Spam Mail Prediction

A PROJECT REPORT

Submitted by

MANNAT MAHAJAN (24MCI10032)

In partial fulfillment for the award of the degree of

MASTERS OF COMPUTER APPLICATIONS

IN ARTIFICIAL INTELLIGENCE AND MACHINE

LEARNING

Chandigarh University

April 2025

BONAFIDE CERTIFICATE

Certified that this project report **"Spam Mail Prediction Using Machine Learning"** is the Bonafide work of **Mannat Mahajan (24MCI10032)** who carried out the project work under my/our supervision.

Subject Teacher

Dr. KRISHAN TULI

📌 Project Overview

The Spam Mail Prediction System is a machine learning-based application designed to automatically classify emails as "spam" or "not spam" (ham). With the increasing volume of digital communication, distinguishing between legitimate emails and spam has become a critical task. This project leverages natural language processing (NLP) and classification algorithms to detect spam messages accurately and efficiently.

🛠️ Tools & Technologies Used

- **Language:** Python
- **Web App Framework:** Flask, Streamlit
- **Libraries:** Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
- **Algorithm:** Random Forest Classifier
- **Dataset:** mail_data.csv

🧠 Machine Learning Pipeline

1. Data Collection

- Load the dataset (`mail_data.csv`) using **pandas**.
- This dataset contains messages labeled as `spam` or `ham`.

---

2. Data Preprocessing

- **Handling Missing Values**: Replace any null entries with empty strings.
- **Text Cleaning**:
  - Remove special characters using regular expressions (`re`).
  - Convert text to lowercase.
  - Remove extra spaces.
- **Label Encoding**: Convert categorical labels:
  - `spam` → `0`
  - `ham` → `1`

---

3. Text Feature Extraction (Vectorization)

- Use **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert raw text into numerical vectors.
- Parameters used:
  - `stop_words='english'`: Removes common English stop words.
  - `ngram_range=(1,2)`: Includes both unigrams and bigrams.

---

4. Train-Test Split

- Use `train_test_split` from **scikit-learn** to split the data:
  - **80% for training**, **20% for testing**
  - Ensures that the model is evaluated on unseen data.

---

5. Model Training

- Train a **Random Forest Classifier**:
  - `n_estimators=100`: Builds 100 decision trees.
  - `class_weight='balanced'`: Handles class imbalance between spam and ham.
  - `random_state=42`: Ensures reproducibility.

---

6. Model Evaluation

- Predict outcomes for both training and testing datasets.
- Metrics used:
  - **Accuracy**
  - **Classification Report** (includes precision, recall, F1-score)
  - **Confusion Matrix** (can be added for further clarity)

---

7. Prediction on New Data

- Take a custom input message from the user.
- Clean and transform it using the same TF-IDF vectorizer.
- Predict the class (spam or ham) and return prediction with confidence.

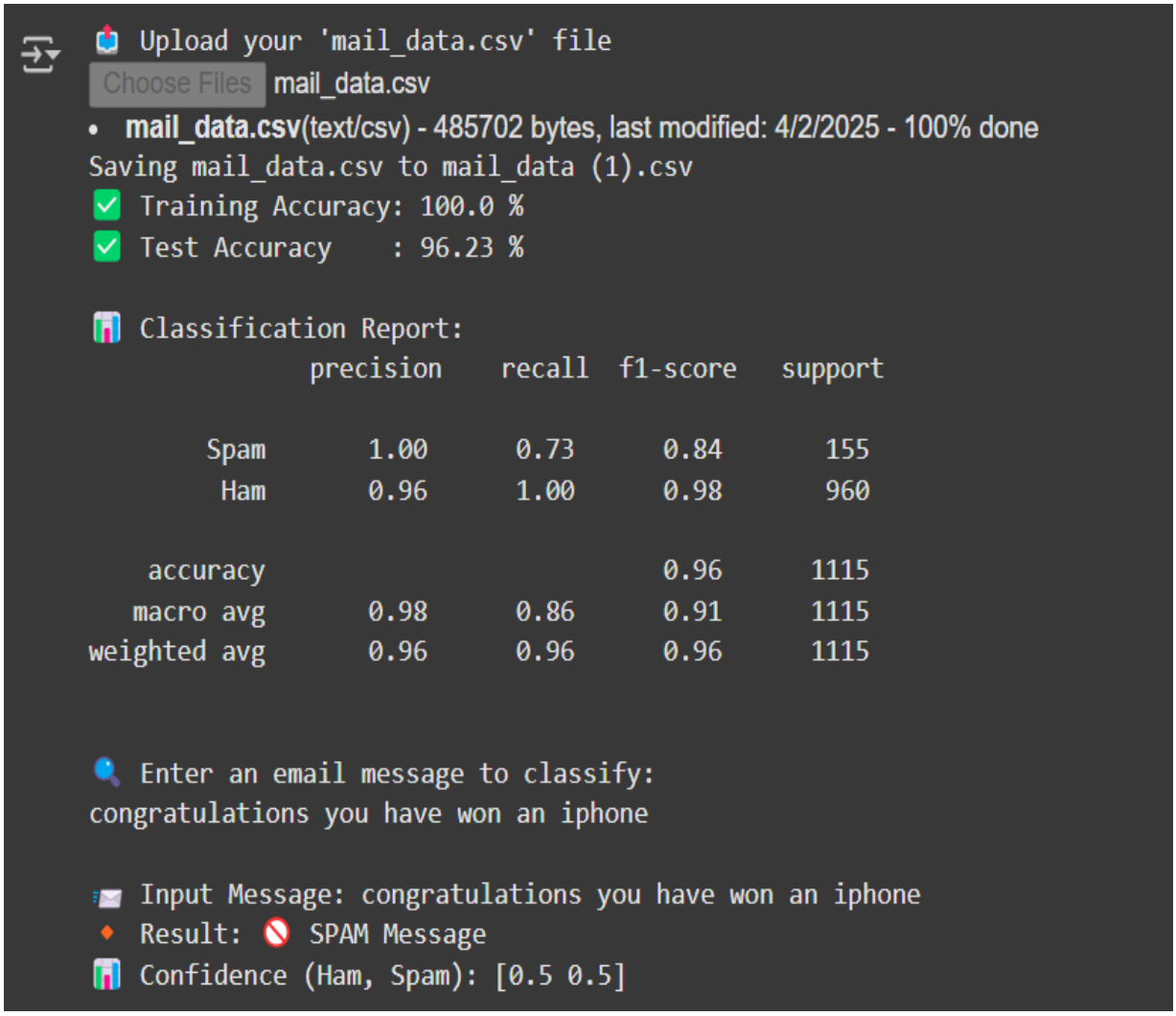
(Optional) 8. Deployment

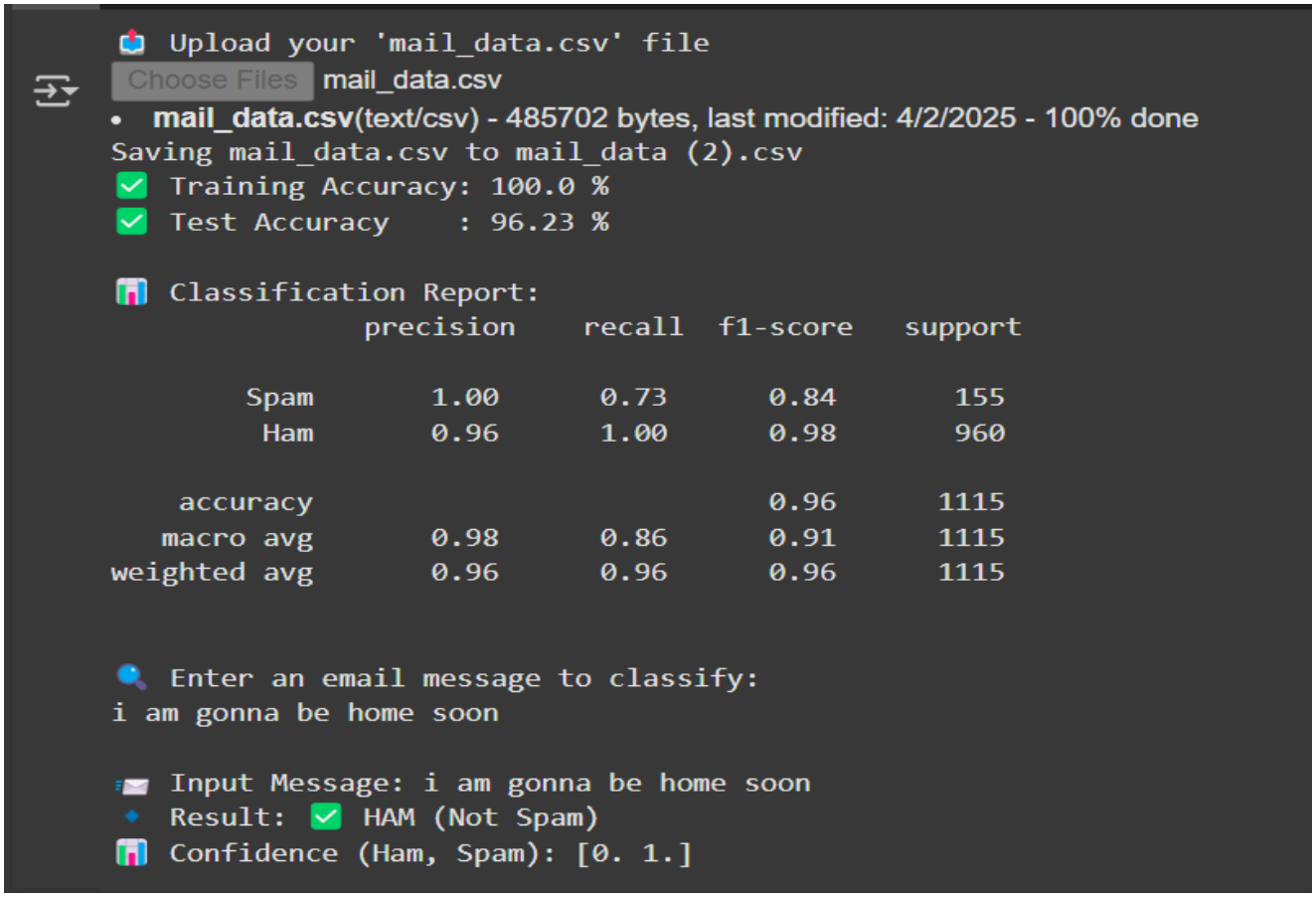
- Wrap the model into a web interface using:
  - **Flask** or **Streamlit**
  - Allow users to input a message and get real-time classification

Feature Importance

It highlights which words or phrases (features) most influence the model's prediction of spam vs. ham. In this project, the Random Forest classifier ranks terms like “free,” “win,” or “click” as highly important, as they frequently appear in spam messages. This helps improve model transparency and performance.

Output Screenshot :





✞ Conclusion :

The Spam Mail Prediction project successfully demonstrates how machine learning can be effectively applied to classify and filter unwanted email messages. By using natural language processing (NLP) techniques for text cleaning and feature extraction, combined with a robust classification algorithm like Random Forest, the system is capable of accurately distinguishing between spam and legitimate (ham) messages.

Through the use of TF-IDF vectorization and a labeled dataset, the model achieves high accuracy and reliability. The inclusion of bigrams in the vectorization step improves the model's understanding of context and word patterns commonly found in spam messages. Evaluation metrics confirm that the model performs well on unseen data, making it suitable for real-world applications.

This project not only highlights the power of machine learning in automating email filtering but also lays the foundation for further improvements, such as real-time spam detection, deployment as a web app, or integration with email clients.

Overall, the Spam Mail Prediction system proves to be a practical and scalable solution for enhanceing email security and improving user experience.

Learning Outcomes

1. **Understanding of Machine Learning Concepts**
    - Gained practical knowledge of supervised learning and classification problems.
    - Learned how to train, test, and evaluate machine learning models using real-world data.
2. **Hands-on Experience with NLP (Natural Language Processing)**
    - Learned techniques to preprocess text data (cleaning, tokenization, removing stopwords, etc.).
    - Understood how to convert textual data into numerical form using **TF-IDF vectorization**.
3. **Application of Classification Algorithms**
    - Implemented and understood the working of the **Random Forest Classifier**.
    - Compared predictions and accuracy using evaluation metrics like **precision**, **recall**, **F1-score**, and **accuracy**.
4. **Data Handling and Preparation Skills**
    - Gained experience in cleaning and preparing datasets using **pandas** and **numpy**.
    - Performed **label encoding**, handled null values, and split datasets for training/testing.
5. **Model Evaluation and Performance Tuning**
    - Learned how to interpret a classification report and adjust model parameters.
    - Explored n-grams and stop word removal to improve model performance.
6. **Interactive Model Testing**
    - Developed a custom message classification feature for real-time prediction and confidence scoring.
7. **Foundation for Web Deployment** *(Optional)*
    - Prepared for deploying ML models using **Flask** or **Streamlit** to make the system accessible via a user-friendly web interface.