Task 04

Email Spam Detection using Machine Learning

Introduction

Spam or junk emails are unwanted messages sent to large groups of users, often containing advertisements, scams, or phishing links.
In this project, a machine learning-based spam detection system was developed to classify emails into **“Spam”** and **“Ham”** (Not Spam).
This task was part of the **Data Science Internship at Oasis Infobyte** under AICTE.

The project combines text preprocessing, natural language processing (NLP), and supervised machine learning techniques to automatically detect spam emails.

Objective

The goal of this task is to build a predictive model that can automatically identify whether an incoming email is spam or not.
The system aims to:

- Preprocess email data.
- Extract useful features from text messages.
- Train a machine learning model for classification.
- Evaluate and visualize model performance.

This project demonstrates how data science can be used to improve communication safety by filtering unwanted messages.

Dataset Description

The dataset used for this project is **spam.csv**, which contains real email and SMS messages labeled as spam or ham.
It includes two main columns:

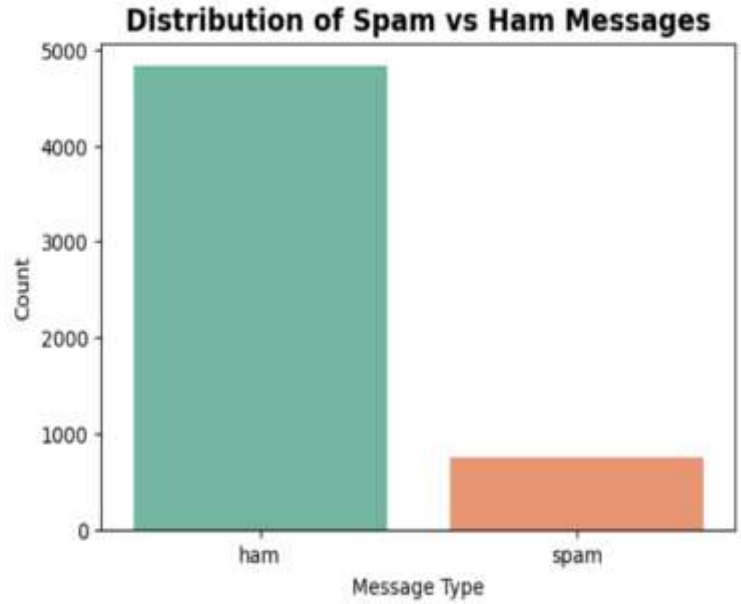
- **v1** → Label (`spam` or `ham`)
- **v2** → Email/SMS message content

After cleaning and renaming:

- **label** → Message type
- **message** → Message text

A new column **label_num** was also added to convert text labels into numeric values:

- `ham = 0`
- `spam = 1`



This dataset is widely used in spam classification research and provides a good mix of spam and normal messages.

Steps Performed

1. Imported required Python libraries (`pandas`, `seaborn`, `matplotlib`, `sklearn`).
2. Loaded and cleaned the dataset, renaming columns for clarity.
3. Converted labels (`ham`, `spam`) into numerical values (0 and 1).
4. Visualized the distribution of spam vs ham messages using a count plot.
5. Split the dataset into **training (80%)** and **testing (20%)** data.
6. Converted message text into numerical format using **CountVectorizer**, which removes common stop words and converts text into word count vectors.
7. Trained a **Multinomial Naive Bayes** classifier on the training data.
8. Predicted message classes on the test data.
9. Evaluated the model using:
   - Accuracy score
   - Classification report (Precision, Recall, F1-score)
   - Confusion matrix visualization
10. Tested the model on new, custom email messages to verify its performance.

Model Used: Multinomial Naive Bayes

The **Multinomial Naive Bayes (MNB)** algorithm is a probabilistic machine learning model that works well for text data.

It calculates the likelihood that a message belongs to a particular class (spam or ham) based on the frequency of each word.

Why Naive Bayes?

- Fast and efficient for text-based problems.
- Performs well even with large vocabularies.
- Works effectively with word count features generated from `CountVectorizer`.
- Requires less training data compared to other models.

How It Works (Example)

If a message contains words like *"win", "free", "click"*, the model assigns it a higher probability of being spam.
If it contains words like *"meeting", "project", "report"*, it's more likely to be ham.

Mathematically, it uses **Bayes' Theorem** to calculate:

$$P(Spam|Words) = P(Words|Spam) \times P(Spam) / P(Words)$$

This makes it highly effective for identifying messages with suspicious word patterns.

Example Workflow

Sample email:

"🎁 Congratulations! You won a free iPhone. Click here to claim now!"

1. The email text is converted into numeric features like:
   `{win: 1, free: 1, click: 1, claim: 1}`
2. The model checks how often these words appear in spam messages.
3. Since words like *"free"* and *"click"* are common in spam, the model predicts:
   Result → Spam (1)

Another example:

"Please find the project report attached for today's meeting."
Here, no spam words are found.
Result → Ham (0)

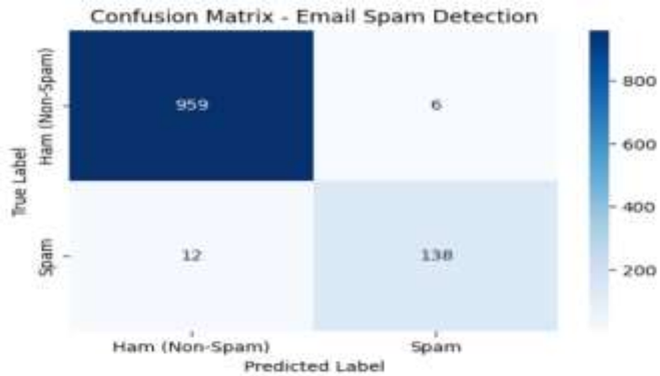
Evaluation and Results

The model was evaluated using multiple performance metrics:

Metric	Description	Result (approx.)
Accuracy	Overall correctness of predictions	97–98%
Precision	How many predicted spams were actually spam	0.96
Recall	How many real spams were correctly detected	0.95
F1-score	Balance between precision and recall	0.96

Confusion Matrix Visualization

A confusion matrix heatmap was plotted to display the number of correct and incorrect classifications.
Most predictions were accurate, with minimal false positives or false negatives.



Key Insights

- Spam messages commonly contain promotional or urgent words like *"free", "win", "offer", "click"*.
- Ham (non-spam) messages are usually shorter and contain professional or conversational language.
- Despite having more ham messages in the dataset, the model handled class imbalance efficiently.
- `CountVectorizer` combined with `MultinomialNB` proved to be a powerful and fast combination for spam filtering.
- The model's real-world predictions on new emails were consistent and accurate.

Conclusion

This project successfully implemented a spam detection model using **Multinomial Naive Bayes** and **Natural Language Processing** techniques.
By analyzing the text patterns in emails, the system achieved excellent accuracy in classifying messages as spam or ham.

This approach can be integrated into **real-world email filtering systems** (like Gmail or Outlook) to automatically detect and filter spam, enhancing user safety and communication efficiency.

Tools & Libraries Used

- Python

- Pandas, NumPy
- Seaborn, Matplotlib
- Scikit-learn (`CountVectorizer`, `MultinomialNB`, `train_test_split`, `metrics`)