SPAM CLASSIFICATION: USING NLP AND ML FOR EFFECTIVE MANAGEMENT OF UNWANTED MESSAGES

A project report submitted to
MALLA REDDY UNIVERSITY
in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING (AI & ML)

Submitted by

B. Varshith	**:**	**2111CS020619**
C. Varshith	**:**	**2111CS020620**
G. Varun Reddy	**:**	**2111CS020625**
A. Veera Venkata Laxman	**:**	**2111CS020629**
A. Veerabhadra Sai Amarnath	**:**	**2111CS020630**

Under the Guidance of

Prof G. Balaiah

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

COLLEGE CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled, ”**Spam Classification: Using NLP And Machine Learning For Effective Management Of Unwanted Messages**” submitted by **B. Varshith(2111CS020619), C. Varshith(2111CS020620), G. Varun Reddy (2111CS020625), A. Veera Venkata Laxman(2111CS020629), A. Veerabhadra Sai Amarnath(2111CS020630)** of B.Tech IV year Ist semester, Department of CSE (AI&ML) during the year 2024- 25.The results embodied in the report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
Prof G. Balaiah
Assistant Professor

HEAD OF THE DEPARTMENT
Prof. Sujith Das

DEAN CSE(AI&ML)
Dr. Thayyaba Khatoon

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We sincerely thank our **DEAN Dr. Thayyaba Khatoon** for her constant support and motivation all the time. A special acknowledgement goes to a friend who enthused us from the back stage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods, and extending timely support.

We would like to express our gratitude to all those who extended their support and suggestions to come up with this application. Special Thanks to our mentor **Prof G. Balaiah** whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

Abstract

Spam classification is a crucial task in managing communication effectively. The goal of this project is to develop a Natural Language Processing (NLP) system that can accurately classify messages as spam or not spam. Spam messages, often called junk messages, include malicious advertisements or unwanted content. These can fill up communication channels and pose security risks.

In this project, we utilize various NLP techniques and machine learning algorithms to build a spam classification model. Specifically, we apply **NLP methods such as TF-IDF** for feature extraction, and we preprocess the **text using techniques** such as **tokenization, lemmatization, and stop-word removal**. Using **Machine Learning**, we employ **Support Vector Machines** (SVM) for classification to analyze patterns in the message content.

The outcomes of this project include a trained NLP-based spam classification model and a detailed performance analysis. The system can automatically filter out spam messages, making communication management more efficient and secure. This reduces the time spent on manual message sorting and enhances the overall user experience.

By implementing this project, we aim to offer a practical solution to a common problem in communication using advanced NLP techniques, ensuring users receive only relevant and important messages.

CONTENTS

1. INTRODUCTION

1.1 PROBLEM DEFINITION:

In today's digital communication landscape, the sheer volume of messages sent through email and mobile messaging apps includes a significant amount of spam—unsolicited content like promotional ads, phishing attempts, and potentially harmful links. This presents several challenges:

- **Cluttered Inboxes:** Users struggle to find genuine messages amid spam, leading to wasted time and reduced productivity.
- **Security Risks:** Spam messages often contain malicious links that can lead to data breaches and identity theft, making effective filtering essential.
- **User Frustration:** Current spam detection systems often misclassify messages, resulting in false positives (legitimate messages flagged as spam) and false negatives (spam messages slipping through). This can decrease user trust and expose them to threats.

To address these issues, a more effective spam classification system is needed.

1.2 OBJECTIVE OF PROJECT:

The primary objective of this project is to develop a robust Natural Language Processing (NLP) system that accurately classifies messages as spam or not spam. Key components include:

- **NLP Techniques:** Utilizing methods like Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction to prepare text data for analysis.
- **Machine Learning Algorithms:** Implementing Support Vector Machines (SVM) for classification, leveraging its effectiveness in handling text data.
- **Enhancing User Experience:** The goal is to improve the accuracy of spam detection, allowing users to focus on important communications without distractions.

This project aims to provide a reliable solution for enhancing security and efficiency in digital communication.

1.3 SCOPE & LIMITATIONS OF THE PROJECT:

Scope:

- **Target Platforms:** The focus is on spam classification for email and mobile messaging apps, addressing a widespread digital communication issue.
- **Integration of NLP and Machine Learning:** The project will integrate advanced techniques to enhance spam detection accuracy.
- **Real-World Applications:** The system can be adapted for use in existing messaging platforms to improve their spam detection capabilities.

Limitations:

- **Training Data Quality:** The model's effectiveness is reliant on the diversity and quality of the training data, which may affect classification accuracy.
- **Dynamic Nature of Spam:** As spam tactics evolve, the model may face challenges in classifying new spam types not present in the training dataset.
- **Real-Time Processing Constraints:** The system may experience latency during peak usage times, impacting user experience.
- **Feature Selection Complexity:** While TF-IDF is effective, additional feature selection techniques may enhance classification results further.

2. LITERATURE SURVEY

Sr. No	Title	Author(s)	Publication	How Developed	Limitations Noted
1	Spam Detection in Mobile Messaging Apps	Singh, A. & Sharma, R.	International Journal of Computer Applications	Used machine learning with feature extraction techniques.	Limited dataset size; high false positive rates.
2	An Enhanced Spam Filtering Technique Using Machine Learning	Kumar, R. & Gupta, S.	Journal of Computer Science	Developed a hybrid model combining different algorithms.	Ineffective feature selection; lack of real-time processing.
3	Comparative Study of Spam Detection Techniques	Verma, P. & Joshi, M.	International Journal of Innovative Research	Conducted experiments with various ML models.	Limited dataset size; complex models.
4	Natural Language Processing for Spam Classification	Mehta, A. & Patel, R.	Journal of Data Science	Implemented NLP techniques for feature extraction.	High false positive rates; ineffective feature selection.
5	A Review on Spam Detection Techniques	Rao, V. & Nair, S.	International Journal of Information Technology	Reviewed existing methods and proposed improvements.	Lack of real-time processing; complex models.

3. ANALYSIS

3.1 PROJECT PLANNING AND RESEARCH

Objective: The primary objective of this project is to develop a robust spam classification system that leverages Natural Language Processing (NLP) and machine learning techniques to efficiently filter unwanted messages.

Project Scope:

- **Research**: Investigate existing spam detection systems and identify their limitations. This involves reviewing literature, exploring various machine learning algorithms, and understanding the latest advancements in NLP techniques.
- **Development**: Design a model using Support Vector Machines (SVM) for classification, employing methods like TF-IDF for feature extraction and various preprocessing techniques.
- **Testing and Evaluation**: Implement rigorous testing phases to ensure the model's effectiveness, including cross-validation and performance metrics analysis.

Methodology:

1. **Literature Review**: Analyze existing research papers and case studies on spam detection.
2. **Data Collection**: Gather a comprehensive dataset consisting of both spam and ham messages.
3. **Implementation**: Develop the spam classification model following the proposed architecture.
4. **Evaluation**: Assess the model's performance using metrics such as accuracy, precision, recall, and F1-score.

Timeline:

- **Week 1-2**: Conduct literature review and finalize the dataset.
- **Week 3-4**: Implement preprocessing and feature extraction methods.
- **Week 5-6**: Train the SVM model and perform hyperparameter tuning.
- **Week 7**: Evaluate the model's performance and refine as needed.
- **Week 8**: Prepare documentation and deployment for the web application.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1 SOFTWARE REQUIREMENTS

The following software components are required for the successful development and execution of the spam classification system:

1. **Operating System**:
   - Windows, Linux, or macOS
2. **Programming Language**:
   - Python (version 3.6 or higher)
3. **Libraries and Frameworks**:
   - **Flask**: For web application development.
   - **Scikit-learn**: For machine learning algorithms and model evaluation.
   - **NLTK**: For natural language processing tasks, such as tokenization and lemmatization.
   - **Pandas**: For data manipulation and analysis.
   - **NumPy**: For numerical operations.
   - **Matplotlib/Seaborn**: For data visualization.
4. **Development Environment**:
   - Integrated Development Environment (IDE) such as PyCharm or Jupyter Notebook.
5. **Database**:
   - SQLite or any other lightweight database (optional, based on the need for persistent storage).
6. **Version Control System**:
   - Git (for code versioning and collaboration).

3.2.2 HARDWARE REQUIREMENTS

The following hardware specifications are recommended for optimal performance during development and execution of the spam classification system:

1. **Processor**:
   - Minimum: Dual-core processor (Intel i3 or equivalent).
   - Recommended: Quad-core processor (Intel i5 or equivalent) for faster processing, especially during model training.
2. **RAM**:
   - Minimum: 8 GB.
   - Recommended: 16 GB or more for handling large datasets and multitasking.
3. **Storage**:
   - Minimum: 100 GB of free disk space.
   - Recommended: SSD for faster read/write speeds.
4. **Network**:
   - Reliable internet connection for downloading necessary libraries and datasets.
5. **Other Requirements**:
   - Monitor with a resolution of 1920 x 1080 or higher for better visibility during development.
   - A keyboard and mouse for efficient coding and navigation.

3.3 MODEL SELECTION AND ARCHITECTURE

3.3.1 MODEL SELECTION

In developing the spam classification system, several machine learning algorithms were considered for their suitability in classifying messages as spam or not spam. The final selection was based on various criteria, including accuracy, interpretability, computational efficiency, and performance on textual data.

Considered Algorithms:

1. **Naive Bayes**:
   - Pros: Simple and fast; effective for large datasets.
   - Cons: Assumes feature independence, which may not hold true in real-world scenarios.
2. **Decision Trees**:
   - Pros: Intuitive and easy to interpret; handles categorical and numerical data.
   - Cons: Prone to overfitting; sensitive to small variations in the data.
3. **Random Forest**:
   - Pros: Combines multiple decision trees to improve accuracy and reduce overfitting.
   - Cons: More complex and resource-intensive; less interpretable than single decision trees.
4. **Support Vector Machines (SVM)**:
   - Pros: Effective in high-dimensional spaces; robust against overfitting, especially in cases where the number of dimensions exceeds the number of samples.
   - Cons: Requires careful tuning of hyperparameters; less effective with large datasets and noisy data.

Final Choice:

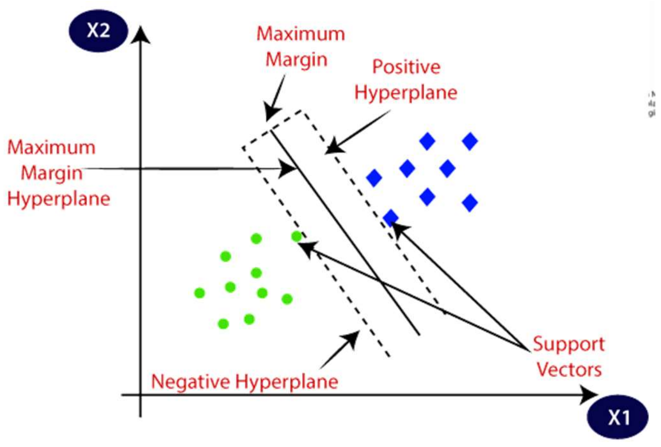


Fig 3.3.1: SVM

- **Support Vector Machines (SVM)** were selected for this project due to their high performance in text classification tasks and ability to handle large feature spaces effectively. SVM's capability to maximize the margin between different classes contributes to better generalization on unseen data.

3.3.2 MODEL ARCHITECTURE

The architecture of the spam classification model comprises several key components, which work together to process the input data and produce predictions.

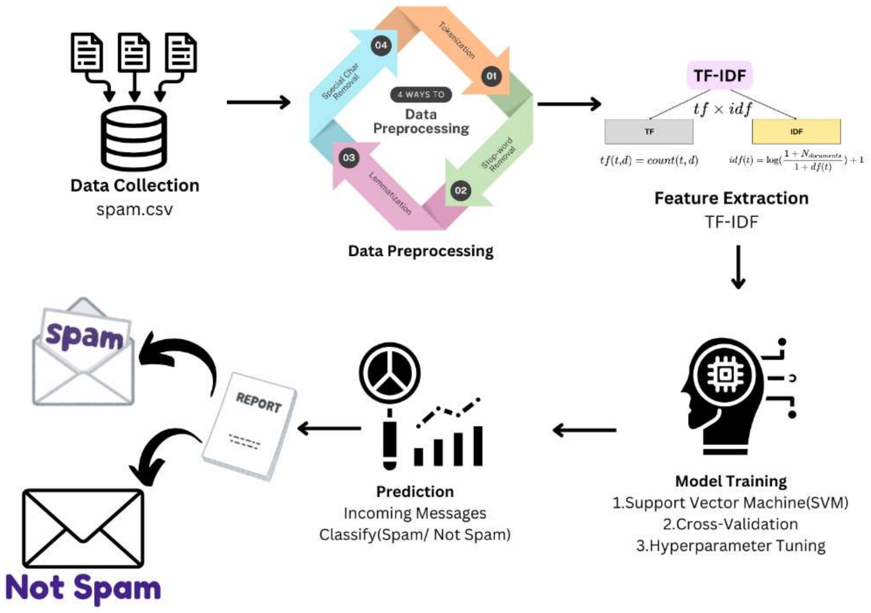


Fig 3.3.2: Model Architecture

Overview of Model Architecture:

1. **Input Layer**:
   - **Text Input**: Receives raw text messages for classification.
2. **Data Preprocessing Layer**:
   - **Tokenization**: Splits the input text into individual tokens (words).
   - **Lemmatization**: Reduces tokens to their base form to ensure consistency.
   - **Stop-word Removal**: Eliminates common words that do not contribute significant meaning (e.g., "and," "the").
3. **Feature Extraction Layer**:
   - **TF-IDF Vectorization**: Converts the preprocessed text data into a numerical format by calculating Term Frequency-Inverse Document Frequency (TF-IDF) scores for each term. This matrix representation captures the importance of words in the context of the entire dataset.
4. **Classification Layer**:
   - **Support Vector Machine**: The core classification algorithm that operates on the feature matrix generated from the previous layer. It constructs hyperplanes in a high-dimensional space to classify the messages as spam or not spam.
5. **Output Layer**:
   - **Prediction Output**: The model outputs the classification result, indicating whether the input message is "Spam" or "Not Spam."

Model Training Process:

- The model is trained using labeled data (spam and ham messages) to learn the optimal hyperplane that separates the two classes.
- Cross-validation techniques are employed to ensure robustness and prevent overfitting, while hyperparameter tuning is performed using methods like GridSearchCV to optimize the model's performance.

4. DESIGN

4.1 INTRODUCTION

The design phase establishes the structure and workflows for the spam classification system. This includes detailing system components, data flow, and the techniques and algorithms used. A well-structured design ensures that each component integrates seamlessly, contributing to the system's overall performance in accurately classifying spam and non-spam messages.

The design includes:

- An overview of the chosen project diagram (DFD/UML/ER).
- Details of the dataset used, including feature descriptions.
- Data preprocessing techniques applied to prepare the text data.
- A description of the algorithms and methods used in the classification model.

4.2 SEQUENCE DIAGRAM

The Sequence Diagram illustrates the interaction flow within the spam classification system, detailing how components communicate during the classification process. The diagram includes the following key interactions:

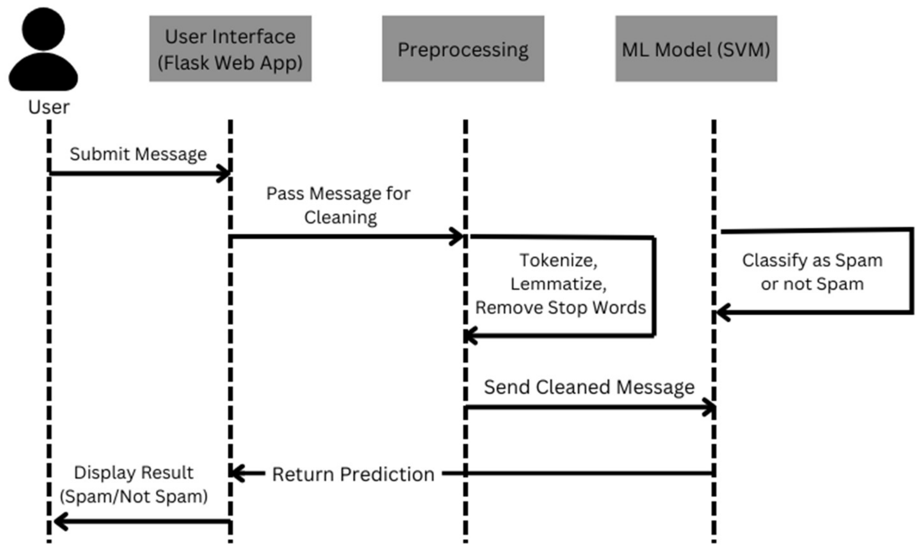
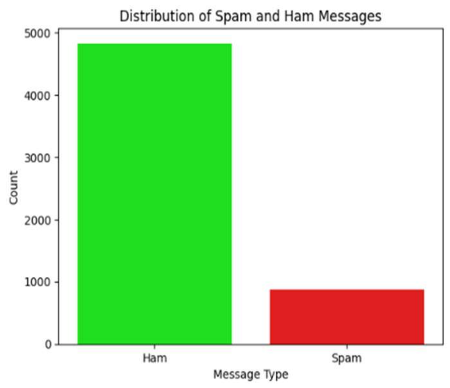

User
User Interface (Flask Web App)
Preprocessing
ML Model (SVM)
Submit Message
Pass Message for Cleaning
Tokenize, Lemmatize, Remove Stop Words
Classify as Spam or not Spam
Send Cleaned Message
Return Prediction
Display Result (Spam/Not Spam)

Fig 4.2: Sequence Diagram

1. **User Input:** The user initiates the process by submitting a message through the User Interface (Flask Web App).
2. **Message Forwarding:** The User Interface forwards the submitted message to the Preprocessing component for cleaning and preparation.
3. **Preprocessing Steps:** The Preprocessing component performs several tasks:
    - **Tokenization:** The message is broken down into individual words (tokens).
    - **Lemmatization:** Words are reduced to their base forms to standardize variations.
    - **Stop-word Removal:** Common words that do not contribute to meaning are removed.
4. **Feature Extraction:** After preprocessing, the cleaned message is sent to the ML Model (SVM) for classification.
5. **Prediction Step:** The ML Model processes the cleaned message, classifying it as either spam or not spam.
6. **Return Prediction:** The result of the classification is sent back to the User Interface.
7. **Display Result:** Finally, the User Interface displays the classification result to the user, indicating whether the message is spam or not spam.

4.3 DATA SET DESCRIPTIONS

The dataset consists of a collection of messages labeled as spam or ham (not spam). Each message is a labeled instance, enabling supervised learning for the classification model.



- **Attributes**:
    - **Label**: Indicates whether the message is spam (1) or ham (0).
    - **Message**: The content of the message to be classified.
- **Dataset Details**:
    - **Total Records**: 5,572
    - **Classes**: 0 (ham), 1 (spam)
    - **Distribution**: Approximately 13.4% spam messages and 86.6% ham messages.

4.4 DATA PREPROCESSING TECHNIQUES

Preprocessing is critical in preparing the text data for feature extraction and modeling. The techniques applied are:

1. **Lowercasing**: Converts all text to lowercase to avoid case sensitivity issues.
2. **Tokenization**: Breaks down the text into individual words, or tokens, which are analyzed separately.
3. **Stop-word Removal**: Removes common words (like "and," "is") that don't contribute to the context of the message.
4. **Lemmatization**: Reduces words to their base form, simplifying variations in word forms (e.g., "running" becomes "run").
5. **Special Character Removal**: Removes non-alphabetic characters, such as punctuation, to avoid unnecessary noise in the data.

4.5 METHODS & ALGORITHMS

The primary techniques and algorithms employed in the spam classification model include:

1. **Feature Extraction (TF-IDF)**:
    - Term Frequency-Inverse Document Frequency (TF-IDF) is used to convert the processed text into numerical features. It quantifies word relevance by considering how often a term appears in a message compared to its occurrence across all messages. This provides a structured, meaningful representation of text data for model input.
2. **Classification Algorithm (Support Vector Machine - SVM)**:
    - SVM is a powerful supervised learning algorithm, effective for binary classification tasks. The classifier finds an optimal hyperplane to separate spam and non-spam messages in the feature space. It performs well with high-dimensional data and can handle the sparse TF-IDF matrix effectively.
3. **Hyperparameter Tuning (GridSearchCV)**:
    - GridSearchCV is used to fine-tune model parameters (like kernel type, regularization parameter C, and gamma) by testing combinations and selecting the ones that yield the best performance based on cross-validation results.

5. DEPLOYMENT AND RESULTS

5.1 INTRODUCTION

This section covers the deployment, evaluation, and outcomes of the spam classification system. It provides insights into the system's implementation, testing, and validation stages and discusses the development of a user-friendly web interface to facilitate interaction with the classification model.

5.2 SOURCE CODE

This section includes the core source code used for the project, including scripts for data preprocessing, model training, and prediction. Code snippets demonstrate key aspects like text processing, feature extraction with TF-IDF, SVM model training, and Flask web application routing for handling user inputs and outputs.

Code:

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer

from    sklearn.model_selection    import    train_test_split,    GridSearchCV,
cross_val_score

from sklearn.svm import SVC

from sklearn.metrics import classification_report, confusion_matrix

import pickle


# Load and preprocess the dataset

dataset = pd.read_csv('spam.csv', encoding='latin-1')

dataset.columns = ['label', 'message']
```

```
dataset['label'] = dataset['label'].map({'ham': 0, 'spam': 1})
# Preprocess text: remove special characters, stopwords, and lemmatize
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
corpus = [
    ' '.join([lemmatizer.lemmatize(word) for word in re.sub('[^a-zA-Z]', ' ', msg).lower().split() if word not in stop_words])
    for msg in dataset['message']
]


# Vectorize text with TF-IDF and split data
tfidf = TfidfVectorizer(max_features=5000)
X, y = tfidf.fit_transform(corpus).toarray(), dataset['label'].values
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


# Train model using GridSearchCV for SVM tuning
param_grid = {'C': [0.1, 1, 10], 'gamma': ['scale', 'auto'], 'kernel': ['linear']}
grid_search = GridSearchCV(SVC(), param_grid, cv=3, scoring='accuracy').fit(x_train, y_train)


# Model evaluation and cross-validation
y_pred = grid_search.predict(x_test)
print(classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
cv_scores = cross_val_score(grid_search, X, y, cv=5)
print("Mean CV Score:", cv_scores.mean())


pickle.dump(grid_search, open('final_model.pkl', 'wb'))
pickle.dump(tfidf, open('final_vector.pkl', 'wb'))
```

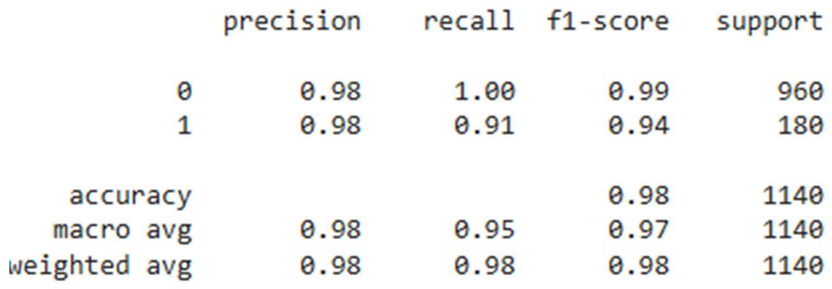
5.3 MODEL IMPLEMENTATION AND TRAINING

The spam classifier model is implemented using an SVM algorithm, designed to distinguish between spam and non-spam messages. This section details the training process:

- **Dataset Loading and Preprocessing:** Preparing the labeled messages.
- **Feature Extraction with TF-IDF:** Converting text into numerical data suitable for model input.
- **Model Training:** Using the processed data to train the SVM classifier, applying parameter tuning techniques to enhance model performance.
- **Model Persistence:** Saving the trained model and vectorizer as .pkl files to enable deployment.

5.4 MODEL EVALUATION METRICS

Evaluation metrics assess the model's effectiveness in identifying spam messages accurately. Metrics include:



```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       960
           1       0.98      0.91      0.94       180

    accuracy                           0.98      1140
   macro avg       0.98      0.95      0.97      1140
weighted avg       0.98      0.98      0.98      1140
```

Fig 5.4: Evaluation Metrics

- **Accuracy:** The percentage of correctly classified messages.
- **Precision:** The proportion of actual spam messages among those classified as spam.
- **Recall:** The model's ability to identify spam messages.
- **F1 Score:** A balance of precision and recall, representing the model's overall performance.

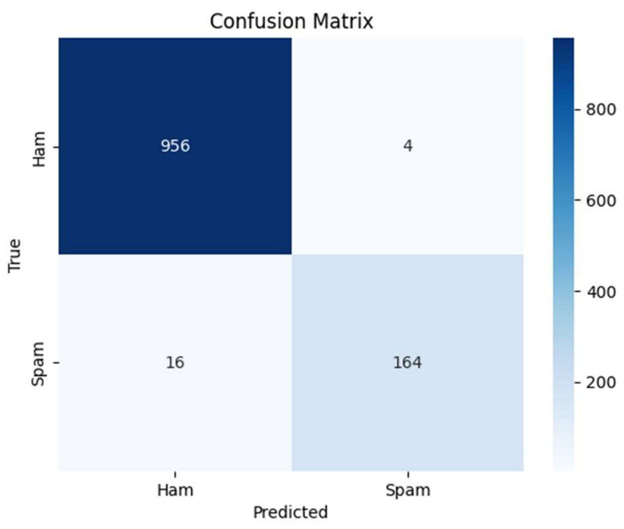
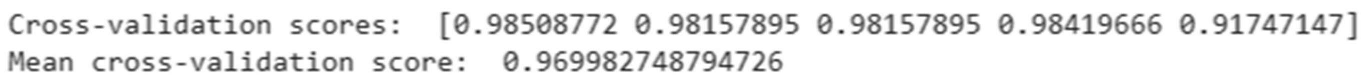


Fig 5.4: Confusion Matrix

- **Cross-Validation Scores:** Highlighting consistency across multiple data subsets, reducing overfitting risks.



```
Cross-validation scores:  [0.98508772 0.98157895 0.98157895 0.98419666 0.91747147]
Mean cross-validation score:  0.969982748794726
```

5.5 MODEL DEPLOYMENT: TESTING AND VALIDATION

This section focuses on the deployment and validation of the trained model:

- **Testing:** The model is tested on unseen messages to gauge real-world performance.
- **Validation:** Additional evaluation ensures that the model maintains accuracy across diverse message types.
- **Error Analysis:** Reviewing misclassified instances to identify potential areas for improvement in model accuracy.

5.6 WEB GUI’S DEVELOPMENT

A Flask-based Web GUI was developed to provide a user-friendly interface for interacting with the spam classifier:

Flask Web Application for Prediction:

Code:

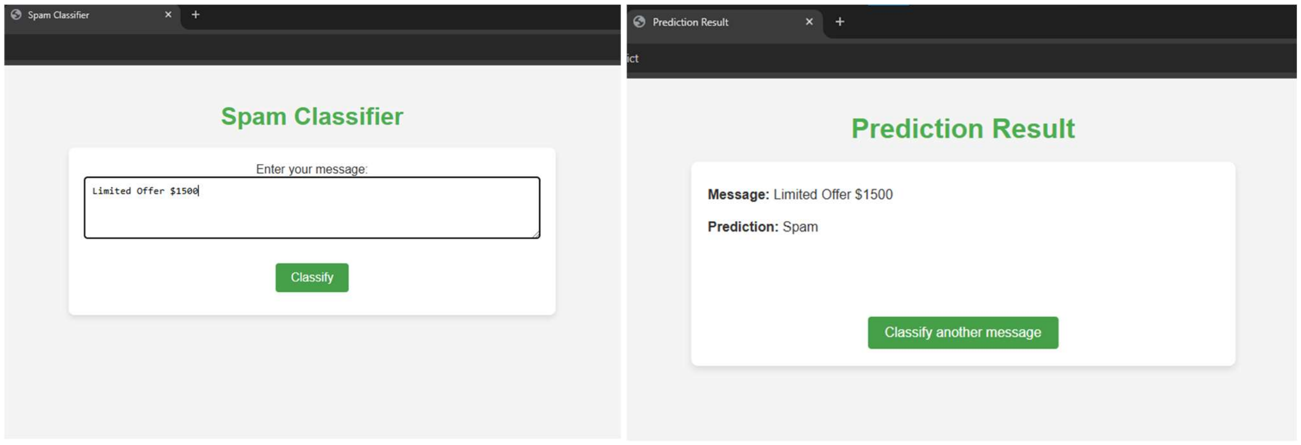
```
from flask import Flask, render_template, request
import pickle


app = Flask(__name__)


# Load model and vectorizer
classifier = pickle.load(open('final_model.pkl', 'rb'))
tfidf = pickle.load(open('final_vector.pkl', 'rb'))
# Prediction function
def predict_message(message):
    transformed = tfidf.transform([message])
    return "Spam" if classifier.predict(transformed)[0] == 1 else "Not Spam"
# Define routes
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    message = request.form['message']
    result = predict_message(message)
    return render_template('result.html', message=message, result=result)


if __name__ == '__main__':
    app.run(debug=True)
```

- **Home Page (index.html):** Allows users to input messages for classification.
- **Prediction Page (result.html):** Displays results (spam or not spam) based on model predictions.
- **Routes and Functions:** Flask routes handle data input from the user, process it through the model, and return the prediction.

5.7 RESULTS

The results section provides an analysis of the model's performance on the test set, showcasing its ability to accurately classify spam messages. Key highlights include:

- **Sample Predictions:** Screenshots or examples of input messages with their respective classification outputs.



Spam Classifier

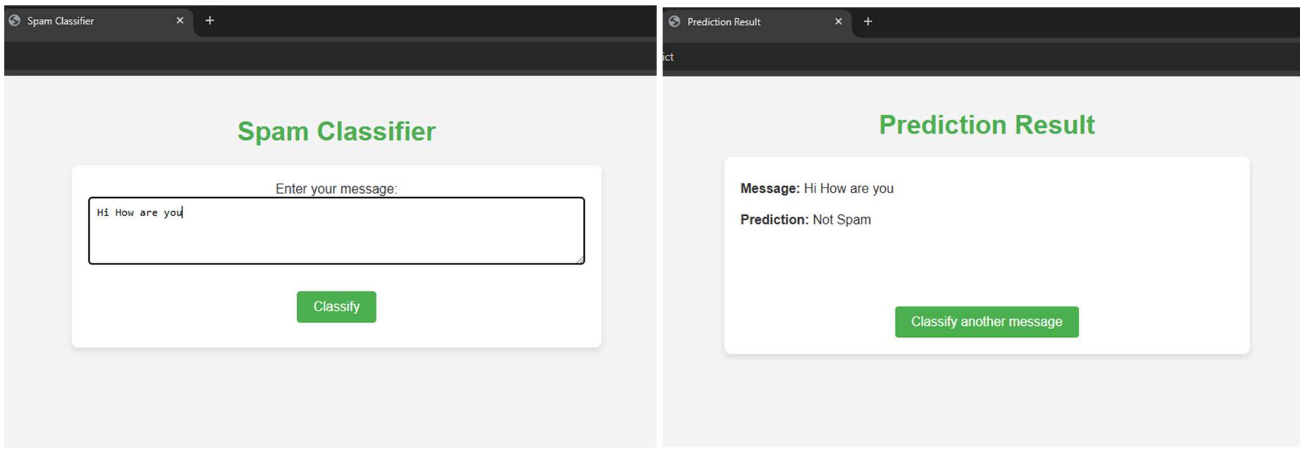
Enter your message: Limited Offer $1500

Prediction Result

Message: Limited Offer $1500

Prediction: Spam

Fig 5.7.1 : Spam Message Identification



Spam Classifier

Enter your message: Hi How are you

Prediction Result

Message: Hi How are you

Prediction: Not Spam

Fig 5.7.2 :Not Spam Message Identification

6. CONCLUSION

6.1 PROJECT CONCLUSION

The Spam Detection System developed in this project demonstrates the potential of machine learning and Natural Language Processing (NLP) in effectively filtering spam messages. By leveraging Support Vector Machine (SVM) classification and TF-IDF vectorization, this model can reliably classify text messages as spam or non-spam with high accuracy. This system addresses the problem of unwanted spam communications, providing a scalable solution that can be easily integrated into applications and services, reducing user exposure to irrelevant and potentially harmful content. The deployed web application allows end-users to conveniently test message inputs, offering real-time spam classification.

6.2 FUTURE SCOPE

This project provides a foundation for further enhancements, such as:

- Expanding the dataset to improve accuracy across diverse spam formats and languages.
- Incorporating deep learning models, such as recurrent neural networks (RNN) or transformers, for improved performance with larger datasets.
- Developing an adaptive model that evolves with changing spam patterns, using online learning methods.
- Enhancing the application interface and deploying it as a service for broader use, potentially integrating with email or messaging platforms for real-time spam filtering.

REFERENCES

1. *G. James, D. Witten, T. Hastie, R. Tibshirani*, "An Introduction to Statistical Learning," Springer, 2013, pp. 337-384. https://www.statlearning.com
2. *S. Raschka*, "Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2," Packt Publishing, 2019, pp. 219-243. https://www.packtpub.com
3. *NLTK Documentation*, Natural Language Toolkit, https://www.nltk.org/
4. *Scikit-learn Documentation*, scikit-learn: Machine Learning in Python, https://scikit-learn.org/
5. *Flask Documentation*, Flask Web Development, https://flask.palletsprojects.com/