (202201040080)
Shivanjali Jagtap (202201040070)
Khushi Narad (202201040084)

Problem Statement -

The objective of this assignment is to implement NLP preprocessing techniques and build a text classification model using machine learning techniques.

Learning Outcomes:

1. Understand and apply NLP preprocessing techniques such as tokenization, stopword removal, stemming, and lemmatization.
2. Implement text vectorization techniques such as TF-IDF and CountVectorizer.
3. Develop a text classification model using a machine learning algorithm.
4. Evaluate the performance of the model using suitable metrics.

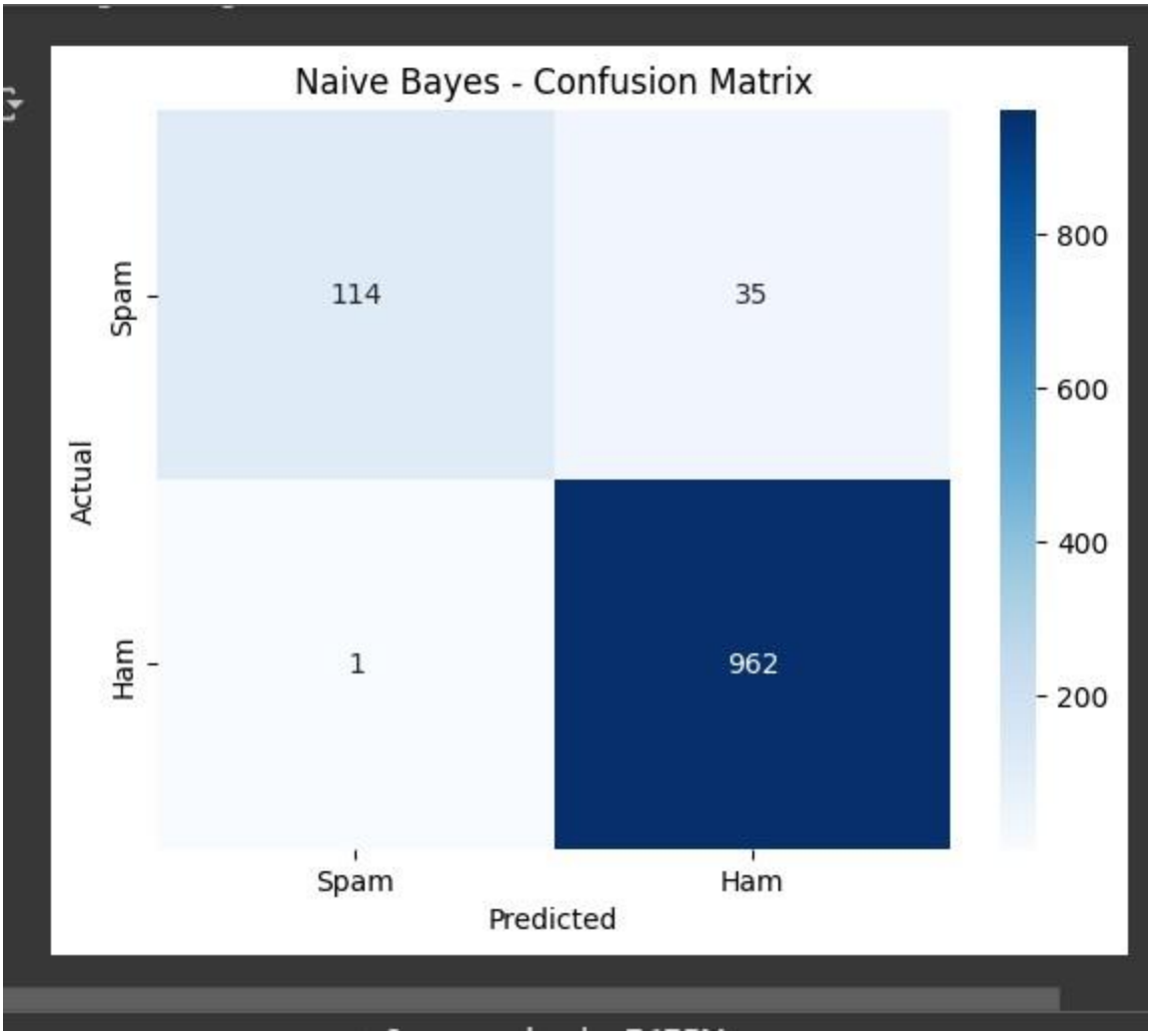
Github Link -

https://github.com/SACHIN9637/Deep-Learning/tree/main/Lab%20Assignment%20No._4

```
=== Naive Bayes ===
Accuracy: 0.9676258992805755
Classification Report:
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98       963
        spam       0.99      0.77      0.86       149

    accuracy                           0.97      1112
   macro avg       0.98      0.88      0.92      1112
weighted avg       0.97      0.97      0.97      1112
```



📌 Objective:

To develop and evaluate a Natural Language Processing (NLP) based text classification model that can accurately detect spam messages from regular (ham) messages using machine learning algorithms such as **Logistic Regression** and **Naive Bayes**.

🕮 Project Information:

🔍 Problem Statement:

In today's digital communication landscape, spam messages are a major nuisance and security threat. Automatically identifying such messages is crucial for maintaining clean and safe messaging platforms.

📂 Dataset:

- **Dataset Used:** [SpamAssassin Public Corpus / SMS Spam Collection Dataset]
- **Data Format:** CSV with columns for `label` (spam or ham) and `message` (text content).

The text is converted to lowercase to avoid treating the same word in different cases as different tokens (e.g., "Apple" vs "apple").

a. Tokenization

- Each sentence is split into words (tokens) using nltk.word_tokenize(). This converts raw text into a list of individual words.

b. Stopword Removal

- Commonly used words that don't carry much meaning (like "the", "is", "in") are removed using NLTK's stopwords list.

c. Stemming

- Words are reduced to their root forms using PorterStemmer (e.g., "running" becomes "run"). This helps group similar terms.

d. Lemmatization

- Words are converted to their dictionary base form using WordNetLemmatizer (e.g., "better" becomes "good"). Lemmatization is more accurate linguistically than stemming.

2. Text Vectorization

Since ML models can't understand raw text, the data is converted to numeric form using two popular methods:

a. CountVectorizer

- Converts the collection of text into a matrix of token counts.

b. TF-IDF Vectorizer

- Converts text into a matrix of term frequency–inverse document frequency values. It downweights commonly used words and boosts rare but informative ones.

3. Splitting the Data

- The transformed data is split into training and testing sets (e.g., 80% training, 20% testing) to evaluate how well the model generalizes to unseen data.

4. Model Building

Two models are trained using scikit-learn:

☑ Logistic Regression:

- A linear model used for classification problems. It models the probability of class membership.

☑ Multinomial Naive Bayes:

- A probabilistic classifier well-suited for text classification tasks. It uses word frequencies to predict class probabilities.

🧠 **Model Training:**

- Models Used:
  - **Logistic Regression** – good for binary classification
  - **Multinomial Naive Bayes** – particularly effective for text classification problems
- Training & Testing:
  - Data split into **80% training** and **20% testing**
  - Input features: **TF-IDF vectorized text**
  - Output labels: `"spam"` or `"ham"`

📊 **Evaluation Metrics:**

- **Accuracy** – overall correctness of the model
- **Precision** – how many predicted spams were actually spam
- **Recall** – how many actual spams were correctly predicted
- **F1-Score** – harmonic mean of precision and recall
- **Confusion Matrix** – visual representation of true vs predicted values

🧪 **Example Prediction:**

The model can take any custom text (e.g., a new SMS) and predict if it's likely to be *spam* or *ham*, demonstrating the practical utility of the system.