**BUILDING A SMARTER AI-POWERED SPAM CLASSFIER**

**PROBLEM DEFINITION:**

The problem is to build an AI-powered spam classifier that can accurately distinguish between spam and non-spam messages in emails or text messages. The goal is to reduce the number of false positives (classifying legitimate messages as spam) and false negatives (missing actual spam messages) while achieving a high level of accuracy.

**DESIGN THINKING :**

**Data Collection:**

Kaggle is a great resource for datasets, and you can find several spam email datasets there.

Ensure that the dataset is balanced, i.e., it has a roughly equal number of spam and non-spam (ham) messages. Imbalanced datasets can lead to biased models.

**Data Preprocessing:**

Cleaning the data is crucial. Along with removing special characters and converting text to lowercase, consider these steps:

Removing stop words (common words like "and," "the," "in" that don't carry much information).

Stemming or lemmatization to reduce words to their root form.

Handling misspellings and abbreviations.

**Feature Extraction:**

TF-IDF is a good choice for converting text into numerical features. You can also experiment with other techniques like word embeddings (Word2Vec, GloVe) or even more advanced techniques like BERT embeddings if you have a large dataset.

**Model Selection:**

Start with simple models like Naive Bayes, as they often work surprisingly well for text classification tasks.

Explore other algorithms like Support Vector Machines, Random Forest, or Gradient Boosting**.**

For deep learning, you can experiment with neural network architectures like LSTM or CNN, or even pre-trained models like BERT or GPT.

**Evaluation:**

Use standard evaluation metrics for binary classification, such as accuracy, precision, recall, and F1-score.

Consider using techniques like cross-validation to get a more robust estimate of your model's performance.

Don't forget to check for overfitting, especially when using complex models.

**Iterative Improvement:**

Experiment with hyperparameter tuning to find the best settings for your chosen model.

Consider ensemble methods to combine the predictions of multiple models, which can often improve performance.

Keep an eye on false positives and false negatives, as the cost of misclassifying spam can vary depending on the application.

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