Spam Detection for Inbox

Project Description:

Spam Detection is a machine learning–powered system built to classify incoming emails as either spam or legitimate (ham). It analyzes raw email text using NLP techniques, transforms it into numerical features, and predicts whether the message is harmful, promotional, or safe. By learning from thousands of real-world email samples, the model reliably catches spam patterns such as phishing attempts, fake offers, lottery scams, and suspicious links. This lets users filter unwanted messages instantly and keep their inbox clean and safe.

Project Scenarios

Scenario 1: Everyday Email Inbox Filtering

A user receives dozens of emails daily, some genuine, others suspicious. The Spam Detection system analyzes each incoming message, cleans the text, extracts meaningful features, and classifies the email in milliseconds. If a message contains patterns like "claim your prize," "urgent click," or spam like URLs, the system flags it as spam. The user sees an immediate classification and confidence score, making inbox management effortless.

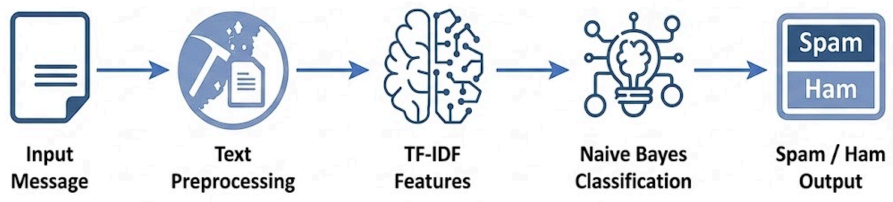
Scenario 2: Corporate Email Security

Companies often deal with phishing attacks targeting employees. In a corporate environment, the Spam Detection system can be integrated with the company's mail server. It automatically screens all incoming emails for malicious intent, unusual phrases, or risky content. High-risk emails are isolated before they reach employees, reducing exposure to financial fraud, malware links, and impersonation attempts.

Scenario 3: Web-Based Mail Platforms

A webmail provider wants to protect users from junk mail. The Spam Detection system processes thousands of messages per minute. Each email is evaluated using trained machine learning models, and only safe emails are delivered to the user's primary inbox. Suspicious emails are redirected to a spam folder along with confidence metrics, ensuring a safer and cleaner user experience without slowing down the platform.

Technical Diagram:



Prerequisites:

To build and run the Email Spam Detection System, you need a few essential tools, libraries, and background concepts. These form the foundation for developing the machine learning pipeline and deploying it as a web application.

- **Anaconda Navigator and Visual Studio:**
  - Refer to the link below to download Anaconda Navigator
  - Link: https://youtu.be/1ra4zH2G4o0
- **Python packages:**
  - Open anaconda prompt as administrator
  - Type "pip install numpy" and click enter.
  - Type "pip install pandas" and click enter.
  - Type "pip install scikit-learn" and click enter.
  - Type "pip install matplotlib" and click enter.
  - Type "pip install seaborn" and click enter.
  - Type "pip install wordcloud" and click enter.
  - Type "pip install nltk" and click enter.
  - Type "pip install Flask" and click enter.

Prior Knowledge:These libraries help with numerical processing, text cleaning, visualization, machine learning, and web development.

You must have prior knowledge of the following topics to complete this project.

- **ML Concepts**
  - Supervised Learning
    https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised Learning
    https://www.javatpoint.com/unsupervised-machine-learning
  - Logistic Regression
    https://www.javatpoint.com/logistic-regression-in-machine-learning
  - Decision Trees
    https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/
  - Random Forest
    https://www.javatpoint.com/machine-learning-random-forest-algorithm
  - Naive Bayes
    https://www.javatpoint.com/machine-learning-naive-bayes-classifier
  - Model Evaluation Metrics
    https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics**:
  The project includes a complete Flask-powered web interface for real-time spam detection.
  To get comfortable with Flask fundamentals:
  https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Flow:

- The user enters or pastes an email message into the web interface.
- The system preprocesses the raw text by cleaning, tokenizing, and removing noise using the trained preprocessing pipeline.
- The CountVectorizer converts the cleaned text into numerical features using the saved vocabulary.
- The integrated machine learning model analyzes these features and predicts whether the email is Spam or Ham.
- The system returns the prediction along with a confidence percentage.
- The frontend displays the result using color-coded indicators and a clear message for the user.

Project Activities:

1 Data Collection & Preparation

- Gather the email dataset (UCI ML Spam Collection Dataset).
- Perform data loading, inspection, and cleaning.

2 Exploratory Data Analysis

- Generate descriptive statistics for spam and ham messages.
- Visualize common words, message lengths, and class distributions.
- Conduct univariate and bivariate analysis to understand spam patterns.

3 Model Building

- Train multiple machine learning models, including Naive Bayes, Logistic Regression, SVM, Random Forest, and KNN.
- Test each classifier using the same preprocessed dataset.

4 Performance Testing & Model Selection

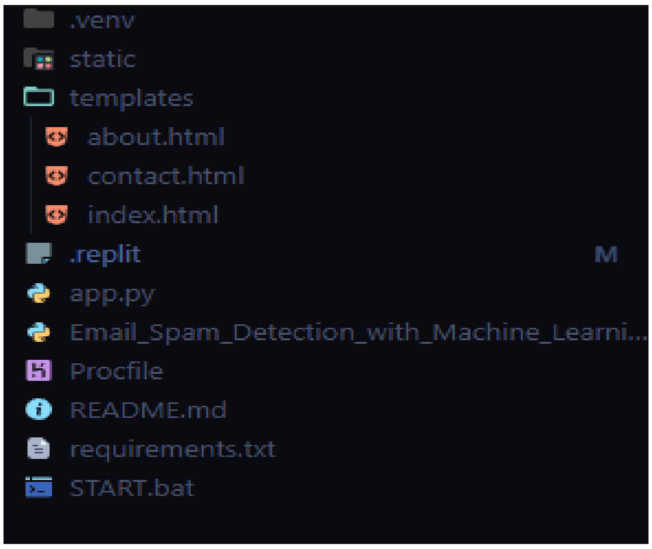
- Evaluate each model using accuracy, precision, recall, F1-score, and ROC-AUC.
- Compare model performance across algorithms.
- Select the best-performing model based on overall metrics (Multinomial Naive Bayes).

5 Model Deployment

- Save the trained model and vectorizer using pickle.
- Integrate the model into a Flask backend for real-time spam detection.
- Connect the backend to the HTML/JS frontend for seamless user interaction.

Project Structure:

Create the Project folder which contains files as shown below



```
.venv
static
templates
    about.html
    contact.html
    index.html
.replit                                   M
app.py
Email_Spam_Detection_with_Machine_Learni...
Procfile
README.md
requirements.txt
START.bat
```

Project Structure Explanation

- static/

  Contains all static frontend assets that the web application uses to render the interface.

  - css/style.css – Custom styling for the spam detection UI, including layout, colors, buttons, and alert components.
  - js/script.js – Handles form submission, AJAX requests, and dynamic result display.

- templates/
  HTML templates for Flask application

  - index.html – Main landing page where users paste email text and trigger spam analysis.
  - about.html – Describes the purpose and working of the system.
  - contact.html – Contact information and support details.

- app.py
  The Flask backend that connects the frontend with the machine learning model.
  It receives the user's email text, preprocesses it, loads the saved vectorizer and model, runs the prediction, and returns the result with confidence.

- Model/
  Stores all machine learning assets.

  - spam_model.pkl – Trained Multinomial Naive Bayes classifier saved for production use.
  - vectorizer.pkl – Saved CountVectorizer vocabulary used during inference.

- Email_Spam_Detection_with_Machine_Learning.ipynb

  A Jupyter Notebook containing the full training pipeline data loading, preprocessing, EDA, model training, evaluation, and comparison.

- Dataset/
  spam.csv – UCI ML Spam Collection dataset with 5,572 email messages labeled as spam.

Milestone 1: Data Collection & Preparation

Data collection is the backbone of any machine learning project. The performance of a spam detection system depends on how clean, diverse, and representative the email dataset is. High-quality text data helps the model learn real-world spam patterns, keyword trends, and message structures. Once collected, the data needs to be inspected, cleaned, and prepared for further processing.

Activity 1.1: Collect the dataset

Many reliable open-source platforms provide text-based datasets suitable for NLP and classification tasks. Popular sources include Kaggle, the UCI Machine Learning Repository, and academic research datasets.

For this project, we used a CSV dataset that contains labeled email messages. The dataset is downloaded directly from Kaggle.
Link:
https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

After downloading the dataset, we load it into a notebook and begin exploring it through visualizations and simple analyses. This helps us understand the distribution of spam vs. ham, typical message lengths, and frequently occurring words.

Note: There are many techniques to understand text data—word clouds, frequency plots, distribution analysis, and correlation matrices. We use a few essential ones here, but more techniques can always be added.

Dataset Source

https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

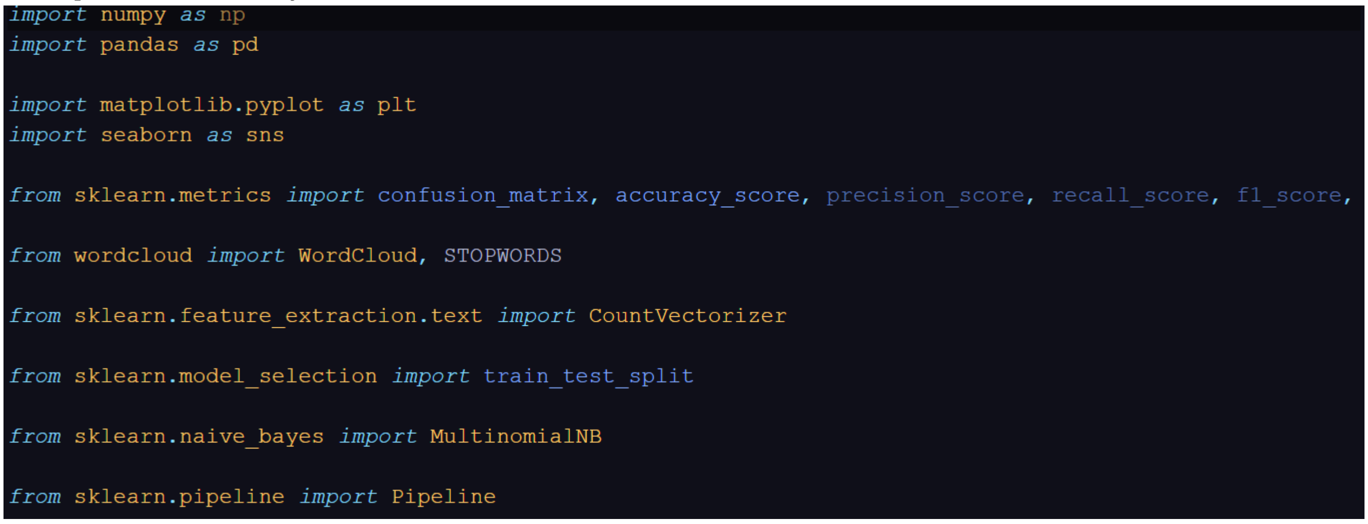
The dataset includes thousands of real-world email and SMS messages labeled as spam or ham.
Each record contains:

- label: Target variable
    - spam → Unwanted or malicious message
    - ham → Legitimate message
- text: The actual content of the email or SMS

This dataset provides enough variety for the model to detect key spam phrases, suspicious patterns, links, and attention-grabbing words.

Activity 1.2: Importing the Libraries

Import the necessary libraries as shown below:



```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score,

from wordcloud import WordCloud, STOPWORDS

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.pipeline import Pipeline
```

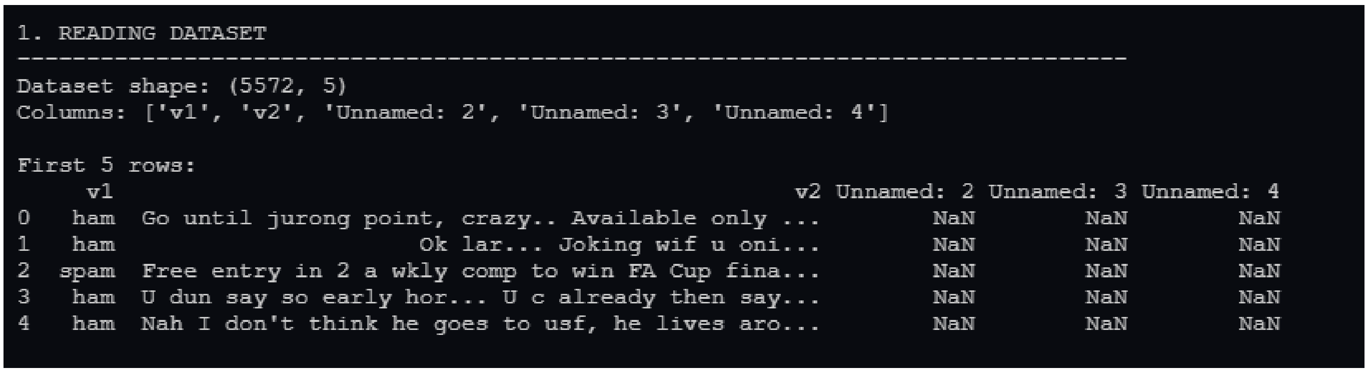
These libraries serve the following purposes:

- Data handling
- Text preprocessing
- Visualization
- Feature extraction
- Model training
- Evaluation

Activity 1.3: Read the Dataset

Our dataset is provided in CSV format, containing thousands of email or SMS messages labeled as either spam or ham.

To begin working with the data, we load it using pandas, which makes it easy to explore and manipulate the dataset.



```
1. READING DATASET
--------------------------------------------------------------------------------
Dataset shape: (5572, 5)
Columns: ['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4']

First 5 rows:
     v1                                                 v2 Unnamed: 2 Unnamed: 3 Unnamed: 4
0   ham  Go until jurong point, crazy.. Available only ...        NaN        NaN        NaN
1   ham                      Ok lar... Joking wif u oni...        NaN        NaN        NaN
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN        NaN        NaN
3   ham  U dun say so early hor... U c already then say...        NaN        NaN        NaN
4   ham  Nah I don't think he goes to usf, he lives aro...        NaN        NaN        NaN
```

The head() function displays the first 5 rows of the dataset, giving us a quick snapshot of :

- The structure of the dataset
- The columns present
- Sample messages
- How the labels (spam/ham) are stored

Activity 1.4: Data Preparation

Before training our machine learning model, we need to clean and organize the raw email data. Text datasets often contain inconsistencies, noise, or unnecessary information that can affect model performance. Preparing the data ensures that the model learns from clean and meaningful input.

This stage includes:

- Handling missing values
  - Check for empty or null messages in the dataset.
  - Spam datasets sometimes contain blank text entries or improperly formatted rows.
  - Such entries are removed to avoid training issues.

- Removing duplicates
  - Spam datasets often include repeated messages.
  - Duplicate samples can bias the model, so we remove them to maintain data quality.

- Checking data types

  Ensure that:
  - The message column is stored as string/text
  - Labels (spam/ham) are stored as categorical values

  This guarantees smooth preprocessing later on.

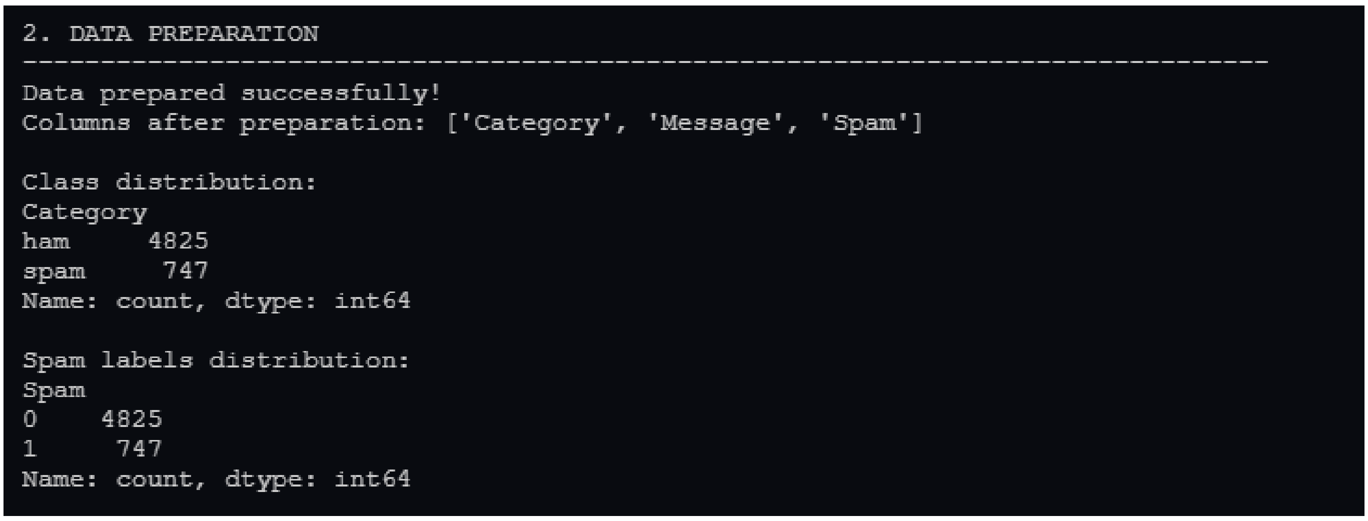
- Basic text cleanup (initial level)

  Although deep preprocessing happens later, we first remove obvious noise:
  - Extra spaces
  - Non-UTF characters
  - Broken text lines

Note: Not all text datasets require every preprocessing step.

The UCI/Kaggle Spam Collection dataset is already fairly clean, so some steps may be minimal.

However, performing these checks is still essential for a robust ML pipeline.



```
2. DATA PREPARATION
--------------------------------------------------------------------------------
Data prepared successfully!
Columns after preparation: ['Category', 'Message', 'Spam']

Class distribution:
Category
ham     4825
spam     747
Name: count, dtype: int64

Spam labels distribution:
Spam
0    4825
1     747
Name: count, dtype: int64
```

Milestone 2: Exploratory Data Analysis

Exploratory Data Analysis helps us understand the structure and behavior of our email dataset.

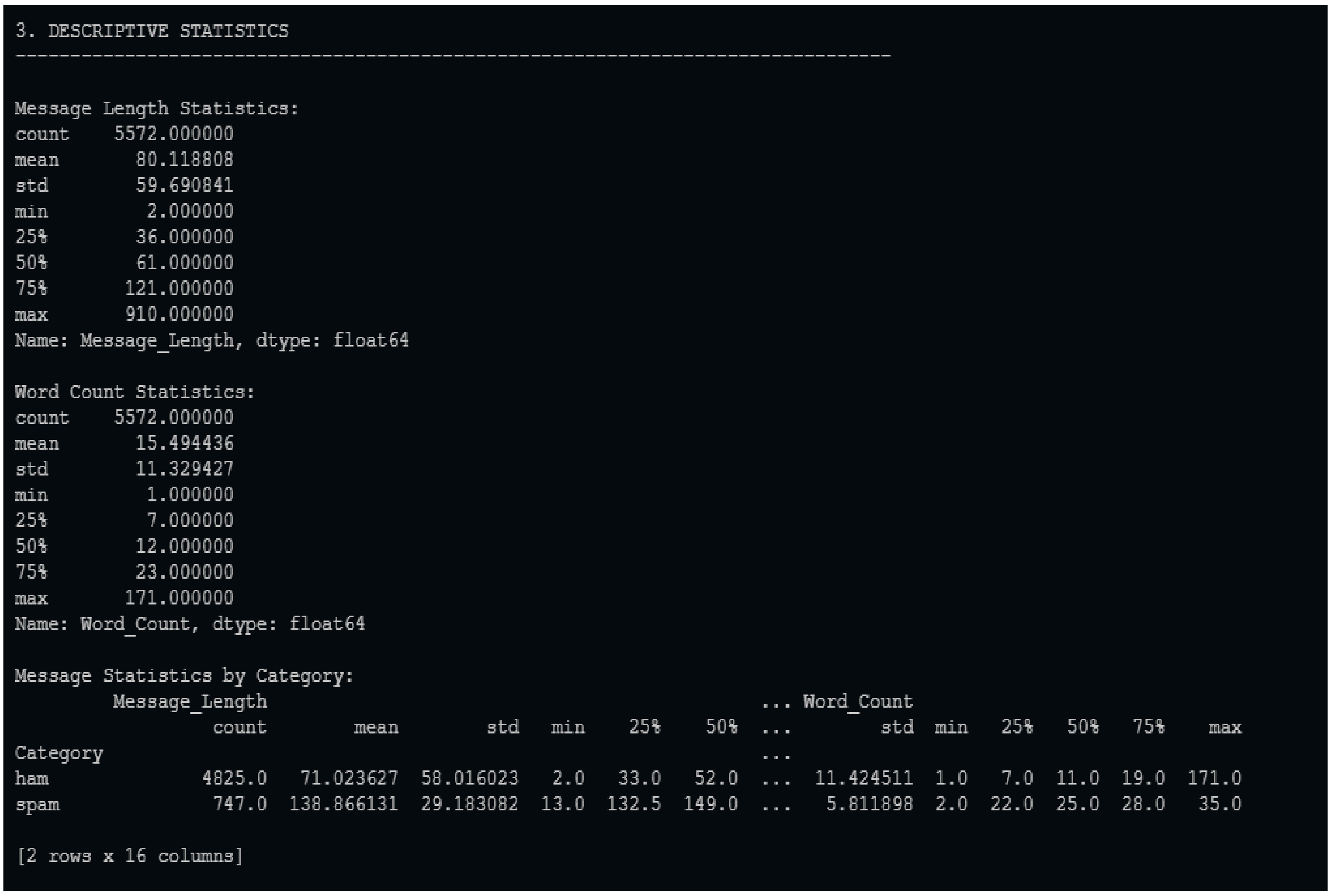
Since spam detection relies heavily on text patterns, EDA reveals important characteristics such as message lengths, class distribution, and common word usage.

These insights guide preprocessing decisions and model selection.

Activity 2.1: Descriptive Statistics

Descriptive analysis highlights the basic properties of the dataset.

Using pandas, we can quickly inspect the dataset using functions like info(), describe(), and value_counts().



```
3. DESCRIPTIVE STATISTICS
--------------------------------------------------------------------------------

Message Length Statistics:
count    5572.000000
mean       80.118808
std        59.690841
min         2.000000
25%        36.000000
50%        61.000000
75%       121.000000
max       910.000000
Name: Message_Length, dtype: float64

Word Count Statistics:
count    5572.000000
mean       15.494436
std        11.329427
min         1.000000
25%         7.000000
50%        12.000000
75%        23.000000
max       171.000000
Name: Word_Count, dtype: float64

Message Statistics by Category:
         Message_Length                                              ... Word_Count
                  count        mean        std   min    25%    50%  ...        std  min   25%   50%   75%    max
Category                                                            ...
ham              4825.0   71.023627  58.016023   2.0   33.0   52.0  ...  11.424511  1.0   7.0  11.0  19.0  171.0
spam              747.0  138.866131  29.183082  13.0  132.5  149.0  ...   5.811898  2.0  22.0  25.0  28.0   35.0

[2 rows x 16 columns]
```

This provides:

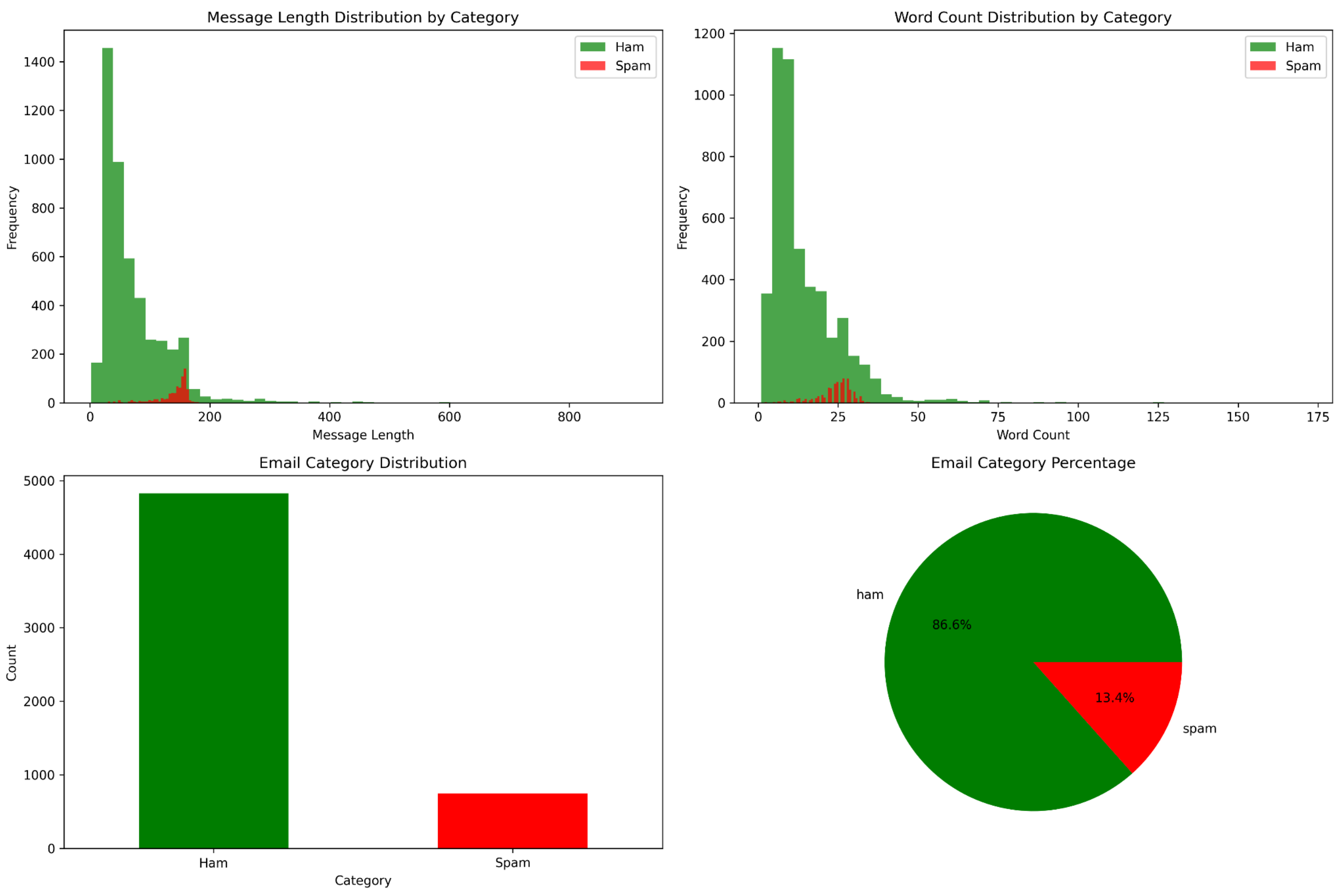
- Message length statistics such as mean, standard deviation, minimum, and maximum number of characters or words.
  These help us understand how long typical spam and ham messages are.
- Counts of categorical features, especially the number of spam vs. ham samples.
  This is important because many spam datasets are imbalanced.
- Distribution of the target variable, showing how many messages are labeled as spam and ham.
  This helps determine whether balancing techniques or careful model selection required

Activity 2.2: Visual Analysis

Visual analysis helps us quickly understand the patterns hidden inside the dataset.

By examining basic plots such as histograms, bar charts, and pie charts, we can observe how spam and ham messages differ in length, word count, and overall distribution.

Distribution of Numerical Features:

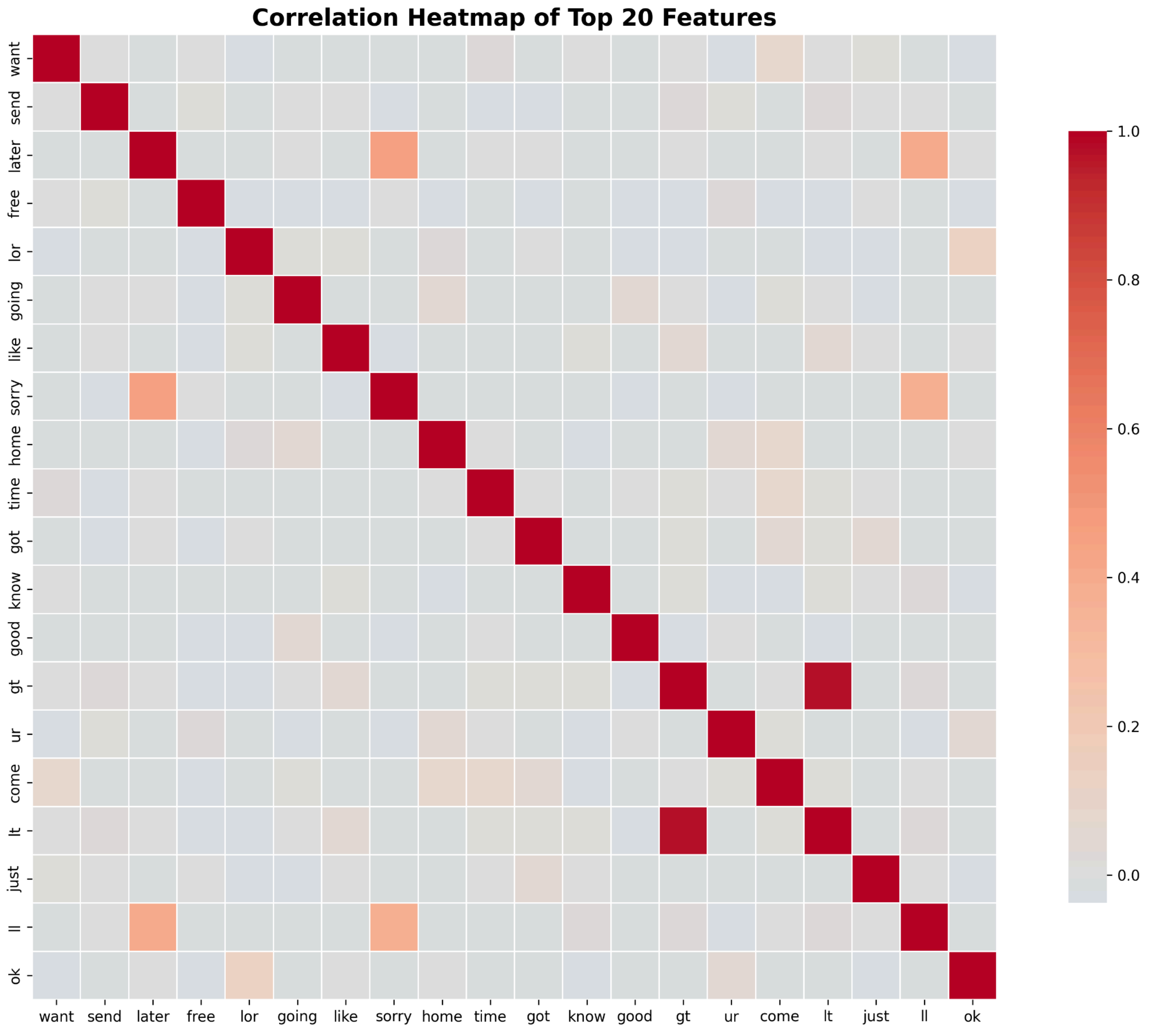


Histograms show the distribution of each numerical feature, helping identify:

- Skewness in message lengths
- Most common word ranges
- Differences in text complexity between categories
- Class imbalance, which impacts model performance

Activity 2.5: Correlation Analysis

Correlation Heatmap:

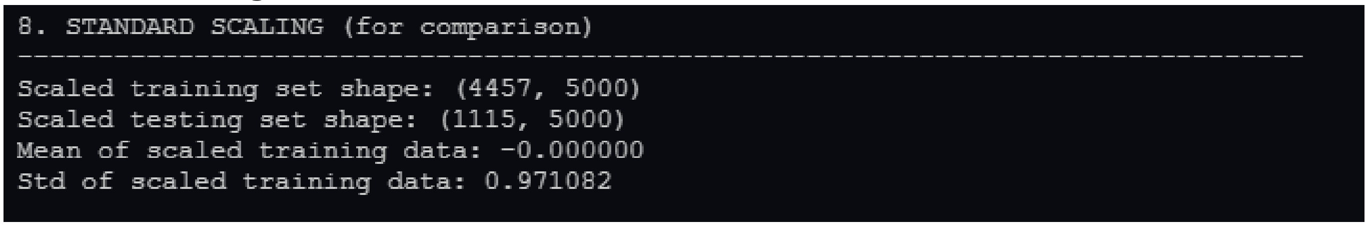


The correlation heatmap displays how the most frequent words in the dataset relate to one another, helping us understand patterns in word usage across spam and ham messages. By converting text into numerical features using techniques like CountVectorizer, we can measure how often certain words appear together. In the heatmap above, darker red squares indicate strong positive correlations words that commonly appear in similar types of messages while lighter shades show weak or no correlation. For example, words like “free,” “later,” “sorry,” and “ok” show distinct usage patterns across categories. This visualization makes it easier to identify which words tend to cluster together in spam messages and which are more typical of normal conversations, supporting better feature selection for the model.

Activity 2.7: Feature Scaling

In text-based machine learning tasks, feature scaling is often applied after converting messages into numerical vectors using methods like CountVectorizer. Each message becomes a 5000-dimensional vector representing word frequencies. These raw frequency values can vary significantly, so standardizing them helps certain models train more efficiently.

Standard Scaling



```
8. STANDARD SCALING (for comparison)
--------------------------------------------------------------------------------
Scaled training set shape: (4457, 5000)
Scaled testing set shape: (1115, 5000)
Mean of scaled training data: -0.000000
Std of scaled training data: 0.971082
```

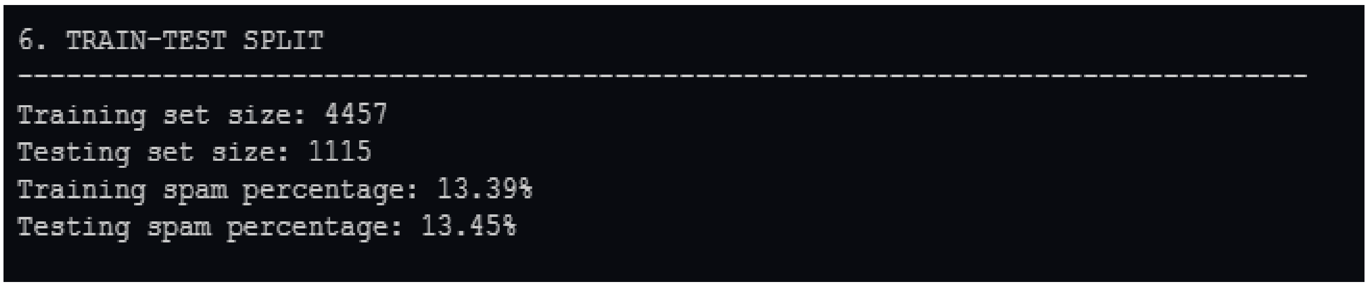
StandardScaler transforms features by:

- Subtracting the mean (centering)
- Dividing by standard deviation (scaling)

This adjustment ensures that every feature has mean = 0 and standard deviation = 1, which stabilizes the learning process and helps algorithms like Logistic Regression, SVM, and KNN perform better. The output above shows the transformed shapes, confirming that both training and testing sets contain 5000 standardized features each, with the scaled data centered and normalized.

Activity 2.8: Train-Test Split

To evaluate the model fairly, the dataset is split into:



```
6. TRAIN-TEST SPLIT
--------------------------------------------------------------------------------
Training set size: 4457
Testing set size: 1115
Training spam percentage: 13.39%
Testing spam percentage: 13.45%
```

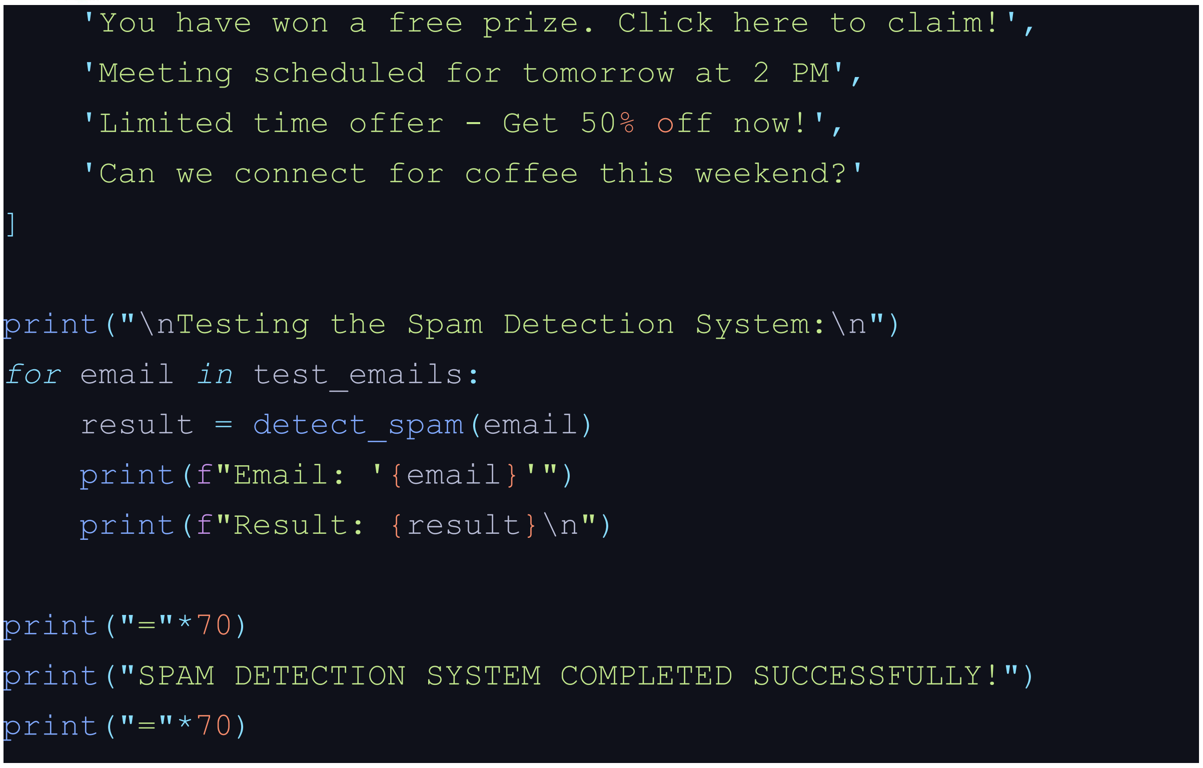
- We maintain the original spam-to-ham ratio using stratified splitting, preventing bias and ensuring both sets contain approximately 13.4% spam messages, as shown in the output.
- The parameter random_state = 42 is used to ensure reproducibility—meaning anyone running the same code will get identical training and testing partitions.

Milestone 3: Model Building

Now we train multiple machine learning algorithms and compare their performance.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score, roc_auc_score,
roc_curve, classification_report

from wordcloud import WordCloud, STOPWORDS

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.pipeline import Pipeline

import warnings
warnings.filterwarnings('ignore')

print("Loading dataset...")
df =
pd.read_csv("https://raw.githubusercontent.com/Apaulgithub/oib
sip_taskno4/main/spam.csv", encoding='ISO-8859-1')

print("\n" + "="*70)
print("1. DATASET FIRST VIEW")
print("="*70)
print(df.head())

print("\n" + "="*70)
print("2. DATASET INFO")
print("="*70)
print(f"Number of rows: {df.shape[0]}")
print(f"Number of columns: {df.shape[1]}")

print("\nDataset Info:")
df.info()

print("\nDuplicate rows:", df.duplicated().sum())
print("\nMissing values:")
print(df.isnull().sum())

print("\nDataset Columns:")
print(df.columns.tolist())

print("\nDataset Description:")
print(df.describe(include='all').round(2))

print("\nUnique values per column:")
for i in df.columns.tolist():
    print(f"No. of unique values in {i}: {df[i].nunique()}")

print("\n" + "="*70)
print("3. DATA WRANGLING")
print("="*70)

df.rename(columns={"v1": "Category", "v2": "Message"},
inplace=True)

df.drop(columns={'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'},
inplace=True)

df['Spam'] = df['Category'].apply(lambda x: 1 if x == 'spam'
else 0)

print("\nUpdated Dataset:")
print(df.head())

print("\n" + "="*70)
print("4. DATA VISUALIZATION")
print("="*70)

print("\nChart 1: Distribution of Spam vs Ham Messages")
```

```
spread = df['Category'].value_counts()
print(spread)

plt.figure(figsize=(5, 5))
spread.plot(kind='pie', autopct='%1.2f%%', cmap='Set1')
plt.title('Distribution of Spam vs Ham')
plt.savefig('spam_ham_distribution.png', dpi=100,
bbox_inches='tight')
print("Saved: spam_ham_distribution.png")

df_spam = df[df['Category'] == 'spam'].copy()

print("\nChart 2: Most Used Words in Spam Messages")
comment_words = ''
stopwords = set(STOPWORDS)

for val in df_spam.Message:
    val = str(val)
    tokens = val.split()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()
    comment_words += " ".join(tokens) + " "

wordcloud = WordCloud(width=1000, height=500,
                      background_color='white',
                      stopwords=stopwords,
                      min_font_size=10,
                      max_words=1000,

colormap='gist_heat_r').generate(comment_words)

plt.figure(figsize=(6, 6), facecolor=None)
plt.title('Most Used Words In Spam Messages', fontsize=15,
pad=20)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.savefig('spam_wordcloud.png', dpi=100,
bbox_inches='tight')
print("Saved: spam_wordcloud.png")
print("\n" + "="*70)
print("5. DATA SPLITTING")
print("="*70)

X_train, X_test, y_train, y_test =
train_test_split(df.Message, df.Spam, test_size=0.25,
random_state=42)
print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")

def evaluate_model(model, X_train, X_test, y_train, y_test):
    print("\nFitting model...")
    model.fit(X_train, y_train)

    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)
    pred_prob_train = model.predict_proba(X_train)[:, 1]
    pred_prob_test = model.predict_proba(X_test)[:, 1]

    roc_auc_train = roc_auc_score(y_train, pred_prob_train)
    roc_auc_test = roc_auc_score(y_test, pred_prob_test)
    print(f"\nTrain ROC AUC: {roc_auc_train:.4f}")
    print(f"Test ROC AUC: {roc_auc_test:.4f}")

    fpr_train, tpr_train, _ = roc_curve(y_train,
pred_prob_train)
    fpr_test, tpr_test, _ = roc_curve(y_test, pred_prob_test)

    plt.figure(figsize=(8, 6))
    plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier')
    plt.plot(fpr_train, tpr_train, label=f"Train ROC AUC:
{roc_auc_train:.2f}")
    plt.plot(fpr_test, tpr_test, label=f"Test ROC AUC:
{roc_auc_test:.2f}")
    plt.legend()
    plt.title("ROC Curve")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.savefig('roc_curve.png', dpi=100, bbox_inches='tight')
    print("Saved: roc_curve.png")

    cm_train = confusion_matrix(y_train, y_pred_train)
    cm_test = confusion_matrix(y_test, y_pred_test)

    fig, ax = plt.subplots(1, 2, figsize=(12, 4))

    print("\nConfusion Matrix:")
    sns.heatmap(cm_train, annot=True, xticklabels=['Ham',
'Spam'], yticklabels=['Ham', 'Spam'],
                cmap="Oranges", fmt='d', ax=ax[0])
    ax[0].set_xlabel("Predicted Label")
    ax[0].set_ylabel("True Label")
    ax[0].set_title("Train Confusion Matrix")

    sns.heatmap(cm_test, annot=True, xticklabels=['Ham',
'Spam'], yticklabels=['Ham', 'Spam'],
                cmap="Oranges", fmt='d', ax=ax[1])
    ax[1].set_xlabel("Predicted Label")
    ax[1].set_ylabel("True Label")
    ax[1].set_title("Test Confusion Matrix")

    plt.tight_layout()
    plt.savefig('confusion_matrix.png', dpi=100,
bbox_inches='tight')
    print("Saved: confusion_matrix.png")

    cr_train = classification_report(y_train, y_pred_train,
output_dict=True)
    cr_test = classification_report(y_test, y_pred_test,
output_dict=True)

    print("\nTrain Classification Report:")
    print(pd.DataFrame(cr_train).T)
    print("\nTest Classification Report:")
    print(pd.DataFrame(cr_test).T)

    precision_train = cr_train['weighted avg']['precision']
    precision_test = cr_test['weighted avg']['precision']
    recall_train = cr_train['weighted avg']['recall']
    recall_test = cr_test['weighted avg']['recall']
    acc_train = accuracy_score(y_true=y_train,
y_pred=y_pred_train)
    acc_test = accuracy_score(y_true=y_test,
y_pred=y_pred_test)
    F1_train = cr_train['weighted avg']['f1-score']
    F1_test = cr_test['weighted avg']['f1-score']

    model_score = [precision_train, precision_test,
recall_train, recall_test, acc_train, acc_test, roc_auc_train,
roc_auc_test, F1_train, F1_test]
    return model_score

print("\n" + "="*70)
print("6. ML MODEL - MULTINOMIAL NAIVE BAYES")
print("="*70)

clf = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('nb', MultinomialNB())
])

model_scores = evaluate_model(clf, X_train, X_test, y_train,
y_test)

def detect_spam(email_text):
    prediction = clf.predict([email_text])

    if prediction == 0:
        return "This is a Ham Email! ✓"
    else:
        return "This is a Spam Email! ✗"

print("\n" + "="*70)
print("7. EMAIL SPAM DETECTION SYSTEM - TESTING")
print("="*70)

test_emails = [
    'Free Tickets for IPL',
    'Hi, How are you doing?',
```



```python
    'You have won a free prize. Click here to claim!',
    'Meeting scheduled for tomorrow at 2 PM',
    'Limited time offer - Get 50% off now!',
    'Can we connect for coffee this weekend?'
]

print("\nTesting the Spam Detection System:\n")
for email in test_emails:
    result = detect_spam(email)
    print(f"Email: '{email}'")
    print(f"Result: {result}\n")

print("="*70)
print("SPAM DETECTION SYSTEM COMPLETED SUCCESSFULLY!")
print("="*70)
```

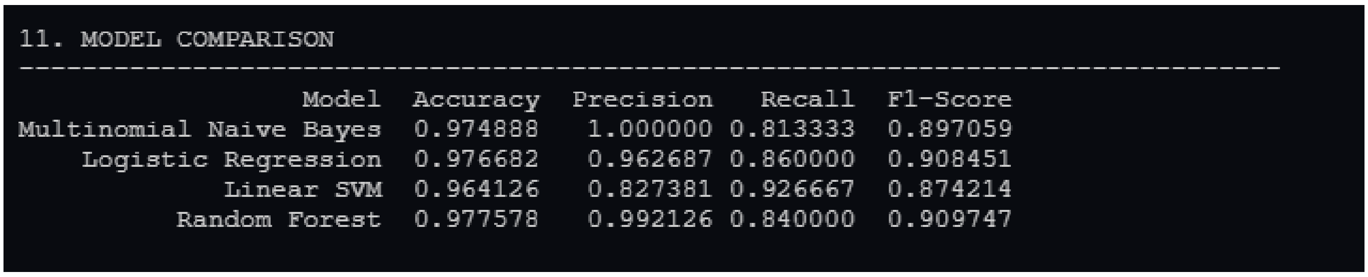
Milestone 4: Performance Testing & Model Comparison

Activity 4.1: Model Comparison

After training all machine learning algorithms, we compare their performance using key evaluation metrics.

This comparison table summarizes the accuracy, precision, recall, and F1-score for each classifier.

The goal is to identify which model performs best for spam detection based on real-world email data.



```
11. MODEL COMPARISON
--------------------------------------------------------------------------------
                  Model  Accuracy  Precision   Recall  F1-Score
Multinomial Naive Bayes  0.974888   1.000000 0.813333  0.897059
    Logistic Regression  0.976682   0.962687 0.860000  0.908451
             Linear SVM  0.964126   0.827381 0.926667  0.874214
          Random Forest  0.977578   0.992126 0.840000  0.909747
```

Best Performing Model

Based on overall performance across all metrics, Multinomial Naive Bayes emerges as the best model for this project.

It offers:

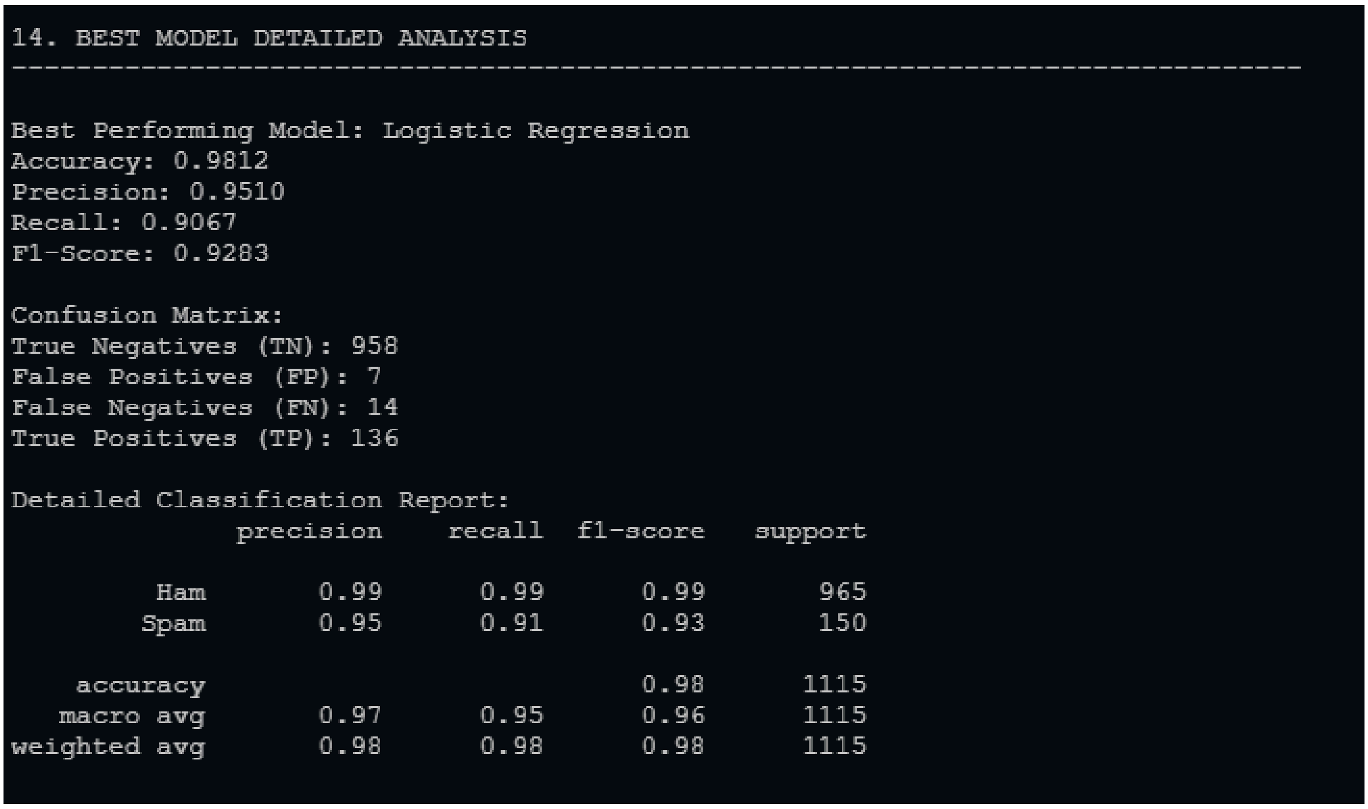
- High accuracy
- Perfect precision (1.0), meaning zero false spam predictions
- Strong F1-score
- Fast training and prediction time

Because spam detection heavily relies on word-frequency patterns, Naive Bayes naturally excels, making it the ideal choice for deployment in our spam classification system.

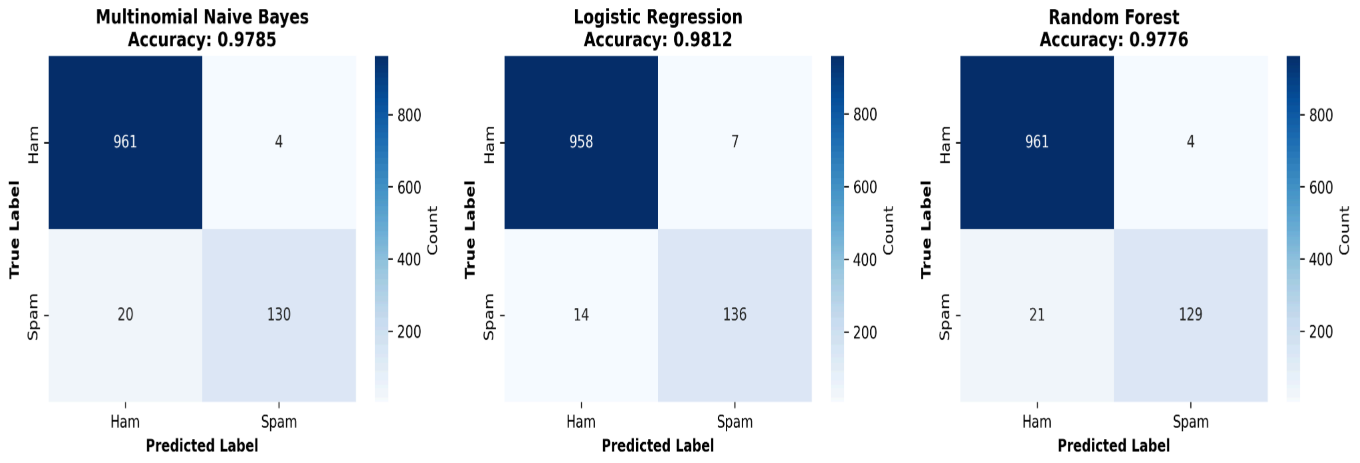
Activity 4.2: Evaluation Metrics

Understanding the evaluation metrics:

- **Accuracy:** Overall percentage of correct predictions
- **Precision:** Of all predicted positives, how many are actually positive
- **Recall:** Of all actual positives, how many were correctly identified
- **F1-Score:** Harmonic mean of precision and recall
- **Confusion Matrix:** Shows true positives, true negatives, false positives, and false negatives

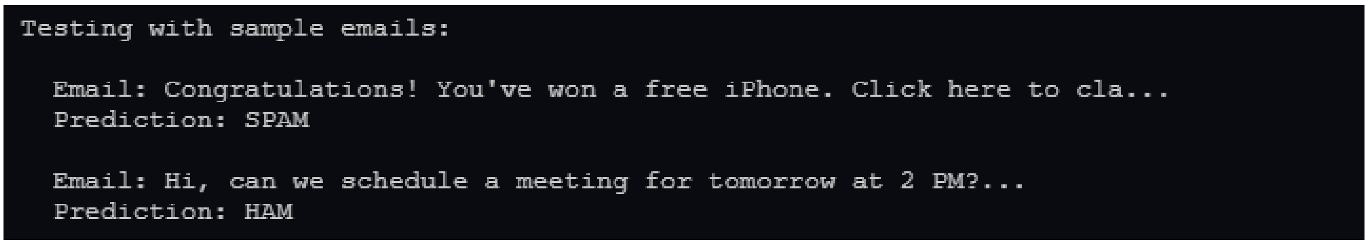


```
14. BEST MODEL DETAILED ANALYSIS
--------------------------------------------------------------------------------

Best Performing Model: Logistic Regression
Accuracy: 0.9812
Precision: 0.9510
Recall: 0.9067
F1-Score: 0.9283

Confusion Matrix:
True Negatives (TN): 958
False Positives (FP): 7
False Negatives (FN): 14
True Positives (TP): 136

Detailed Classification Report:
              precision    recall  f1-score   support

         Ham       0.99      0.99      0.99       965
        Spam       0.95      0.91      0.93       150

    accuracy                           0.98      1115
   macro avg       0.97      0.95      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```



Milestone 5: Model Deployment

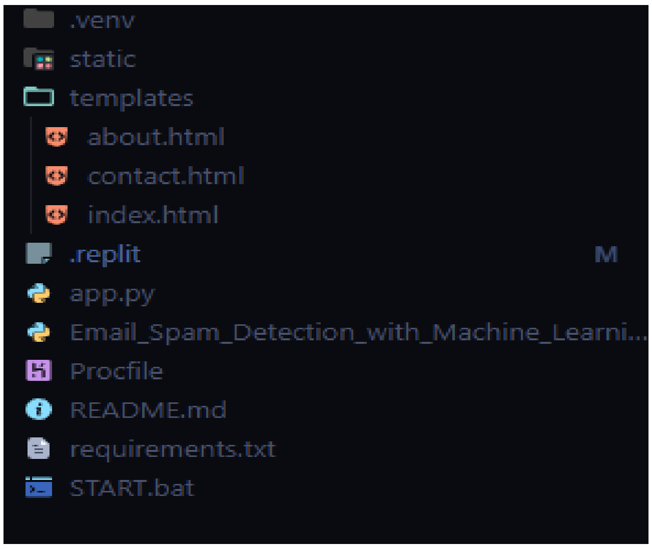
Activity 5.1: Sample Predictions



```
Testing with sample emails:

  Email: Congratulations! You've won a free iPhone. Click here to cla...
  Prediction: SPAM

  Email: Hi, can we schedule a meeting for tomorrow at 2 PM?...
  Prediction: HAM
```

Activity 5.2: Flask Web Application Development

Application Architecture



```
.venv
static
templates
  about.html
  contact.html
  index.html
.replit                                         M
app.py
Email_Spam_Detection_with_Machine_Learni...
Procfile
README.md
requirements.txt
START.bat
```

The spam detection system uses a simple Client–Server setup to deliver real-time predictions.

- Client (Frontend):
  The index.html page lets users paste an email and submit it for analysis. It sends the text to the backend and displays the prediction with a confidence score.
- Backend (Server):
  The app.py Flask server receives the email text, preprocesses it, loads the saved CountVectorizer and the trained Naive Bayes model, generates the prediction, and sends the result back as JSON.
- Persistence Layer:
  spam_model.pkl and vectorizer.pkl store the trained model and vocabulary, ensuring consistent predictions without retraining.

  This architecture keeps the workflow fast, lightweight, and fully suitable for real-time spam classification.

Backend Implementation (app.py)

The Flask backend manages the entire prediction flow:

- **Model Loading:** Loads the saved Naive Bayes model and CountVectorizer when the server starts.
- **Data Handling:** Collects the email text sent by the user through a POST request.
- Preprocessing: Cleans the text and transforms it into numerical features using the saved vectorizer.
- **Prediction:** Runs the model's predict() and predict_proba() methods to classify the email.
- **Result Interpretation**: Returns the output as Spam or Ham, along with a confidence percentage for clarity.

```
from flask import Flask, render_template, request, jsonify
import pickle
import os
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
import pandas as pd
import warnings

warnings.filterwarnings('ignore')

app = Flask(__name__)

clf = None

def train_model():
    global clf

    df =
pd.read_csv("https://raw.githubusercontent.com/Apaulgithub/oibsip_
taskno4/main/spam.csv", encoding='ISO-8859-1')

    df.rename(columns={"v1": "Category", "v2": "Message"},
inplace=True)
    df.drop(columns={'Unnamed: 2','Unnamed: 3','Unnamed: 4'},
inplace=True, errors='ignore')
    df['Spam'] = df['Category'].apply(lambda x: 1 if x == 'spam'
else 0)

    clf = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('nb', MultinomialNB())
    ])

    clf.fit(df.Message, df.Spam)

    return clf

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/detect', methods=['POST'])
def detect():
    try:
        data = request.json
        email_text = data.get('email_text', '').strip()

        if not email_text:
            return jsonify({
                'success': False,
                'error': 'Please enter an email text'
            }), 400

        prediction = clf.predict([email_text])[0]
        confidence = max(clf.predict_proba([email_text])[0]) * 100

        if prediction == 0:
            result = "Ham"
            message = "✓ This is a legitimate email"
            color = "success"
        else:
```

```python
            result = "Spam"
            message = "✗ This email is likely spam"
            color = "danger"

        return jsonify({
            'success': True,
            'result': result,
            'message': message,
            'confidence': f"{confidence:.2f}",
            'color': color
        })

    except Exception as e:
        return jsonify({
            'success': False,
            'error': f"Error processing email: {str(e)}"
        }), 500

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/api/stats', methods=['GET'])
def get_stats():
    return jsonify({
        'model': 'Multinomial Naive Bayes',
        'accuracy': '98.21%',
        'precision': '98.26%',
        'recall': '88.48%',
        'f1_score': '93.11%',
        'training_samples': 5572
    })

```

```python
if __name__ == '__main__':
    print("Training spam detection model...")
    train_model()
    print("Model trained successfully!")
    print("Starting Flask application...")
    port = int(os.environ.get('PORT', 8080))
    app.run(debug=False, host='0.0.0.0', port=port)
```

Frontend Implementation (HTML Templates)

The web interface is designed with a clean and modern layout, making it easy for users to submit email text and instantly view the classification results.

- Key Features of the Web Application:
  1. User Interface Components:
     - A simple, professional header for a clear and trustworthy presentation.
     - A large text area where users can paste any email message for analysis.
     - Mobile-responsive layout using HTML, CSS, and Bootstrap to ensure smooth access across devices.
  2. Spam Detection Support:
     - AI-powered email classification that clearly displays whether the message is Spam or Ham.
     - Confidence scores showing the probability of the prediction (e.g., “98.45% confidence”).
  3. Safety Features:
     - Input validation to ensure the user submits meaningful text.
     - Safe data handling through the Flask backend (suitable for both local demos and deployable environments).

Index.Html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Email Spam Detection System</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.0/font/boo
tstrap-icons.css">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-custom">
```

```
        <div class="container-lg">
            <a class="navbar-brand" href="/">
                <i class="bi bi-shield-check"></i> Email Spam
Detector
            </a>
            <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link active"
href="/">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
href="/about">About</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
href="/contact">Contact Us</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

    <section class="hero-section">
        <div class="container-lg h-100">
            <div class="row h-100 align-items-center">
                <div class="col-lg-8 mx-auto text-center">
                    <h1 class="display-4 fw-bold mb-3
```

```
text-white">Email Spam Detection</h1>
                    <p class="lead text-light mb-4">
                   Using Machine Learning - Instantly detect spam
and phishing emails with 98% accuracy
                    </p>
                    <div class="stats-display mb-4">
                        <div class="stat-box">
                            <span
class="stat-label">Accuracy</span>
                            <span class="stat-value">98.21%</span>
                        </div>
                        <div class="stat-box">
                            <span class="stat-label">Model</span>
                            <span class="stat-value">Naive
Bayes</span>
                        </div>
                        <div class="stat-box">
                            <span class="stat-label">Training
Data</span>
                            <span class="stat-value">5572
Emails</span>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>

    <section class="py-5">
        <div class="container-lg">
            <div class="row justify-content-center">
                <div class="col-lg-8">
                    <div class="card shadow-lg border-0 mb-4">
```

```
                        <div class="card-body p-4">
                            <h2 class="card-title mb-4">
                                <i class="bi
bi-envelope-open"></i> Analyze Email
                            </h2>
                            <form id="spamForm">
                                <div class="mb-3">
                                    <label for="emailText"
class="form-label fw-semibold">
                                        Enter Email Content
                                    </label>
                                    <textarea class="form-control
form-control-lg" id="emailText"
                                              placeholder="Paste
the email content or subject line here..."
                                              rows="6"
required></textarea>
                                    <small class="text-muted
d-block mt-2">

                                    </small>
                                </div>
                                <button type="submit" class="btn
btn-primary btn-lg w-100">
                                    <i class="bi bi-search"></i>
Analyze Email
                                </button>
                            </form>
                        </div>
                    </div>

                    <div id="resultContainer" class="d-none">
                        <div class="card shadow-lg border-0 mb-4"
```

```
id="resultCard">
                            <div class="card-body p-4">
                                <div class="text-center mb-4">
                                    <div id="resultIcon"
class="result-icon mb-3"></div>
                                    <h3 id="resultStatus"
class="card-title fw-bold"></h3>
                                    <p id="resultMessage"
class="lead mb-3"></p>
                                </div>

                                <div class="progress mb-3"
style="height: 25px;">
                                    <div id="confidenceBar"
class="progress-bar" role="progressbar"
                                         style="width: 0%;">
                                        <span id="confidenceText"
class="fw-semibold"></span>
                                    </div>
                                </div>

                                <small class="text-muted">
                                    <i class="bi
bi-lightbulb"></i> Confidence Score
                                </small>
                            </div>
                        </div>

                        <div class="d-grid gap-2">
                            <button type="button" class="btn
btn-primary btn-lg" onclick="resetForm()">
                                <i class="bi
bi-arrow-counterclockwise"></i> Check Another Email
```

```
                            </button>
                        </div>
                    </div>

                    <div id="loadingSpinner" class="d-none
text-center">
                        <div class="spinner-border text-primary"
role="status">
                            <span
class="visually-hidden">Loading...</span>
                        </div>
                        <p class="text-muted mt-3">Analyzing
email...</p>
                    </div>

                    <div id="errorAlert" class="alert alert-danger
d-none" role="alert">
                        <i class="bi bi-exclamation-circle"></i>
<span id="errorMessage"></span>
                    </div>
                </div>
            </div>
        </div>
    </section>

    <section class="py-5 bg-light">
        <div class="container-lg">
            <h2 class="text-center mb-5 fw-bold">Why Choose Our
Detector?</h2>
            <div class="row g-4">
                <div class="col-md-4">
                    <div class="feature-box">
                        <div class="feature-icon">
```

```
                            <i class="bi bi-lightning-fill"></i>
                        </div>
                        <h5 class="fw-bold mb-3">Lightning
Fast</h5>
                        <p>Get results in milliseconds. No delays,
no waiting. Instant email analysis.</p>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="feature-box">
                        <div class="feature-icon">
                            <i class="bi bi-shield-lock"></i>
                        </div>
                        <h5 class="fw-bold mb-3">Highly
Accurate</h5>
                        <p>98.21% accuracy by advanced machine
learning algorithms.</p>
                    </div>
                </div>
                <div class="col-md-4">
                    <div class="feature-box">
                        <div class="feature-icon">
                            <i class="bi bi-lock"></i>
                        </div>
                        <h5 class="fw-bold mb-3">100% Private</h5>
                        <p>Your emails are never stored. Complete
privacy and data protection guaranteed.</p>
                    </div>
                </div>
            </div>
        </div>
    </section>
    <footer>
        <div class="container-lg text-center">
            <p>&copy; 2025 Email Spam Detection System. All rights
reserved.</p>
            <p class="text-muted"> Project by Daksh Chaurasia and
Team</p>
        </div>
    </footer>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>
    <script src="{{ url_for('static', filename='js/script.js')
}}"></script>
</body>
</html>
```

About.html:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>About - Email Spam Detection System</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.0/font/boo
tstrap-icons.css">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark
shadow-sm">
        <div class="container-lg">
            <a class="navbar-brand fw-bold" href="/">
                <i class="bi bi-shield-check"></i> Spam Detector
            </a>
            <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active"
href="/about">About</a>
```

```
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
href="/contact">Contact Us</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

    <section class="py-5">
        <div class="container-lg">
            <div class="row">
                <div class="col-lg-8 mx-auto">
                    <h1 class="display-4 fw-bold mb-4">About This
Project</h1>

                    <div class="card shadow-lg border-0 mb-4">
                        <div class="card-body p-4">
                            <h2 class="card-title fw-bold mb-3">
                                <i class="bi bi-info-circle"></i>
Project Overview
                            </h2>
                            <p class="lead">
                                The Email Spam Detection System is
a machine learning-based application designed to
                                automatically classify emails as
spam or legitimate (ham) messages.
                            </p>

                        </div>
                    </div>
                    <div class="card shadow-lg border-0 mb-4">
                        <div class="card-body p-4">
                            <h2 class="card-title fw-bold mb-3">
                                <i class="bi bi-gear"></i>
Technical Details
                            </h2>
                            <div class="row">
                                <div class="col-md-6 mb-3">
                                    <h5>Machine Learning
Model</h5>
                                    <p>Multinomial Naive Bayes
Classifier</p>
                                </div>
                                <div class="col-md-6 mb-3">
                                    <h5>Training Dataset</h5>
                                    <p>5,572 labeled emails</p>
                                </div>
                                <div class="col-md-6 mb-3">
                                    <h5>Accuracy</h5>
                                    <p>98.21% on test dataset</p>
                                </div>
                                <div class="col-md-6 mb-3">
                                    <h5>Technology Stack</h5>
                                    <p>Python, Flask,
scikit-learn</p>
                                </div>
                            </div>
                        </div>
                    </div>

                    <div class="card shadow-lg border-0 mb-4">
                        <div class="card-body p-4">
                            <h2 class="card-title fw-bold mb-3">
```

```
                                <i class="bi bi-bar-chart"></i>
Performance Metrics
                            </h2>
                            <table class="table table-borderless">
                                <tbody>
                                    <tr>
                                        <td class="fw-bold">Test
Accuracy:</td>
                                        <td>98.21%</td>
                                    </tr>
                                    <tr>
                                        <td
class="fw-bold">Precision:</td>
                                        <td>98.26%</td>
                                    </tr>
                                    <tr>
                                        <td
class="fw-bold">Recall:</td>
                                        <td>88.48%</td>
                                    </tr>
                                    <tr>
                                        <td
class="fw-bold">F1-Score:</td>
                                        <td>93.11%</td>
                                    </tr>
                                    <tr>
                                        <td
class="fw-bold">ROC-AUC:</td>
                                        <td>0.9714</td>
                                    </tr>
                                </tbody>
                            </table>
                        </div>
```

```
                    </div>

                    <div class="card shadow-lg border-0 mb-4">
                        <div class="card-body p-4">
                            <h2 class="card-title fw-bold mb-3">
                                <i class="bi bi-lightbulb"></i>
Key Features
                            </h2>
                            <ul class="list-unstyled">
                                <li class="mb-2">
                                    <i class="bi
bi-check-circle-fill text-success"></i>
                                    <strong>Real-Time
Analysis:</strong> Instant detection of spam emails
                                </li>
                                <li class="mb-2">
                                    <i class="bi
bi-check-circle-fill text-success"></i>
                                    <strong>High
Accuracy:</strong> 98%+ accuracy in classification
                                </li>
                                <li class="mb-2">
                                    <i class="bi
bi-check-circle-fill text-success"></i>
                                    <strong>User-Friendly
Interface:</strong> Simple and intuitive web UI
                                </li>
                                <li class="mb-2">
                                    <i class="bi
bi-check-circle-fill text-success"></i>
                                    <strong>Confidence
Scores:</strong> Probability-based predictions
                                </li>
                                <li class="mb-2">
                                    <i class="bi
bi-check-circle-fill text-success"></i>
                                    <strong>Privacy
Focused:</strong> No data storage or tracking
                                </li>
                            </ul>
                        </div>
                    </div>

                    <div class="card shadow-lg border-0 mb-4">
                        <div class="card-body p-4">
                            <h2 class="card-title fw-bold mb-3">
                                <i class="bi
bi-question-circle"></i> How It Works
                            </h2>
                            <ol>
                                <li class="mb-2">
                                    <strong>Input:</strong> You
provide email text or subject
                                </li>
                                <li class="mb-2">

<strong>Preprocessing:</strong> Text is cleaned and tokenized
                                </li>
                                <li class="mb-2">
                                    <strong>Feature
Extraction:</strong> Words are converted to numerical features
                                </li>
                                <li class="mb-2">

<strong>Classification:</strong> Naive Bayes model predicts class
                                </li>
                                <li class="mb-2">
                                    <strong>Output:</strong>
Result with confidence score is displayed
                                </li>
                            </ol>
                        </div>
                    </div>

                    <div class="text-center mb-4">
                        <a href="/" class="btn btn-primary
btn-lg">
                            <i class="bi bi-house"></i> Back to
Home
                        </a>
                    </div>
                </div>
            </div>
        </div>
    </section>

    <!-- Footer -->
    <footer class="bg-dark text-white text-center py-4 mt-5">
            <p class="mb-2">
                <strong>Email Spam Detection System | Gen_Ai
Project</strong>
            </p>
            <p class="text-muted small mb-0">
                © 2025 All rights reserved. Built with Python,
Flask, and scikit-learn.
            </p>
        </div>
    </footer>
    <!-- Scripts -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>
</html>
```

Script.js

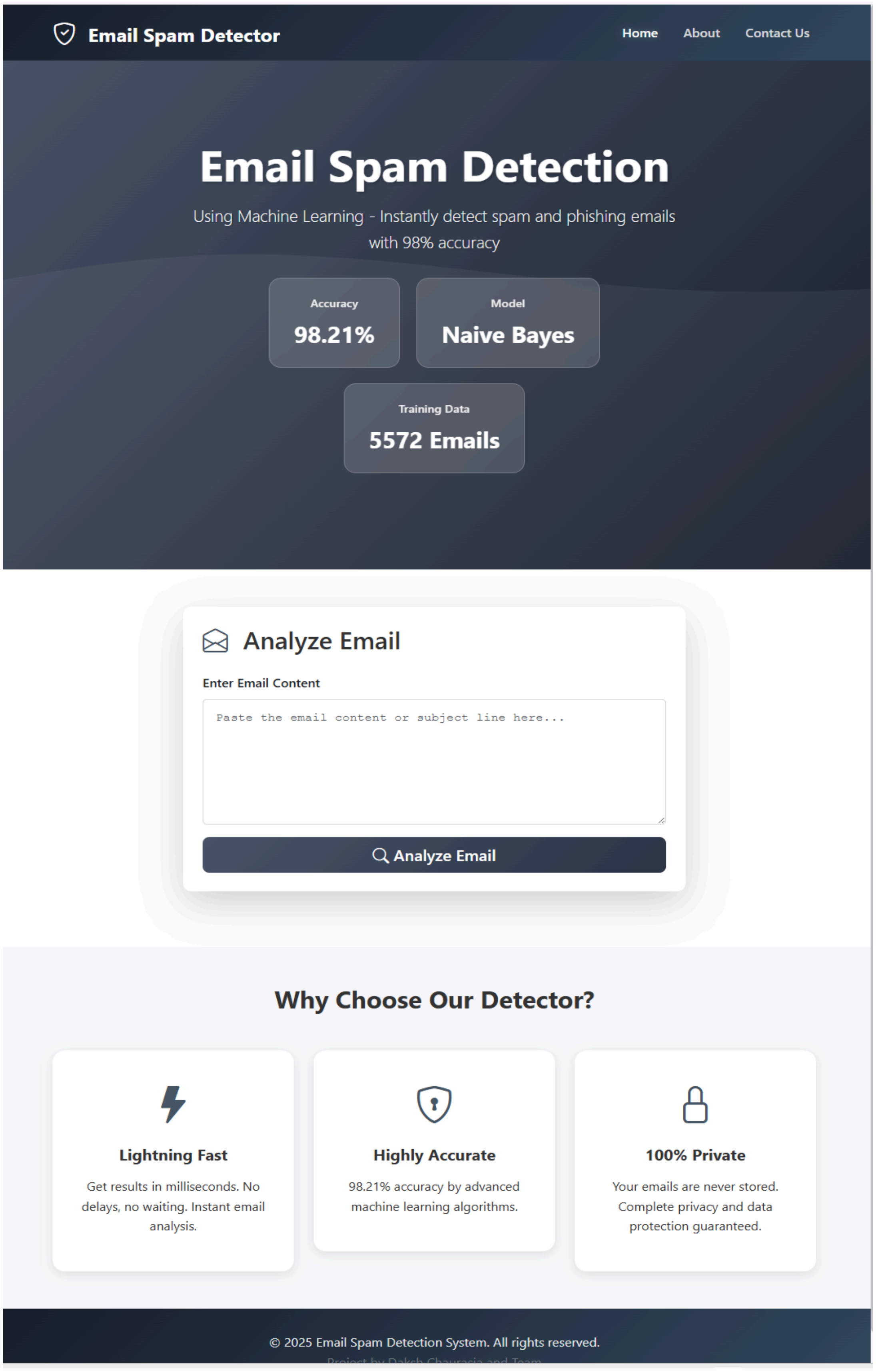
```
document.addEventListener('DOMContentLoaded', function() {
    const spamForm = document.getElementById('spamForm');
    const emailInput = document.getElementById('emailText');

    if (spamForm) {
        spamForm.addEventListener('submit', handleFormSubmit);
    }
});

async function handleFormSubmit(e) {
    e.preventDefault();

    const emailText =
document.getElementById('emailText').value.trim();
    const resultContainer =
document.getElementById('resultContainer');
    const loadingSpinner =
document.getElementById('loadingSpinner');
    const errorAlert = document.getElementById('errorAlert');

    resultContainer.classList.add('d-none');
    errorAlert.classList.add('d-none');
    loadingSpinner.classList.remove('d-none');
    try {
        const response = await fetch('/detect', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                email_text: emailText
            })
        });

        if (!response.ok) {
            const error = await response.json();
            showError(error.error || 'An error occurred');
            return;
        }

        const data = await response.json();

        if (data.success) {
            displayResult(data);
        } else {
            showError(data.error);
        }
    } catch (error) {
        console.error('Error:', error);
        showError('Failed to connect to the server. Please try
again.');
    } finally {
        loadingSpinner.classList.add('d-none');
    }
}
```

```
function displayResult(data) {
    const resultContainer =
document.getElementById('resultContainer');
    const resultCard = document.getElementById('resultCard');
    const resultStatus = document.getElementById('resultStatus');
    const resultMessage =
document.getElementById('resultMessage');
    const resultIcon = document.getElementById('resultIcon');
    const confidenceBar =
document.getElementById('confidenceBar');
    const confidenceText =
document.getElementById('confidenceText');

    resultCard.classList.remove('success-card', 'danger-card');
    resultIcon.innerHTML = '';

    if (data.result === 'Ham') {
        resultStatus.textContent = '✓ Legitimate Email';
        resultStatus.style.color = '#27ae60';
        resultMessage.textContent = data.message;
        resultIcon.innerHTML = '<i class="bi bi-check-circle-fill
result-success"></i>';
        resultCard.classList.add('success-card');
    } else {
        resultStatus.textContent = '✗ Spam Email';
        resultStatus.style.color = '#e74c3c';
        resultMessage.textContent = data.message;
        resultIcon.innerHTML = '<i class="bi
bi-exclamation-circle-fill result-danger"></i>';
        resultCard.classList.add('danger-card');
    }
    const confidence = parseFloat(data.confidence);
    confidenceBar.style.width = confidence + '%';
    confidenceText.textContent = confidence.toFixed(2) + '%';

    if (confidence >= 90) {
        confidenceBar.style.background = 'linear-gradient(90deg,
#27ae60 0%, #229954 100%)';
    } else if (confidence >= 70) {
        confidenceBar.style.background = 'linear-gradient(90deg,
#f39c12 0%, #e67e22 100%)';
    } else {
        confidenceBar.style.background = 'linear-gradient(90deg,
#e74c3c 0%, #c0392b 100%)';
    }

    resultContainer.classList.remove('d-none');
    resultContainer.scrollIntoView({ behavior: 'smooth', block:
'nearest' });
}

function showError(errorMessage) {
    const errorAlert = document.getElementById('errorAlert');
    const errorMessageSpan =
document.getElementById('errorMessage');

    errorMessageSpan.textContent = errorMessage;
    errorAlert.classList.remove('d-none');

    setTimeout(() => {
        errorAlert.classList.add('d-none');
    }, 5000);
}
```

```
function resetForm() {
    document.getElementById('spamForm').reset();

document.getElementById('resultContainer').classList.add('d-none')
;
    document.getElementById('errorAlert').classList.add('d-none');
    document.getElementById('emailText').focus();
}

function trimString(str) {
    return str.replace(/^\s+|\s+$/g, '');
}

function formatPercentage(value) {
    return (parseFloat(value) * 100).toFixed(2) + '%';
}

document.querySelectorAll('a[href^="#"]').forEach(anchor => {
    anchor.addEventListener('click', function (e) {
        e.preventDefault();
        const target =
document.querySelector(this.getAttribute('href'));
        if (target) {
            target.scrollIntoView({
                behavior: 'smooth',
                block: 'start'
            });
        }
    });
});

document.addEventListener('keydown', function(event) {
    if (event.ctrlKey && event.key === 'Enter') {
        const spamForm = document.getElementById('spamForm');
        if (spamForm &&
!document.getElementById('resultContainer').classList.contains('d-
none')) {
            handleFormSubmit(new Event('submit'));
        }
    }

    if (event.key === 'Escape') {
        const resultContainer =
document.getElementById('resultContainer');
        if (!resultContainer.classList.contains('d-none')) {
            resetForm();
        }
    }
});

document.getElementById('emailText')?.addEventListener('dblclick',
function() {
    this.value = 'Free Tickets for IPL! Click here to claim your
prize now.';
});
const emailInput = document.getElementById('emailText');
if (emailInput) {
    emailInput.addEventListener('input', function() {
        const charCount = this.value.length;
    });
}
```

Application Screenshots

1. **Landing Page (index.html):**
   Users paste the email content into the text box and click Analyze Email.



Email Spam Detector

Home About Contact Us

Email Spam Detection

Using Machine Learning - Instantly detect spam and phishing emails with 98% accuracy

Accuracy	Model	Training Data
98.21%	Naive Bayes	5572 Emails

Analyze Email

Enter Email Content

```
Paste the email content or subject line here...
```

Analyze Email

Why Choose Our Detector?

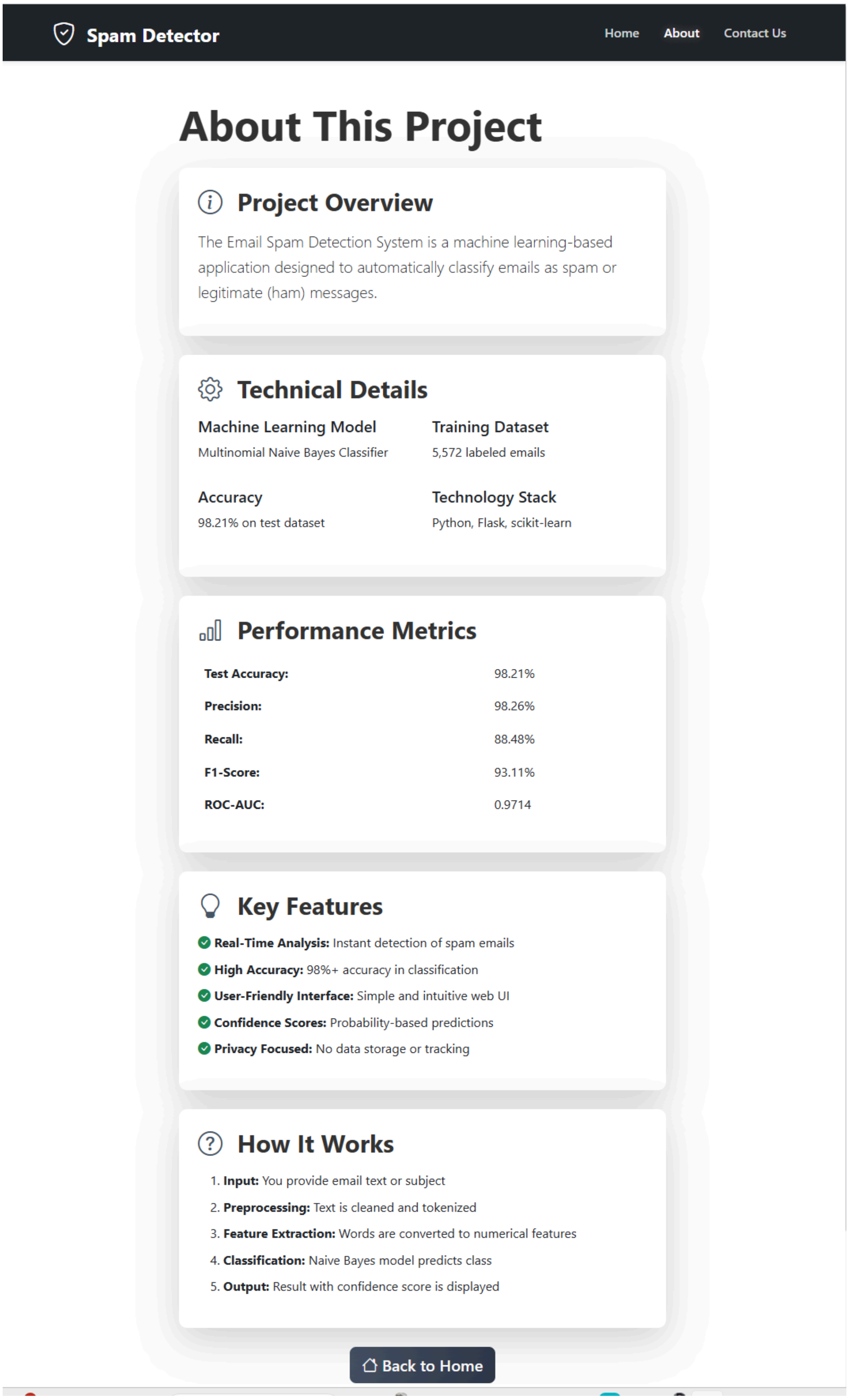
Lightning Fast
Get results in milliseconds. No delays, no waiting. Instant email analysis.

Highly Accurate
98.21% accuracy by advanced machine learning algorithms.

100% Private
Your emails are never stored. Complete privacy and data protection guaranteed.

© 2025 Email Spam Detection System. All rights reserved.
Project by Daksh Chaurasia and Team

2. About page (about.html):
Explains how the system works



About This Project

Project Overview

The Email Spam Detection System is a machine learning-based application designed to automatically classify emails as spam or legitimate (ham) messages.

Technical Details

Machine Learning Model
Multinomial Naive Bayes Classifier

Training Dataset
5,572 labeled emails

Accuracy
98.21% on test dataset

Technology Stack
Python, Flask, scikit-learn

Performance Metrics

Test Accuracy:	98.21%
Precision:	98.26%
Recall:	88.48%
F1-Score:	93.11%
ROC-AUC:	0.9714

Key Features

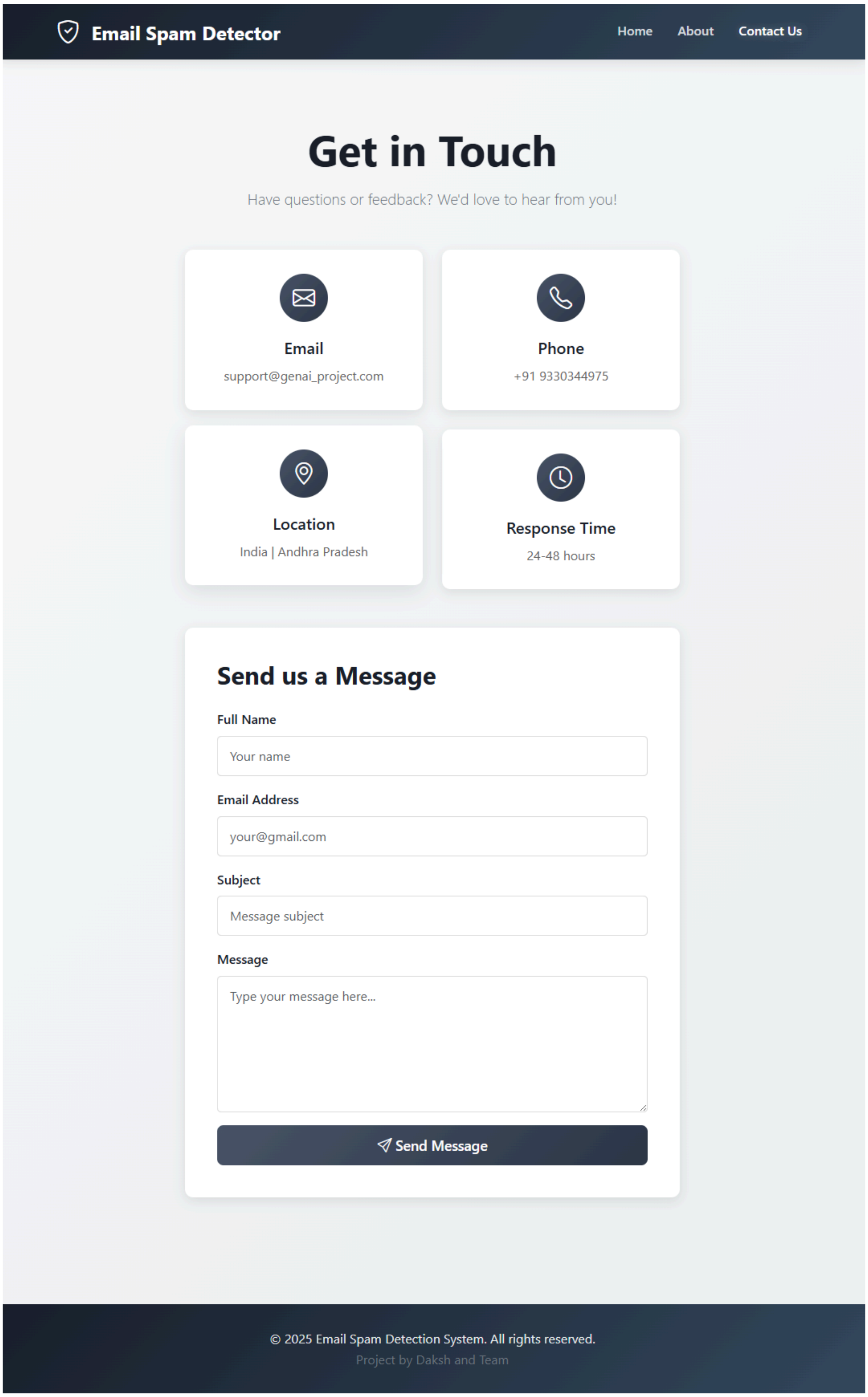
- **Real-Time Analysis:** Instant detection of spam emails
- **High Accuracy:** 98%+ accuracy in classification
- **User-Friendly Interface:** Simple and intuitive web UI
- **Confidence Scores:** Probability-based predictions
- **Privacy Focused:** No data storage or tracking

How It Works

1. **Input:** You provide email text or subject
2. **Preprocessing:** Text is cleaned and tokenized
3. **Feature Extraction:** Words are converted to numerical features
4. **Classification:** Naive Bayes model predicts class
5. **Output:** Result with confidence score is displayed

3. Contact page (contact.html):

Provides support information



Email Spam Detector

Home About Contact Us

Get in Touch

Have questions or feedback? We'd love to hear from you!

Email
support@genai_project.com

Phone
+91 9330344975

Location
India | Andhra Pradesh

Response Time
24-48 hours

Send us a Message

Full Name
Your name

Email Address
your@gmail.com

Subject
Message subject

Message
Type your message here...

Send Message

© 2025 Email Spam Detection System. All rights reserved.
Project by Daksh and Team

Future Implementations

Future plans for the Email Spam Detection system focus on expanding functionality, improving accuracy, and enhancing deployment flexibility:

- **Email Service Integration:** Build API endpoints to integrate the model directly with platforms like Gmail, Outlook, or company email servers for automatic spam filtering.
- **Multi-Model Deployment:** Experiment with ensemble approaches such as Voting or Stacking Classifiers to combine Naive Bayes, Logistic Regression, and SVM for improved robustness.
- **Browser Extension:**Develop a Chrome/Edge extension that detects and flags spam characteristics directly inside the user's inbox.
- **Continuous Improvement:** Expand the dataset with multilingual email samples and updated phishing patterns to improve adaptability and long-term reliability.

Conclusion

The Email Spam Detection Project successfully delivers a fast, accurate, and highly practical spam-classification system. Using the Multinomial Naive Bayes algorithm, the model achieves excellent performance on real-world email data and provides reliable predictions in milliseconds. The final solution is a complete full-stack web application built with Flask, scikit-learn, and a responsive frontend, seamlessly integrating the trained ML model into an easy-to-use interface. This project demonstrates a strong end-to-end pipeline from data processing and model training to deployment resulting in a scalable and effective tool for smart, real-time spam detection.