

TEAM MEMBERS

SUDHAGAR.P

PRIYANGA.S

KOWSALAYA.S

RAJALAKSHMI.P

MENTOR’S NAME

RAJKUMAR.S (AP\ECE)

Building a Smarter AI-  
Powered Spam Classifier

ABSTRACT:

Develop an AI-powered spam classifier using natural language processing (NLP) and machine learning Techniques to accurately distinguish between spam and non-spam messages in emails or text messages

Problem statement

**Objectives:**

* Achieve a high level of accuracy in classifying emails, minimizing both false positives and false negatives.
* Adapt to evolving spamming techniques through continuous monitoring and updates.
* Handle diverse content types including text, images, links, and attachments.
* Be contextually sensitive to distinguish between known contacts and new senders.
* Effectively address adversarial attacks by identifying and mitigating deliberate evasion tactics.

Deliverables

* A trained and validated AI-powered spam classifier capable of integration into existing email systems.
* Documentation outlining the model architecture, training process, and deployment instructions.
* Evaluation metrics demonstrating the classifier's performance, including accuracy, precision, recall, and F1-score.
* Ongoing monitoring and update mechanism to adapt to new spamming tactics and maintain effectiveness over time

Constraints

* The solution should be resource-efficient to minimize computational overhead in processing a high volume of emails.
* Privacy and data protection regulations must be strictly adhered to in handling email content.
* The solution should be scalable to handle a growing volume of emails without compromising performance.
* The classifier should not introduce significant delays in email delivery or retrieval processes.

design thinking process

Design thinking is a human-centered approach to problem-solving that focuses on empathy, ideation, and prototyping. It can be applied to building a smarter AI-powered spam classifier to ensure that the solution addresses user needs effectively. Here's how the design thinking process can be applied

Empathize

* Understand the users: Begin by conducting interviews or surveys with users (such as email users and administrators) to gain insights into their pain points, preferences, and challenges related to email communication and spam.

Analyze feedback: Review existing spam filter systems to identify their strengths and weaknesses. Gather feedback from users about their experiences with false positives and false negatives

Define

Define the problem statement: Based on user feedback and analysis, clearly define the problem statement. This could involve specifying the desired improvements in spam classification, such as reducing false positives or adapting to evolving spam tactics.

Ideate

* Brainstorm solutions: Gather a multidisciplinary team including data scientists, engineers, and user experience designers to generate a wide range of ideas for building a smarter spam classifier.
* Encourage creativity: Encourage unconventional thinking and explore innovative approaches, such as leveraging cutting-edge machine learning techniques or combining rule-based systems with AI.

Prototype

Develop a Minimum Viable Product (MVP): Create a functional prototype of the spam classifier. This could involve setting up a simplified version of the system with a small dataset for initial testing.

Test with stakeholders: Gather feedback from users, administrators, and other stakeholders to evaluate the prototype's performance and usability.

PROGRAM:

# Import necessary libraries

import numpy as np

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, classification\_report

# Step 1: Data Collection (Assuming you have spam and ham datasets)

# Load and pre-process your data. This could involve reading files, cleaning text, etc.

# Step 2: Data Pre-processing (Assuming you have a list of emails in 'emails' and their corresponding labels in 'labels')

# Perform tokenization, stop word removal, etc.

# This step is crucial but can be complex, and might involve more advanced techniques depending on your data.

# Step 3: Feature Extraction (Using TF-IDF)

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000) # Adjust max\_features based on your dataset

X = tfidf\_vectorizer.fit\_transform(emails)

# Step 4: Model Selection and Training

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, labels, test\_size=0.2, random-state=42)

classifier = Multinomia lNB()

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

# Step 5: Evaluation

y\_pred = classifier.predict(X-test)

accuracy = accuracy\_score(y\_test, y pred)

report = classification\_report(y\_test, y\_pred)

printf(“Accuracy: {accuracy}")

printf(“Classification Report:\n{report}")

# Step 6: Hyper parameter Tuning (Optional)

# Experiment with different hyper parameters, or try different models to improve performance.

# Step 7: Model Testing (On a separate test set)

# Step 8: Deployment

# Integrate the trained model into your email client or server.

# Step 9: Monitoring and Maintenance

# Keep an eye on the model's performance in production and retrain as needed.

# Note: This is a simplified example. Depending on your specific use case, you may need more advanced techniques and libraries.

FLOW CHART:

START

Split the data into training and testing sets

Train the chosen model on the training data

STEP 4: TRAINING

Choose a machine learning algorithm

STEP 3: MODEL SELECTION

Clean and preprocess the text data

Convert text to numerical Features (e.g., TF-IDF word embedding

STEP 2:DATA PRE PROCESSING

Gather a diverse dataset of labeled emails (spam or span spam)

STEP 1:DATA COLLECTION

STEP 5: MODEL EVALUATION

END

Integrate the trained model into a production environment for real time batch classification

Step 7: DEPLOYMENT

Explore techniques like hyper parameter tuning, cross validation

Consider using more advanced models are ensemble techniques

STTEP 6: MODEL IMPROVEMENT

Use the trained model to make predictions on the text set

Evaluate the models performance