A

SYNOPSIS

of

MINOR PROJECT

on

SPAM EMAIL DETECTOR

Submitted by

NEERAJ PAREEK

ROLL NO. 21EGICA023

Project Guide
Mr. Ritesh kumar jain

Head of Department
Dr. Mayank Patel

Geetanjali Institute of Technical Studies, Dabok , Udaipur (Raj.)
Department of Computer Science and Engineering
July,2024

Problem Statement

Spam emails are a major issue in today's digital communication, leading to wasted time, reduced productivity, and potential security threats. The challenge is to develop a reliable spam detection system that can automatically classify emails as spam or ham (not spam) to improve email filtering processes.

Brief Description

The Spam Email Detector project automates spam email filtering using machine learning. Utilizing the Email Spam Collection dataset, it involves data loading, exploration, text pre-processing, and TF-IDF vectorization. The Multinomial Naive Bayes algorithm trains the model, with performance evaluated through accuracy, precision, recall, and F1-score. Custom input testing is included, showcasing NLP and machine learning to enhance email filtering efficiency and accuracy.

Objective and Scope

Objective:

1. Accurately classify Email messages as spam or ham using machine learning.
2. Achieve high performance in accuracy, precision, recall, and F1-score.
3. Provide a scalable solution with user testing capability.

Scope:

1. Pre-process the Email Spam Collection dataset for model training.
2. Implement and optimize a Multinomial Naive Bayes classifier.
3. Optionally deploy the model using Flask for user-friendly testing.

Methodology

The methodology outlines the step-by-step process followed to achieve the project's objectives:

1. **Data Collection**: Download the Email Spam Collection dataset, load it into a pandas Data-Frame.
2. **Data Preprocessing**: Clean text (remove special characters, convert to lowercase, eliminate stop-words), normalize text (stemming/lemmatization), and encode labels (spam/ham to 1/0).
3. **Feature Extraction**: Convert cleaned text into numerical vectors using TF-IDF.
4. **Model Training**: Split data into training/testing sets and train a Multinomial Naive Bayes classifier.
5. **Model Evaluation**: Use accuracy, precision, recall, F1-score, and confusion matrix to evaluate performance.
6. **Model Testing**: Test the model with custom input messages for real-world validation.

Hardware and Software Requirements

Hardware:

1. Standard computer with at least 4 GB of RAM.
2. Modern CPU for data processing and model training.

Software:

1. **Operating System**: Windows, macOS, or Linux.
2. **Programming Language**: Python.
3. **Development Environment**: Jupyter Notebook or Google Colab, VSCode

Technologies

1. **Python**: Primary programming language.
2. p**andas**: Used for dataset loading and preprocessing.
3. **scikit-learn**: Utilized for feature extraction, model training, and evaluation.
4. **TF-IDF Vectorizer**: Converts text data into numerical format.
5. **Multinomial Naive Bayes**: Chosen algorithm for spam and ham classification.
6. **Flask**: Web framework for optional model deployment via a simple web interface.

Testing Techniques

1. **Cross-Validation**: Split dataset for training/testing to validate model performance.
2. **Evaluation Metrics**: Assess accuracy, precision, recall, and F1-score for model effectiveness.
3. **Metrics**: Accuracy measures overall correctness, precision evaluates positive prediction accuracy (spam), recall assesses true positive detection (spam), and F1-score balances precision and recall for overall performance.

Project Contribution

1. **Enhanced Email Security**: Robust solution to detect and filter spam, improving email security and user experience.
2. **Educational Value**: Demonstrates complete model development process from data preprocessing to evaluation.
3. **Scalability and Adaptability**: Versatile model adaptable to various types of textual data.
4. **Practical Application**: Integration into existing email systems or standalone use for spam filtering.
5. **Interactive Testing**: User-friendly interface for testing with custom messages, ensuring real-world applicability.

Project Screenshots

```python
# Step 1: Import libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```python
# Step 2: Load the dataset
# Upload the 'email.csv' file to Google Colab before running this cell
data = pd.read_csv('/content/emails.csv')
```

```python
data.head()
```

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

```python
# Step 6: Test the model with custom input
def predict_spam(message):
    message_vector = vectorizer.transform([message])
    prediction = model.predict(message_vector)
    return 'Spam' if prediction[0] == 1 else 'Ham'

# Example usage
test_message_spam = "Congratulations! You've won a $1000 Walmart gift card. Click here to claim your prize."
test_message_ham = "Hey, are you free for lunch today?"

print(f'Test message (Spam): {test_message_spam}')
print(f'Prediction: {predict_spam(test_message_spam)}')

print(f'\nTest message (Ham): {test_message_ham}')
print(f'Prediction: {predict_spam(test_message_ham)}')
```

```
Test message (Spam): Congratulations! You've won a $1000 Walmart gift card. Click here to claim your prize.
Prediction: Spam

Test message (Ham): Hey, are you free for lunch today?
Prediction: Ham
```