EMAIL SPAM DETECTION USING MACHINE LEARNING

SUBMITTED BY:
SRUJANA SIRISOLLA

UNDER THE GUIDANCE OF :

HUNARINTERN ORGANISATION

DOMAIN:MACHINE LEARNING

SUBMITTED ON:
[25-06-2025]

PROJECT OBJECTIVE:

The main goal of this project is to build a machine learning-based model that can classify emails as either spam or not spam (ham). The system aims to automate email filtering and enhance digital communication security.

DATASET USED:

Source: Kaggle

DESCRIPPTION: The dataset contains two main columns:

LABEL: Indicates whether the message is 'spam' or 'ham'

MESSAGE: The content of the email

🛠 TOOLS & TECHNOLOGIES:

Programming Language: Python

LIBRARIES: pandas, numpy, sklearn, matplotlib

ML ALGORITHMS: Multinomial Naive Bayes, Support Vector Machine (SVM)

TEXT PROCESSING: CountVectorizer, TfidfVectorizer

Methodology:

Data Preprocessing:

Load dataset using pandas

Clean and normalize text data

Encode labels (ham = 0, spam = 1)

Feature Extraction:

Use CountVectorizer or TF-IDF to convert text into numerical features

Model Training:

Split dataset into training and testing sets (e.g., 80/20)

Train models using:

Multinomial Naive Bayes

Support Vector Machine

Model Evaluation:

Use metrics like accuracy, precision, recall, F1-score

Plot confusion matrix to evaluate performance

CODE EXPLANATION:

```
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Step 2: Load the dataset
df = pd.read_excel("C:/datasets/spam email.xlsx",usecols=[0, 1], names=['label', 'message'])

# Step 3: Preprocessing the data
df.dropna(inplace=True)  # Remove any null values
```

```
df['label'] = df['label'].map({'ham': 0, 'spam': 1})  # Convert labels to 0 (ham) and 1 (spam)
#convert all msgs to string
df['message']=df['message'].astype(str)
# Step 4: Splitting into train and test sets
X = df['message']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Convert text to numerical features using TF-IDF
vectorizer = TfidfVectorizer()
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)

# Step 6: Train Naive Bayes model
nb_model = MultinomialNB()
nb_model.fit(X_train_vect, y_train)
nb_preds = nb_model.predict(X_test_vect)

# Step 7: Train Support Vector Machine model
svm_model = SVC()
svm_model.fit(X_train_vect, y_train)
svm_preds = svm_model.predict(X_test_vect)
```

Step 8: Evaluation

print("=== Naive Bayes Results ===")

print("Accuracy:", accuracy_score(y_test, nb_preds))

print(confusion_matrix(y_test, nb_preds))

print(classification_report(y_test, nb_preds))

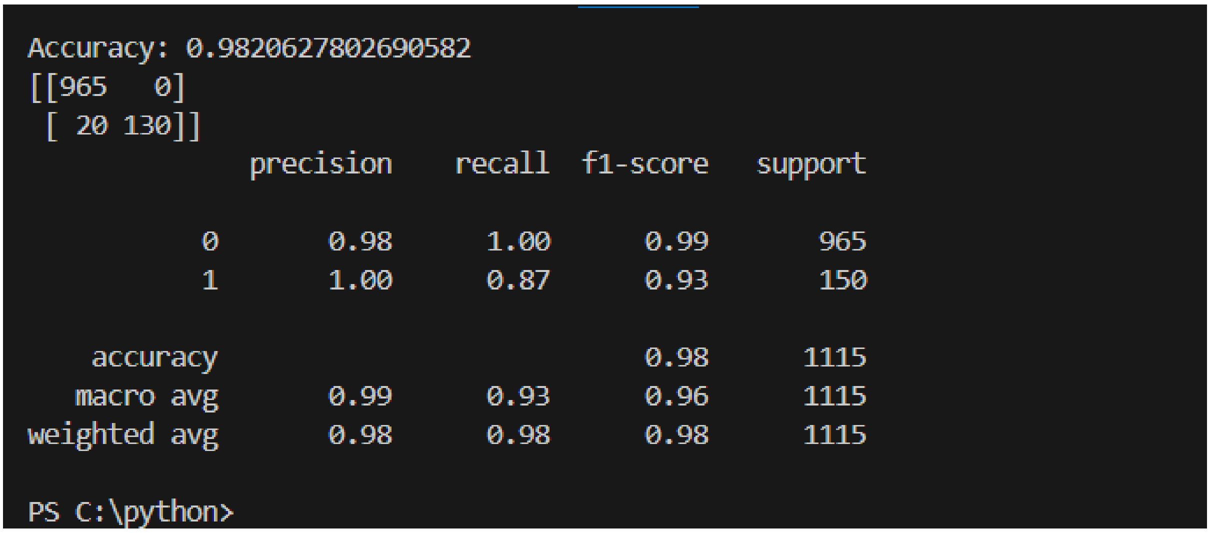
print("\n=== Support Vector Machine Results ===")

print("Accuracy:", accuracy_score(y_test, svm_preds))

print(confusion_matrix(y_test, svm_preds))

print(classification_report(y_test,svm_preds))

TESTING AND OUTPUT:



```
Accuracy: 0.9820627802690582
[[965   0]
 [ 20 130]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       965
           1       1.00      0.87      0.93       150

    accuracy                           0.98      1115
   macro avg       0.99      0.93      0.96      1115
weighted avg       0.98      0.98      0.98      1115

PS C:\python>
```

Conclusion:

The developed email spam classifier successfully distinguishes between spam and non-spam emails with high accuracy. Naive Bayes performed slightly better due to its suitability for text classification problems Confusion matrix and classification reports were used to compare performance.