**EMAIL SPAM CLASSIFICATION USING NAÏVE BAYES**

**A PROJECT REPORT**

**for**

**Introduction to AI (****AI101B)**

**Session (2024-25)**

**Submitted by**

**RIDHI RAJPUT**

**(202410116100165)**

**SANKET PUNDHIR**

**(202410116100183)**

**NAMRTA SINGH**

**(202410116100129)**

**Submitted in partial fulfilment of the**

**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

### Ms. Komal Salgotra

### Assistant Professor



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**(APRIL- 2025)**

**CERTIFICATE**

Certified that **Ridhi Rajput** **(202410116100165), Sanket Pundhir (202410116100183), Namrta Singh (202410116100129)** have carried out the project work having “**Email Spam Classification”** (**Introduction to AI, ID201B**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Ms. Komal Salgotra Dr. Akash Rajak**

**Assistant Professor Dean**

**Department of Computer Applications Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

**ii**

**ABSTRACT**

In today's digital era, email remains one of the most widely used means of communication. However, the increasing volume of spam emails poses a serious threat to users’ security and productivity. This project presents an Email Spam Classifier built using Naive Bayes, a popular machine learning algorithm for text classification, to automatically detect and filter out spam messages.

The system is integrated with a user-friendly web interface developed using Flask. Users can input an email message, and the classifier will predict whether the message is Spam or Not Spam. The frontend includes interactive design elements such as animated RGB text, video background, and glowing result boxes, enhancing the user experience.

The project effectively demonstrates the application of machine learning in real-world scenarios, emphasizing the importance of natural language processing and model deployment in cybersecurity solutions. It provides a scalable foundation for further enhancements such as advanced NLP techniques and real-time email filtering.

Keywords: Email spam, Naïve Bayes, Web Application, Python

**iii**

**ACKNOWLEDGEMENTS**

Success in life is never attained single-handedly. My deepest gratitude goes to our project supervisor, **Ms. Komal Salgotra** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express our gratitude to Dr. Akash Rajak, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, we have many understanding friends, who have helped us a lot on many critical conditions.

Finally, our sincere thanks go to our family members and all those who have directly and indirectly provided us with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep our life filled with enjoyment and happiness.

**Ridhi Rajput**

**Sanket Pundhir**

**Namrta Singh**

**iv**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| Certificate | ii |
| Abstract | iii |
| Acknowledgement | iv |
| Table of Contents | v |
| 1. Introduction  1.1 Background  1.2 Problem Statement  1.3 Objectives of the project  1.4 Scope of the project  1.5 Significance | 6-8  6  6  7  7  7 |
| 2. Methodology | 9-10 |
| 3. Code  4. Result | 11-15  16-17 |
| References | 18 |

**v**

**INTRODUCTION**

* 1. **Background**

In the digital age, email is one of the most widely used communication tools. However, with its popularity comes a significant drawback — the constant influx of spam emails. These unwanted messages not only clutter users' inboxes but also pose threats like phishing attacks, scams, and malware dissemination. Traditional filtering mechanisms often fall short, leading to the development of automated spam classification systems using machine learning.

* 1. **Problem Statement**

In today’s digitally connected world, email has become an essential mode of communication for both personal and professional purposes. However, the effectiveness of this communication channel is heavily compromised by the exponential rise in spam emails. These unsolicited messages not only clutter inboxes but often carry malicious intent — from phishing scams to the distribution of harmful software.

Despite the existence of traditional rule-based spam filters, many of them either misclassify genuine emails as spam (false positives) or fail to detect new spam patterns (false negatives). This creates a pressing need for a more intelligent, adaptive, and data-driven solution that can evolve with the changing nature of spam tactics.

The main problem addressed in this project is the automatic classification of email messages into "Spam" or "Not Spam" using a machine learning-based approach that ensures higher accuracy, adaptability, and user-friendly accessibility through a web-based interface.

* 1. **Objectives of the Project**

The primary objective of this project is to develop an intelligent email spam classifier that can effectively differentiate between spam and non-spam (ham) emails using Machine Learning, specifically the Naive Bayes algorithm. The project also aims to present this functionality through an interactive and visually appealing web interface that is accessible to end users.

The specific objectives include:

1. Data Preprocessing & Model Training

* Clean and preprocess textual email data.
* Train a Naive Bayes classification model for spam detection using a labelled dataset.

2. Efficient Text Classification

* Convert email text into numerical vectors using feature extraction techniques (e.g., CountVectorizer or TF-IDF).
* Classify incoming messages with high accuracy based on learned patterns.

6

3. Frontend and Backend Integration

* Build a responsive and visually attractive UI using HTML and CSS.
* Use Flask (Python) for server-side logic and integration between the model and the frontend.

4. Real-Time Prediction and Feedback

* Provide users with real-time feedback on whether the email is spam or not.
* Display visual cues like icons and styled result boxes to enhance user experience.

5. User Experience and Accessibility

* Incorporate modern design features such as background videos, glowing animations, and responsive layouts to ensure the app is both functional and engaging.
  1. **Scope of the Project**

This project demonstrates how machine learning can be applied in the cybersecurity domain, particularly for personal and enterprise email filtering. In future iterations, this classifier can be integrated into real-time email systems, expanded using deep learning, and trained on larger, multilingual datasets for broader applicability.

* 1. **Significance**

In today’s digital era, email remains one of the most widely used forms of communication, both personally and professionally. However, with the increasing volume of emails exchanged daily, the prevalence of spam messages has also grown significantly. These unsolicited emails not only waste users' time but may also carry malicious content such as phishing links, malware, or deceptive advertisements, posing serious security and privacy risks.

This project is significant for the following reasons:

1. **Enhanced Email Security**

The system helps safeguard users by accurately identifying and filtering out harmful or irrelevant messages, reducing the risk of falling victim to phishing or scam attempts.

1. **Time and Productivity Optimization**

By classifying spam messages automatically, users can focus on important emails, thereby saving time and improving their overall efficiency.

1. **Scalability and Real-World Application**

The Naive Bayes algorithm used in this project is computationally efficient and can be scaled for real-world applications such as email service providers, corporate mail servers, and spam filters in messaging apps.

7

1. **Educational Value and Practical Learning**

This project serves as a practical example of how machine learning, natural language processing (NLP), and web development can be combined to solve real-world problems. It strengthens understanding of concepts like data preprocessing, model training, deployment, and user interface design.

1. **User-Friendly Deployment**

The project’s intuitive and modern user interface makes it accessible to both technical and non-technical users, showcasing the importance of good design in machine learning applications.

8

**METHODOLOGY**

The development of the Email Spam Classifier involved several well-structured phases, from data acquisition and preprocessing to model training and web integration. The following methodology outlines the step-by-step approach taken in building this project:

**1. Data Collection**

* A publicly available dataset containing labeled email messages (as spam or not spam) was used.
* Each record in the dataset had a message and its corresponding label.
* This dataset provided the foundation for training and evaluating the machine learning model.

**2. Data Preprocessing**

Before feeding data into the model, preprocessing was essential to clean and standardize the input text:

* **Lowercasing:** Converted all text to lowercase to maintain uniformity.
* **Removing special characters and punctuation:** Non-alphabetic characters were removed to eliminate noise.
* **Stopword removal**: Common English words (e.g., "the", "is", "in") that don’t add much meaning were filtered out.
* **Tokenization:** The text was split into individual words or tokens.
* **Stemming:** Words were reduced to their base/root form (e.g., “running” → “run”).

This preprocessing ensured that only meaningful features were passed to the model.

**3. Feature Extraction**

* TF-IDF (Term Frequency-Inverse Document Frequency) vectorization was used to convert text data into numerical format.
* Each email message was represented as a vector based on the importance of each word in the corpus.
* This step transformed unstructured data into a structured form that the model could understand.

**4. Model Selection and Training**

* The Naive Bayes classifier, particularly Multinomial Naive Bayes, was chosen due to its effectiveness in text classification tasks.
* The dataset was split into training and testing sets (e.g., 80% for training, 20% for testing).
* The model was trained on the training set using the vectorized features.
* Accuracy, precision, recall, and F1-score were used to evaluate model performance on the test set.

**5. Model Serialization**

* Once trained and tested, the model and the TF-IDF vectorizer were serialized using Python’s pickle library.
* This allowed the model to be reused without retraining each time the application runs.

**6. Web Application Development**

* The frontend was designed using HTML and CSS to create a clean and user-friendly interface.
* A background video and glowing elements were added for visual appeal.
* The backend was built using Flask, a lightweight Python web framework.

Features:

* Users can paste email text into a form on the web page.
* Upon clicking “Check,” the message is sent to the backend using a POST request.
* The backend loads the model and vectorizer, processes the input, predicts the label (Spam or Not Spam), and returns the result.

**7. User Interface Enhancements**

* Added a responsive layout with animations and effects.
* Included background video and images (e.g., shield for Ham, ad symbol for Spam).
* Result section styled with glowing borders and color-coded backgrounds for clarity.

**8. Testing and Deployment**

* Thorough testing was done to ensure proper integration of frontend, backend, and machine learning components.
* The app was run locally using Flask.
* Model predictions and user interface responses were verified for different email types

10

**CODE**

**Python 1st file**

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

import pickle

# Load dataset

df = pd.read\_csv("https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv", sep='\t', names=["label", "message"])

df['label'] = df['label'].map({'ham': 0, 'spam': 1})

# Features and labels

X = df['message']

y = df['label']

# Vectorize text

vectorizer = CountVectorizer()

X\_vectorized = vectorizer.fit\_transform(X)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_vectorized, y, test\_size=0.2, random\_state=42)

# Train model

model = MultinomialNB()

model.fit(X\_train, y\_train)

# Save model and vectorizer

with open('spam\_classifier.pkl', 'wb') as f:

    pickle.dump(model, f)

with open('vectorizer.pkl', 'wb') as f:

    pickle.dump(vectorizer, f)

**Python 2nd file**

from flask import Flask, render\_template, request

import pickle

app = Flask(\_\_name\_\_)

# Load model and vectorizer

with open('spam\_classifier.pkl', 'rb') as f:

    model = pickle.load(f)

with open('vectorizer.pkl', 'rb') as f: 11

    vectorizer = pickle.load(f)

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

    message = request.form.get('message')

        if not message or not message.strip():

        return render\_template('index.html', prediction=" No message entered.", original="")

    vect = vectorizer.transform([message])

    prediction = model.predict(vect)[0]

    result = "Spam" if prediction == 1 else "Not Spam"

    return render\_template('index.html', prediction=result, original=message)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

**HTML File**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Email Spam Classifier</title>

    <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <!-- Background Video -->

    <video autoplay muted loop id="bgVideo">

        <source src="https://videos.pexels.com/video-files/5473967/5473967-uhd\_2732\_1440\_25fps.mp4" type="video/mp4">

        Your browser does not support the video tag.

    </video>

    <div class="container">

        <!-- Optional Icon -->

        <img src="{{ url\_for('static', filename='images/email.png') }}" class="icon" alt="Mail Icon">

        <h1>Email Spam Classifier</h1>

        <form action="/predict" method="post">

            <textarea name="message" placeholder="Type or paste your email message here..."></textarea>

            <button type="submit">Check</button>

        </form>

        {% if prediction %}

    <div class="result {% if prediction == 'Spam' %}spam{% else %}ham{% endif %}">

        Prediction: <strong>{{ prediction }}</strong><br>

12

        {% if prediction == "Spam" %}

            <img src="{{ url\_for('static', filename='images/ad.png') }}" class="result-img" alt="Spam">

        {% else %}

            <img src="{{ url\_for('static', filename='images/shield.png') }}" class="result-img" alt="Ham">

        {% endif %}

    </div>

{% endif %}

    </div>

</body>

</html>

**CSS File**

#bgVideo {

    position: fixed;

    top: 0;

    right: 0;

    width: 100%;

    height: 100%;

    object-fit: cover;

    z-index: -1;

    filter: brightness(0.6);

}

body {

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    margin: 0;

    padding: 0;

    display: flex;

    flex-direction: row;

    height: 100vh;

    overflow: hidden;

}

.container {

    position: absolute;

    top: 50%;

    left: 15%;

    transform: translate(-25%, -50%);

    background: rgba(255, 255, 255, 0.2);

    backdrop-filter: blur(12px);

    -webkit-backdrop-filter: blur(10px);

    padding: 30px;

    border-radius: 16px;

    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.2);

    width: 35%;

    text-align: center;

13

    z-index: 10;

    color: #000;

}

.icon {

    width: 60px;

    margin-bottom: 1px;

}

textarea {

    width: 85%;

    height: 150px;

    padding: 12px;

    font-size: 16px;

    border-radius: 8px;

    border: 1px solid #ccc;

    resize: none;

    margin-bottom: 20px;

}

button {

    background-color: #007bff;

    color: white;

    border: none;

    padding: 12px 24px;

    font-size: 16px;

    border-radius: 8px;

    cursor: pointer;

    transition: background-color 0.3s ease;

}

button:hover {

    background-color: #0056b3;

}

.result {

    margin-top: 20px;

    font-size: 16px;

    padding: 10px;

    border-radius: 6px;

    font-weight: bold;

    width: 80%;

    margin-left: auto;

    margin-right: auto;

}

.result.spam {

    background-color: #ffe5e5;

    color: #d60000;

14

    box-shadow: 0 0 12px #ff4d4d, 0 0 20px #ff1a1a;

    border: 2px solid #ff1a1a;

    animation: pulseGlowRed 2s infinite;

        padding: 7px;

    font-size: 1.4rem;

    width: 70%;

    max-width: 600px;

    margin: 20px auto;

    border-radius: 15px;

    text-align: center;

}

.result.ham {

    background-color: #e6ffe6;

    color: #007f00;

    box-shadow: 0 0 12px #66ff66, 0 0 20px #33cc33;

    border: 2px solid #33cc33;

    animation: pulseGlowGreen 2s infinite;

}

.result-img {

    width: 80px;

    margin-top: 8px;

    border-radius: 8px;

}

@keyframes pulseGlowRed {

    0% { box-shadow: 0 0 12px #ff4d4d, 0 0 20px #ff1a1a; }

    50% { box-shadow: 0 0 20px #ff1a1a, 0 0 30px #ff0000; }

    100% { box-shadow: 0 0 12px #ff4d4d, 0 0 20px #ff1a1a; }

}

@keyframes pulseGlowGreen {

    0% { box-shadow: 0 0 12px #66ff66, 0 0 20px #33cc33; }

    50% { box-shadow: 0 0 20px #33cc33, 0 0 30px #00cc00; }

    100% { box-shadow: 0 0 12px #66ff66, 0 0 20px #33cc33; }

}

h1 {

        color: white;

}

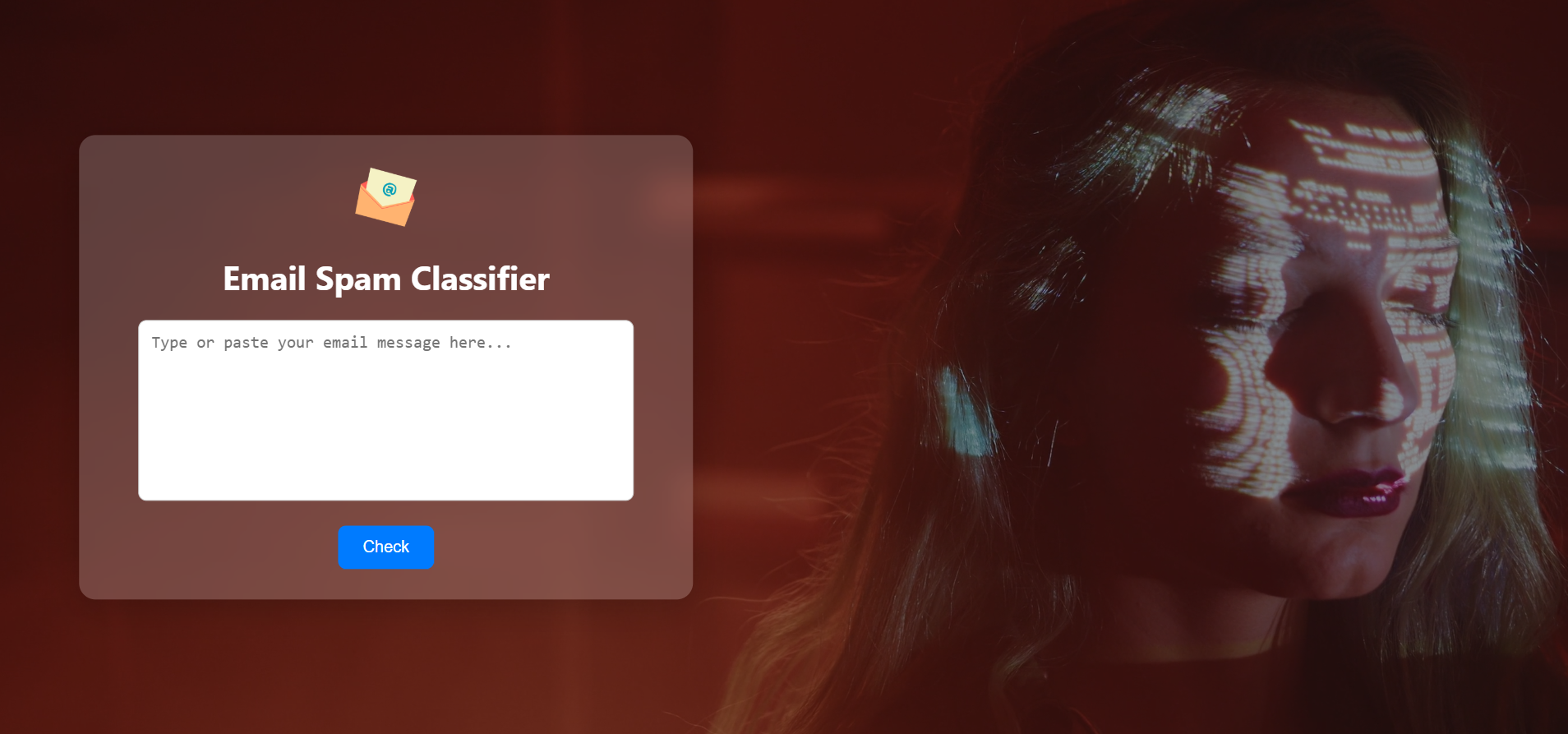
15

**RESULT**

The Fig. 1 displays the final user interface of the Email Spam Classifier web application. The interface is designed with simplicity and interactivity in mind, allowing users to easily input any email message and check whether it is classified as Spam or Not Spam.

* The interface features a clean layout with a central input box where users can type or paste an email.
* A “Check” button below the input triggers the prediction process using the machine learning model hosted on the backend.
* Based on the prediction, the result is displayed dynamically on the same page with a visual icon and background styling that clearly distinguishes between spam and non-spam messages.

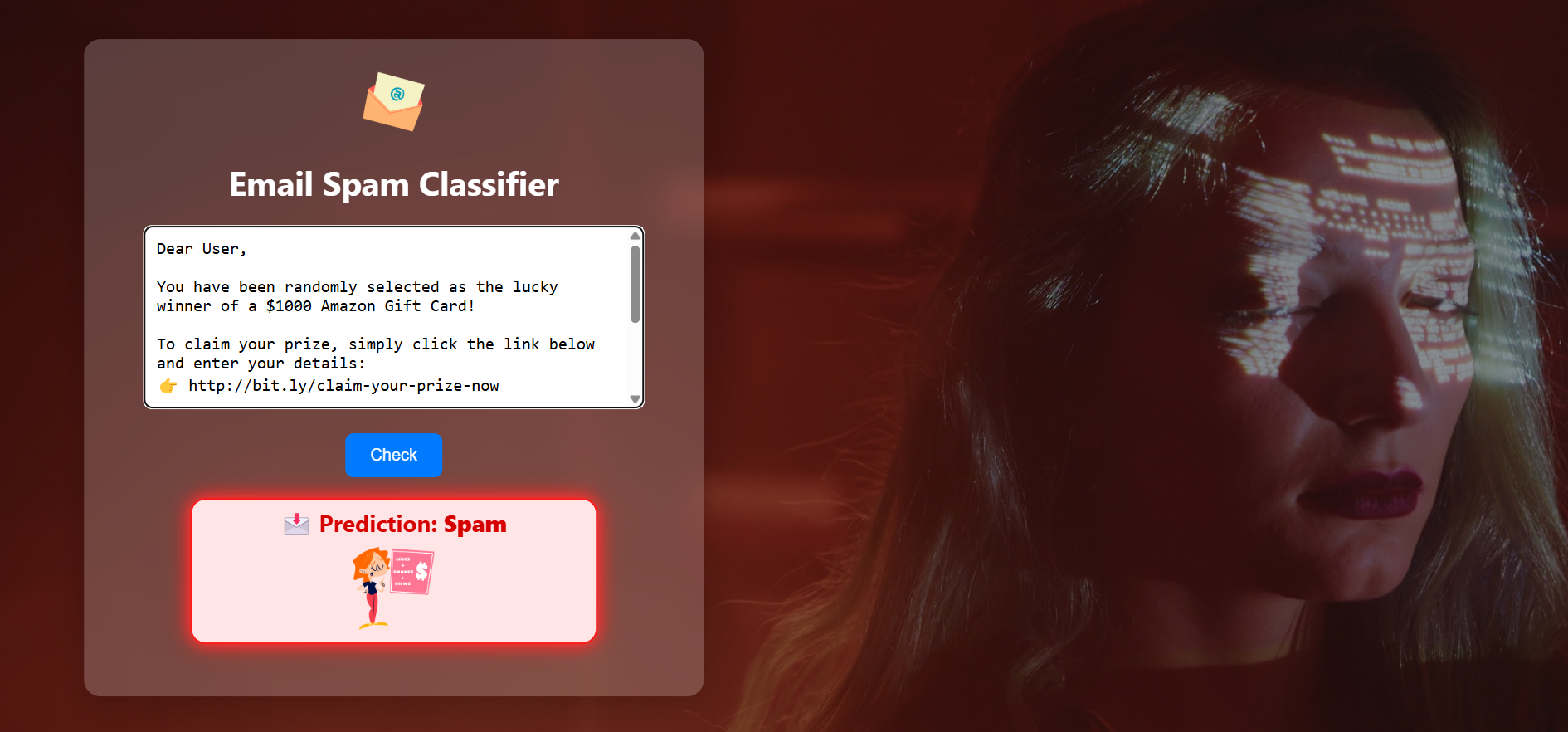
This user interface reflects the core functionality and aesthetic quality of the project, showcasing how machine learning can be made accessible and visually engaging for real-world users.

****

**Fig. 1**

16

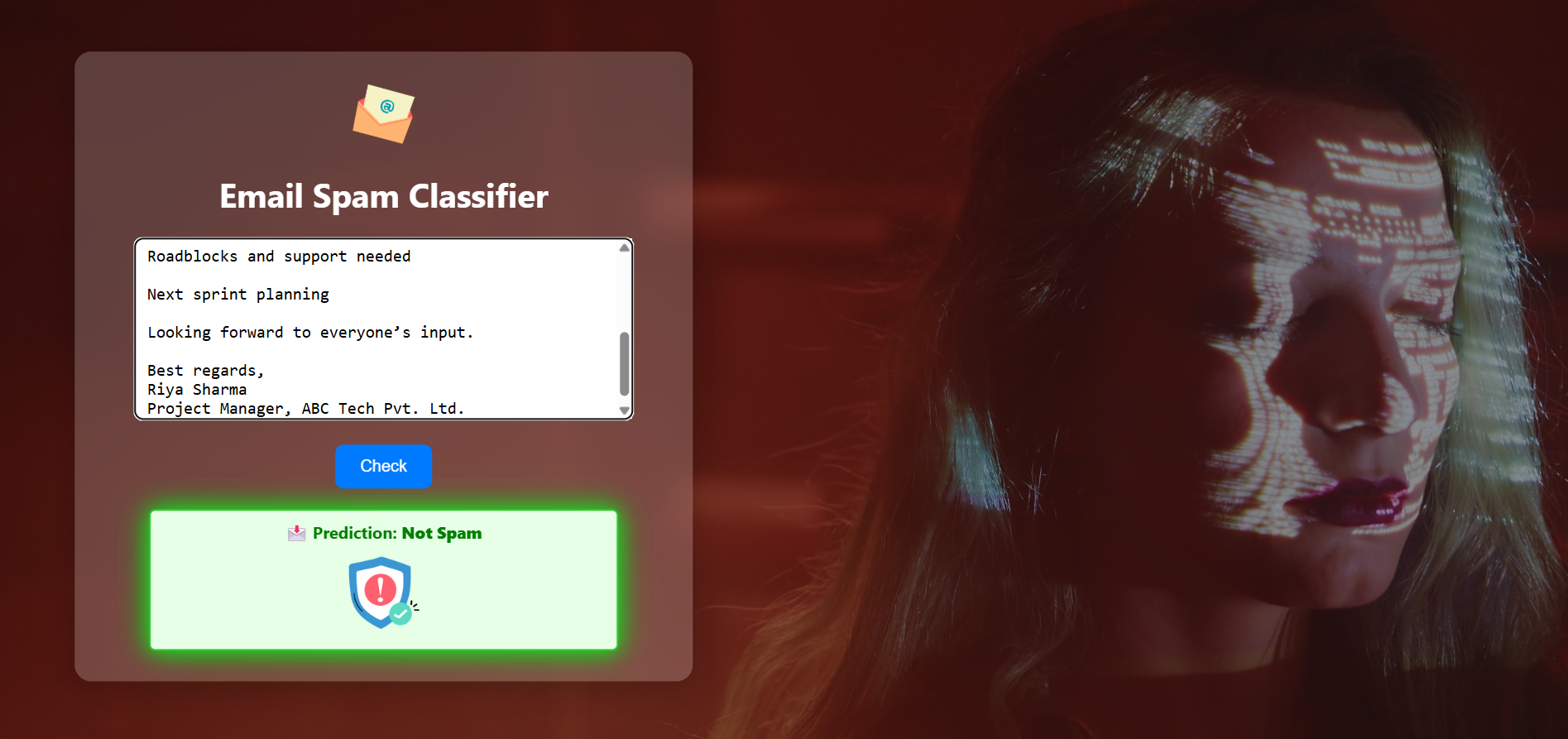
This Fig. 2 display output screen effectively demonstrates that the application can detect and alert users to potentially harmful or unsolicited messages.

****

**Fig. 2**

After analyzing the input using the Naive Bayes classification algorithm, the system successfully identifies the email as “Not Spam”. This shows that the classifier accurately distinguishes between genuine work-related communication and potential spam, based on the absence of spam-triggering keywords, urgency, or suspicious links.

This Fig. 3 result confirms that the model is capable of recognizing professional and meaningful content as safe and trustworthy.

****

**Fig. 3**

17

**REFERENCES**

1. Scikit-learn: Machine Learning in Python

https://scikit-learn.org/

2. Natural Language Toolkit (NLTK)

https://www.nltk.org/

3. SpamAssassin Public Corpus – Dataset used for spam/ham classification.

https://spamassassin.apache.org/publiccorpus/

4. Python Official Documentation

https://docs.python.org/3/

5. Flask – Web Framework Documentation

https://flask.palletsprojects.com/

6. Pickle Module Documentation – Python Object Serialization

https://docs.python.org/3/library/pickle.html

7. TF-IDF Explanation – Towards Data Science

https://towardsdatascience.com/understanding-tf-idf-5a3f9cf4b16d

8. Freepik – Free Stock Videos and Images (used for background video)

<https://www.freepik.com/videos/>

18