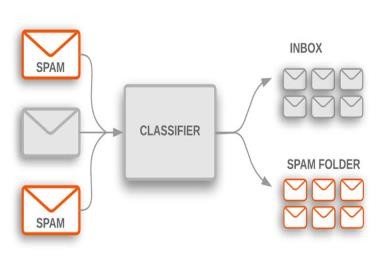
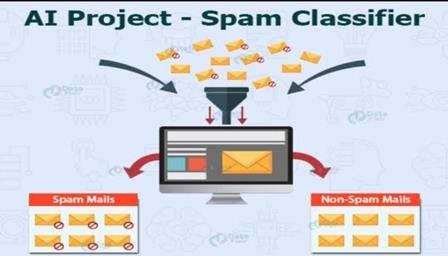
### BUILDING A SMATER AI-POWEREDSPAMCLASSIFIER



**TEAM MEMBER**

### 210221104007 : DIVYA M PHASE 5 :- DOCUMENTATION



**THE GOAL OF THIS PROJECT IS TO DEVELOP A SMATER AI- POWERED SPAM CLASSIFIER THAT CAN EFFECTIVELY DISTINGUSH BETWEEN SPAM AND LEGITIMATE MESSAGES WITH HIGH PRECISION AND RECALL.**

**DATASET LINK:-**

**(**[**https://www.kaggle.com/datasets/uciml/sms-spam-collection-**](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)[**dataset**](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)**)**

**The above is the dataset link used for this project**

**(spam\_classifier.py) : It is a python file that contains the Python script that contains the code for data preprocessing, feature extraction, model training, and evaluation.**

Page **1** of **28**



# OBJECTIVE:-

The objective of building a Smater AI-Powered Spam Classifier project is to enchance the accuracy and efficiency of spam detection to reduce false positives and negatives to improve Spam detection and create a system capable of processing incomming messages or e-mails in real time to prevent spam from reaching user’s inboxes for real time processing. And develop a classifier that can adapt and learn from new spam patterns and tactics used by

spammmer’s for Adapability and allow user’s to customize their spam filtter’s to suite their preferences and needs for user

customization and to explore the incorporation of various data type’s like text images or Url’s for more robust spam detection for multi model classification and optimize the AI model for resources efficience, so it can run effectively even on devices with limited computing power for efficient resource usage and to make privacy and security,integration,evaluation,metrics,user education,sealavility,continuous improvement, compliance and cost effectiveness by adding these objectives, we can create a smater AI- Powered Spam Classifier that effectively identifies and filter’s out unwanted message while minimizing disruptions to legitimate communication.

# PROBLEM STATEMENT:-

In the age of digital communication, email and messaging platforms have became essential tools for personal and professional communication. However, the proliferation of spam

Page **2** of **28**

messages has became a significant nuisance, causing inconvenience



and potentially leading to security risks. Conventional spam filter’s opten fall short in accurately identifying and filtering out spam, resulting in genuine messages being classified as spam or voice versa.

**THE PRIMARY CHALLENGES INCLUDE’S:-**

* **INCREASING ACCURACY**
* **REDUCING FALSE POSTIVIES**
* **HANDLING MULTI MODEL DATA**
* **REAL TIME PROCESSING**
* **SCALABILITY AND GENERALIZATION**
* **USER FRIENDLY INTEGRATION PROBLEM DEFINITION:-**

The main thing is to determine the scope of your spam classifier. Are you focusing on email spam, SMS spam, Comment spam, or a combination? Each may have unique characteristics and challenges to identify the scope and

* **DEFINE OBJECTIVES:-**

Clearly state the project’s objectives for example, reducing false positives, achiving a certain accuracy threshold, or handling a specific volume of messsage per day.

* **USER PERSONA:-**

Create a user persona to understand the needs of the end-

user

(example- e-mail user moderaters, administraters) and their pain point’s related to spam

Page **3** of **28**

# DESIGN THINKING:



* **DATA COLLECTION:-**

We will need a dataset containing labeled examples of spam and non spam messages. We can use a kaggle for this purpose

* **DATA PREPROCESSING:-**

The text dat needs to be cleaned and preprocessed. This involves removing special charcters, converting text to lower case, and tokenizing the text into individual words

* **FUTURE EXTRACTION:-**

We will convert the tokenized words into numerical fetures using techniques like TF-IDF(term frequency:- inverse document frequency)

* **MODEL SELECTION:-**

We can experiment with various meachine learning algorithms such as Naïve Bay’s, support vector machine, and more advanced techiques like Deep learning using Nerual networks.

* **EVALUATION:-**

We will measure the model’s performance using metrics like accuracy, precision, recall, and F1-score.

* **ITERATIVE IMPROVEMENT:-**

We will fine-tune the model and experiment with hyper perameters to improve its accuracy.

Page **4** of **28**

# INNOVATION;



Our project aims to revolutionize spam classification by

developing a highly intelligent AI-powered spam classifier. This document outlines the key innovative approaches and strategies we will employ.

This project aims to push the boundaries of spam detection by introducing several innovative elements that set it apart from existing solutions:

* **Advanced Machine Learning**

**Models**

## Unsupervised Learning

* **Multimodal Analysis**

## Natural Language Understanding

* **Transfer Learning**

## User Feedback Loop

* **Privacy-Preserving Techniques**

## Real-time Analysis

* **Behavioral Analysis**

## \* Human-in-the-Loop AI

Page **5** of **28**

## Ethical Considerations



* **Continuous Learning**

## Advanced Machine Learning Models:

Implement state-of-the-art machine learning models like deep neural networks, transformers, or GPT-based models to improve spam detection accuracy.

## Unsupervised Learning:

Incorporate unsupervised learning techniques to detect new, previously unseen spam patterns by clustering similar messages.

## Multimodal Analysis:

Combine text, image, and voice analysis to identify spam in various forms, such as multimedia messages or voice messages.

## Natural Language Understanding:

Develop a deeper understanding of context and semantics to distinguish between legitimate and spam messages with similar content.

## Transfer Learning:

Page **6** of **28**

Leverage pre-trained AI models and



fine-tune them specifically for spam classification, saving time and resources.

## User Feedback Loop:

Implement a feedback system where users can report false positives and false negatives, allowing the model to continuously improve.

## Privacy-Preserving Techniques:

Ensure user data privacy by using techniques like federated learning or differential privacy when training the classifier.

## Real-time Analysis:

Make the classifier capable of real-time analysis to swiftly detect and block spam messages as they arrive.

## Behavioral Analysis:

Analyze user behavior patterns, like interaction frequency or time of day, to improve the accuracy of spam detection.

## Human-in-the-Loop AI:

Page **7** of **28**

Employ human reviewers to



validate challenging cases, providing feedback to enhance the model's performance.

## Ethical Considerations:

Ensure the AI classifier avoids biases and follows ethical guidelines in its decision- making.

## Continuous Learning:

Implement continuous model updates to adapt to evolving spam tactics and maintain high accuracy.

## A step-by-step innovation algorithm for building a smarter AI-powered spam classifier requires incorporating advanced techniques and approaches. Here's a detailed outline of the algorithm:

**Step 1: Data Collection and Preprocessing**

* + **Collect Diverse Data:** Gather a diverse dataset of spam and legitimate messages across various communication channels (e.g., emails, text messages, multimedia messages).

Page **8** of **28**

* + **Data Cleaning:** Clean and preprocess the dataset by removing duplicates, handling missing values, and standardizing text data. Consider using techniques like stemming or lemmatization.



## Step 2: Feature Engineering

* + **Multimodal Feature Extraction:** Extract features from different modalities, including text, images, and audio. Utilize techniques like TF-IDF for text, image embeddings, and voice recognition for audio.
  + **Semantic Analysis:** Implement semantic analysis to capture the meaning and context of messages, going beyond simple keyword matching.

## Step 3: Advanced Machine Learning Models

* + **Transfer Learning:** Utilize pre-trained AI models (e.g., BERT, ResNet, or custom models) and fine- tune them for spam classification. Transfer learning can enhance model performance.
  + **Ensemble Learning:** Experiment with ensemble methods (e.g., stacking or boosting) to combine

Page **9** of **28**

predictions from multiple models for improved



accuracy.

## Step 4: Real-time Analysis and User Interaction

* + **Real-time Processing :** Enable real-time analysis to process messages as they arrive, utilizing stream processing technologies (e.g., Apache Kafka) for low-latency detection.
  + **User Feedback Loop:** Create a user-friendly interface for users to report false positives and false negatives, contributing to continuous model improvement.

## Step 5: Unsupervised and Reinforcement Learning

* + **Unsupervised Learning**: Implement unsupervised techniques like clustering or autoencoders to detect new and unseen spam patterns without relying solely on labeled data.
  + **Reinforcement Learning:** Explore reinforcement learning to train the classifier to make sequential decisions, adapting to changing spam tactics.

## Step 6: Privacy and Ethics

Page **10** of **28**

* + **Privacy-Preserving Techniques:** Ensure user data privacy using techniques like federated learning or differential privacy during model training and inference.



* + **Bias Mitigation:** Implement strategies to reduce bias in the classifier to ensure fair and ethical decision-making.

## Step 7: Continuous Learning and Monitoring

* + **Concept Drift Detection:** Develop algorithms to detect concept drift, where spam tactics change over time. Regularly retrain the model to adapt to evolving spam patterns.
  + **Model Monitoring:** Continuously monitor the classifier's performance in production, employing tools like Grafana and Prometheus to track key metrics.

## Step 8: Explainability and Transparency

* + **Explainability Techniques:** Implement methods to explain the AI's decision-making process, making it

Page **11** of **28**

transparent and interpretable to users and



regulators.

## Step 9: Cross-Platform Integration

* + **Integration Across Channels:** Extend the spam classifier's functionality to different platforms and communication channels, such as email, social media, and messaging apps.

## Step 10: Evaluation and Optimization

* + **A/B Testing:** Conduct A/B testing to evaluate different model configurations and algorithms, allowing data-driven optimization.

## Step 11: User Education and Feedback

* + **User Training:** Educate users on how to use the spam classifier effectively and how to provide feedback for model improvement.
  + **Continuous Feedback Analysis:** Regularly analyze user feedback to identify areas for enhancement and fine-tuning.

Page **12** of **28**



## Step 12: Scaling and Maintenance

* + **Scaling Infrastructure:** If needed, scale the infrastructure to handle increased load and data volume as the user base grows.
  + **Regular Maintenance:** Maintain and update the system regularly to ensure its effectiveness and security.

# DEVELOPMENT:- PHASE:-1

Building a smarter AI-powered spam classifier involves several steps, starting with loading and preprocessing the dataset. In this, we'll use Python and popular libraries like pandas and scikit-learn to load and preprocess a spam email dataset. It utilizes the NLTK library for text preprocessing and the SCIKIT-learn library for meachine learning.

**The script uses a Multinomial Naive Bayes classifier to classify messages as spam or non-spam. The accuracy of the model is evaluated.**

**THE STEPS ARE AS FOLLOW’S:**

* + - **1: IMPORT LIBRARIES**
    - **2: LOAD AND EXPLORE THE DATASET**
    - **3: DATA PREPROCESSING**

Page **13** of **28**

* + - **4: SPLIT DATA INTO TRAINING AND TESTING SETS**



* + - **5: BUILD AND TRAIN THE SPAM CLASSIFIER**
    - **6: EVALUATE THE MODEL**

**STEP 1:- (IMPORT LIBRARIES)**

import pandas as pd import nltk

from nltk.corpus import stopwords

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,

confusion\_matrix, classification\_report

Before running the code, make sure you have the following libraries installed:

* + - **Pandas**
    - **Nltk**
    - **scikit-learn**

You can install these libraries using **`pip`: ( pip install pandas nltk scikit-learn)**

Page **14** of **28**

Additionally, you'll need to download the NLTK stopwords



dataset. You can do this by running the following code:

**(nltk.download('stopwords')**

**STEP 2:- (LOAD AND EXPLORE THE DATASET)**

**Load the dataset from the Kaggle link**

You need to download the dataset and provide the appropriate file path

**Data=pd.read\_csv("C:\\Users\\prath\\Downloads\\spam.cs v", encoding="latin-1")**

**Explore the dataset**

print(data.head())

print(data['label'].value\_counts ())

**STEP 3:-(DATA PREPROCESSING)**

The following data preprocessing steps are performed:

* Irrelevant columns are removed: Unnamed: 2, Unnamed: 3, and Unnamed: 4.
* Column names are renamed for better understanding: Label, Message.
* Labels are converted to binary format (spam: 1, non-spam: 0).

Page **15** of **28**

* Text data is preprocessed by removing stopwords and applying



TF-IDF vectorization.

# 1. Remove irrelevant columns (if any)

data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)

# 2. Rename columns for better understanding data.columns = ["Label", "Message"]

# 3. Convert labels to binary (spam: 1, non-spam: 0)

data["Label"] = data["Label"].map({"spam": 1, "ham": 0})# 4. Text preprocessing (removing stopwords and applyingTF-IDF)

stop\_words = 'english' # Use the built-in English stop words

tfidf\_vectorizer=TfidfVectorizer(stop\_words=stop\_words,max\_feat ures=5000)

x=tfidf\_vectorizer.fit\_transform(data["Message"]) #Assuming "v2" contains your text data

**STEP 4:-( SPLIT DATA INTO TRAINING AND TESTING SET’S)**

Split the dataset into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, data["Label"], test\_size=0.2, random\_state=42)

Page **16** of **28**

**STEP 5:-( BUILD AND TRAIN THE SPAM CLASSIFIER)**



# Build and Train the Spam Classifier

classifier = MultinomialNB() classifier.fit(x\_train, y\_train) # Make predictions

y\_pred = classifier.predict(x\_test)

**STEP 6:-( EVALUATE THE MODEL)**

Evaluating the spam classifier on the testing data using metrics like accuracy, precision, recall, and F1-score.

# Evaluate the classifier

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

confusion = confusion\_matrix(y\_test, y\_pred)

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

print("Accuracy:

{:.2f}%".format(accuracy \* 100))

print("Confusion Matrix:\n", confusion)

print("Classification Report:\n", report)

**THE BELOW PYTHON PROGRAM SHOWS THE RESULTS THAT ARE**

Page **17** of **28**

**OBTAINED FROM TRAING AND TESTING THE DATASET’S**



# Import necessary libraries import pandas as pd import nltk

from nltk.corpus import stopwords

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, confusion\_matrix, classification\_report

# Download the NLTK stopwords if not already downloaded nltk.download('stopwords')

# Load the dataset from the Kaggle link

# You need to download the dataset and provide the appropriate file path

data = pd.read\_csv("C:\\Users\\prath\\Downloads\\spam.csv", encoding="latin-1")

# Explore the dataset print(data.head())

# Data Preprocessing

# 1. Remove irrelevant columns (if any)

data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)

Page **18** of **28**

# 2. Rename columns for better understanding



data.columns = ["Label", "Message"]

# 3. Convert labels to binary (spam: 1, non-spam: 0) data["Label"] = data["Label"].map({"spam": 1, "ham": 0})

# 4. Text preprocessing (removing stopwords and applying TF-IDF)

stop\_words = 'english' # Use the built-in English stop words tfidf\_vectorizer = TfidfVectorizer(stop\_words=stop\_words,

max\_features=5000)

x = tfidf\_vectorizer.fit\_transform(data["Message"]) # Assuming "v2" contains your text data

# 5. Split the dataset into training and testing sets

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, data["Label"], test\_size=0.2, random\_state=42)

# Build and Train the Spam Classifier classifier = MultinomialNB() classifier.fit(x\_train, y\_train)

# Make predictions

y\_pred = classifier.predict(x\_test) # Evaluate the classifier

accuracy = accuracy\_score(y\_test, y\_pred) confusion = confusion\_matrix(y\_test, y\_pred)

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

Page **19** of **28**

print("Accuracy: {:.2f}%".format(accuracy \* 100))



print("Confusion Matrix:\n", confusion) print("Classification Report:\n", report)

**OUTPUT:-**

**Accuracy: 97.58% Confusion Matrix:**

**[[965 0]**

**[ 27 123]]**

**Classification Report:**

**Precision recall f1-score support**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **0.97** | **1.00** | **0.99** | **965** |
| **1** | **1.00** | **0.82** | **0.90** | **150** |

**accuracy 0.98 1115**

**macro avg 0.99 0.91 0.94 1115**

**weighted avg 0.98 0.98 0.97 1115**

Page **20** of **28**

# DEVELOPMENT:- PHASE:-2



**Building a smarter AI-powered spam classifier involves several steps, starting with loading and preprocessing the dataset and feature engineering and model training and evaluating it . In this, we'll use Python language to develop this project. This project is a spam classifier that use to classify SMS messages as spam or not spam (ham). It is implemented in Python and uses the scikit- learn library for machine learning tasks.**

**(In this development part we will made feature extraction and model training and evaluating the model)**

**Feature Engineering:**

**Feature selection: Identify relevant features from the dataset that can help differentiate between spam and non-spam messages. Common features include word frequency, sender information, and message length.**

**Text preprocessing: Clean and preprocess the text data, including tasks like removing punctuation, stop words, and stemming or lemmatization.**

**Model Training:**

**Select an appropriate machine learning or deep learning model for your task. Common choices include Naive Bayes, Support Vector Machines, or Recurrent Neural Networks (RNNs).**

**Split the dataset into training and testing sets for model evaluation. Train the model on the training data.**

**Evaluation:**

Page **21** of **28**

**Use evaluation metrics such as accuracy, precision,**



**recall, F1-score, and ROC AUC to assess the model's performance.**

**Consider a confusion matrix to understand true positives, true negatives, false positives, and false negatives.**

**Dataset**

**The dataset used for this project is the "SMS Spam Collection" dataset from Kaggle, which can be found**

**[here]**

**(**[**https://www.kaggle.com/datasets/uciml/sms-spam-collection-**](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)[**dataset**](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)**)**

**PROJECT STRUCTURE:-**

**The project is structured as follows:**

**(spam\_classifier.py) : The Python script that contains the code for data preprocessing, feature extraction, model training, and evaluation.**

**(AI-PHASE4.dox) : This document consists of the documentation of this project.**

**The script will load the dataset, preprocess the data, train a Naive Bayes classifier, and evaluate its performance. The results will be displayed in the terminal.**

**Configuration**

**You can adjust the `max\_features` parameter in the**

**`TfidfVectorizer` to control the number of features used for text representation.**

Page **22** of **28**

**You can replace the machine learning algorithm**

**(currently using Naive Bayes) with other algorithms from scikit- learn, such as Support Vector Machine or Random Forest, for experimentation.**

**THE BELOW CODE IS THE PYHTON SCRIPT THAT WAS IMPLEMENTED IN THIS PROJECT;**

**IMPORTING LIBRARIES**

**import pandas as pd**

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

**from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.naive\_bayes import MultinomialNB**

**from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score**

**LOAD THE DATASET**

**data=pd.read\_csv("C:\\Users\\prath\\Downloads\\spam.csv", encoding='latin-1')**

**data = data[['v1', 'v2']] data.columns = ['label', 'text']**

**CONVERT LABELS TO BINARY (SPAM: 1, HAM: 0)**

**data['label'] = data['label'].apply(lambda x: 1 if x=='spam' else 0)**

**FEATURE EXTRACTION**



Page **23** of **28**

**tfidf\_vectorizer = TfidfVectorizer(max\_features=5000)**



**X = tfidf\_vectorizer.fit\_transform(data['text']) y = data['label']**

**SPLIT THE DATA**

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

**CHOOSE A MACHINE LEARNING ALGORITHM**

**# Using Naive Bayes classifier = MultinomialNB()**

**TRAIN THE MODEL**

**classifier.fit(X\_train, y\_train) EVALUATE MODEL PERFORMANCE**

**y\_pred = classifier.predict(X\_test) ACCURACY**

**accuracy = accuracy\_score(y\_test, y\_pred) print(f'Accuracy: {accuracy}') CLASSIFICATION REPORT**

**print(classification\_report(y\_test, y\_pred)) CONFUSION MATRIX**

**print(confusion\_matrix(y\_test, y\_pred))**

Page **24** of **28**

**THE BELOW IS THE COPY OF THE PYTHON**



**SCRIPT WITH THE OUTPUT**

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

# Load the dataset

data = pd.read\_csv("C:\\Users\\prath\\Downloads\\spam.csv", encoding='latin-1') data = data[['v1', 'v2']]

data.columns = ['label', 'text']

# Convert labels to binary (spam: 1, ham: 0)

data['label'] = data['label'].apply(lambda x: 1 if x == 'spam' else 0)

# 2. Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000) X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# 3. Split the Data

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

# 4. Choose a Machine Learning Algorithm # Using Naive Bayes

classifier = MultinomialNB()

# 5. Train the Model classifier.fit(X\_train, y\_train)

# 6. Evaluate Model Performance y\_pred = classifier.predict(X\_test) # Accuracy

accuracy = accuracy\_score(y\_test, y\_pred) print(f'Accuracy: {accuracy}')

OUTPUT;

Accuracy: 0.9704035874439462

# Classification Report

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0.97 | 1.00 | 0.98 | 965 |
| 1 | 1.00 | 0.78 | 0.88 | 150 |

Page **25** of **28**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| accuracy |  |  | 0.97 | 1115 |
| macro avg | 0.98 | 0.89 | 0.93 | 1115 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1115 |
| # Confusion Matrix print(confusion\_matrix(y\_test, y\_pred)) OUTPUT;  [[965 0]  [ 33 117]] | | | | |



**THE CODE WILL DISPLAY THE FOLLOWING**

**RESULTS:**

* Accuracy: The accuracy of the spam classifier on the test data.
* Classification Report: A summary of precision, recall, F1-score, and support for each class (spam and ham).
* Confusion Matrix: A matrix showing true positive, true negative, false positive, and false negative counts.

# To Run the Code

**Follow these steps to run the code**

1. **Clone this repository or download the code files.**

Page **26** of **28**

1. **Ensure you have Python and the necessary libraries (scikit-**



**learn, pandas) installed.**

1. **Download the dataset from the provided Kaggle link and save it as `spam.csv` in the project directory.**
2. **Open a terminal or command prompt.**
3. **Navigate to the project directory.**
4. **Run the following command to execute the spam classifier: ( spam\_classifier.py )**

**CONCLUSION:-**

This project aims to create a solution that is robust, adaptable, and user-friendly, ultimately improving the online communication experience for user’s. "Building a Smarter AI-Powered Spam Classifier," is poised to bring innovation and intelligence to spam detection. By combining advanced technologies, user feedback, and ethical considerations,**we aim to create a highly effective and user-friendly spam classifier that sets new standards**

Page **27** of **28**

**in the field.** And in the development part we have



collected and preprocessed data, engineered features, selected and trained a model,evaluated its performance from these steps we have lay the foundatation for a successful spam classification system.This conclusion represents the culmination of our efforts in Phase 2(development part), and it sets the stage for the next steps in our project. **We anticipate that this classifier will have a positive impact in addressing the issue of spam messages, making communication channels safer and more efficient. . We look forward to continued progress and innovation in the development of our AI-powered spam classifier."**

Page **28** of **28**