# AI-Powered Spam Classifier

**Our team:**

R . Bagavathi Raja

R. Nithish

M.Yogesh

M.Kavinesh

T.Nithyanantham

## INDEX

* Problem Statement
* Designing Thinking process
* Implementation using libraries and integration of NLP techniques.
* Programs

## Problem Statement

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.

## Designing Thinking Process

1. The provided code loads and preprocesses a dataset of SMS messages, preparing it for training a spam classifier. It applies text preprocessing techniques like converting text to lowercase and removing special characters, punctuation, and numbers.

2. NLP Integration: The NLP integration in the provided code involves text preprocessing to clean and tokenize the text data, as well as label encoding to convert text labels into numerical format. It also demonstrates the use of TF-IDF vectorization to transform text data into numerical features for machine learning.

3.Testing and Improvement: Testing and improvement involve evaluating the model's performance using metrics like accuracy, precision and F1-score. Based on the evaluation results, the model is fine-tuned, and data preprocessing techniques are refined to enhance its accuracy and effectiveness. This iterative process continues until the desired level of performance is achieved.

## Implementation using libraries and integration of NLP techniques

* + Python was used as the primary programming language.
  + The code relies on libraries like Pandas, Scikit-Learn, and NumPy for data manipulation, preprocessing, and TF-IDF vectorization.
  + It integrates NLP techniques such as text preprocessing and label encoding.
  + The use of the TF-IDF vectorizer is an essential NLP technique for converting text data into numerical features, which is a common practice in text classification tasks.

# **Program**:

## Dataset Link:

## <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.preprocessing import Label Encoder

from sklearn.utils import shuffle

# Load the dataset

data = pd.read\_csv('spam.csv', encoding='latin-1')

# Drop unnecessary columns

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

# Rename columns for clarity

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

# Shuffle the dataset

data = shuffle(data)

# Preprocess the text data

def preprocess\_text(text):

# Convert text to lowercase

text = text.lower()

# Tokenize the text (split it into individual words)

tokens = text.split()

# Remove special characters, punctuation, and numbers

tokens = [word for word in tokens if word.isalpha()]

# Remove common stop words (optional)

# You can use NLTK or SpaCy for more extensive stop word removal

# tokens = [word for word in tokens if word not in stopwords]

# Rejoin the tokens into a single string

text = ' '.join(tokens)

return text

data['message'] = data['message'].apply(preprocess\_text)

# Encode the labels (spam as 1, ham as 0)

le = LabelEncoder()

data['label'] = le.fit\_transform(data['label'])

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['message'], data['label'], test\_size=0.2, random\_state=42)

# TF-IDF Vectorization

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

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

from sklearn.metrics import accuracy\_score

# Calculate accuracy

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

# Display the accuracy

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

# THANK YOU