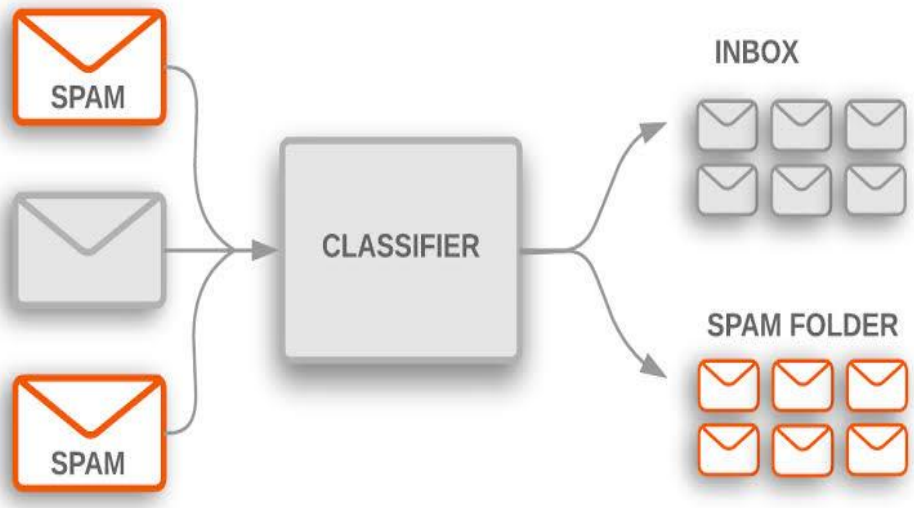
# BUILDING A SMATER AI-POWERED SPAM CLASSIFIER

## TEAM MEMBER

210221104005 : BALAMURUGAN

## PHASE 3 :- DEVELOPMENT PART-1




**PROJECT:-** SPAM CLASSIFIER

## DEVELOPMENT:-

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**
- **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)**

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.csv", 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).

- 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_features=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)
```

**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 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)

# 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)

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 |

**CONCLUSION:-**

In this 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.