**Building a Smarter AI-Powered Spam Classifier**

**Overview**

This project is aimed at building a spam classifier to differentiate between spam and non-spam (ham) messages. It involves preprocessing text data, feature extraction using TF-IDF vectorization, training a machine learning model, and evaluating its performance.

**Design Thinking Process**

The project followed a structured design thinking process:

* Understanding the Problem: We began by understanding the problem of spam classification and its importance in communication.
* Ideation: We brainstormed various approaches for building a spam classifier, considering data preprocessing, feature extraction, and machine learning models.
* Prototyping: We implemented and tested different components, iterating until achieving satisfactory results.
* Testing and Evaluation: Rigorous testing and evaluation were performed to ensure the model's effectiveness.

**Dependencies**

Ensure you have the following dependencies installed to run the code:

- Python 3.x

- Pandas

- Scikit-Learn

- NumPy

You can install the required libraries using pip:

pip install pandas scikit-learn numpy

**Dataset**

* Dataset Source: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset
* Description: The dataset contains SMS messages labeled as 'ham' (non-spam) and 'spam'. It is provided in a CSV format.

**Instructions**

* Download the dataset (spam.csv) from the provided Kaggle source or is in the repository.
* Clone the project repository to the local machine or download the code files.

**Phases of Development**

The project can be divided into the following phases:

* Data Preprocessing: Handling data cleaning, missing values, duplicates, and text-specific preprocessing.
* Feature Extraction: Using TF-IDF vectorization to convert text data into numerical features.
* Model Selection: Choosing the Naive Bayes classifier as the machine learning algorithm.
* Model Training: Training the classifier on the preprocessed data.
* Model Evaluation: Assessing the model's performance using various evaluation metrics.

**Data Preprocessing**

* Load the dataset into your preferred data analysis environment.
* Run the AI\_phase3 script to clean and preprocess the data. This script will handle missing values, duplicates, and apply text-specific preprocessing.

**Choice of Machine Learning Algorithm**

The Naive Bayes classifier was chosen as the machine learning algorithm due to its effectiveness in text classification tasks and relatively low complexity.

**Model Training**

* After preprocessing, you can train the spam classifier.
* Run the AI\_phase4 to train the model using a Naive Bayes classifier and TF-IDF features.

**Model Evaluation**

* Finally, evaluate the model's performance using appropriate metrics.
* Run the AI\_phase4 script to generate a confusion matrix, classification report, and accuracy score.

**Results**

You will find the model's performance metrics, including accuracy, precision, recall, and F1-score, in the terminal output after running the AI\_phase4.py script.