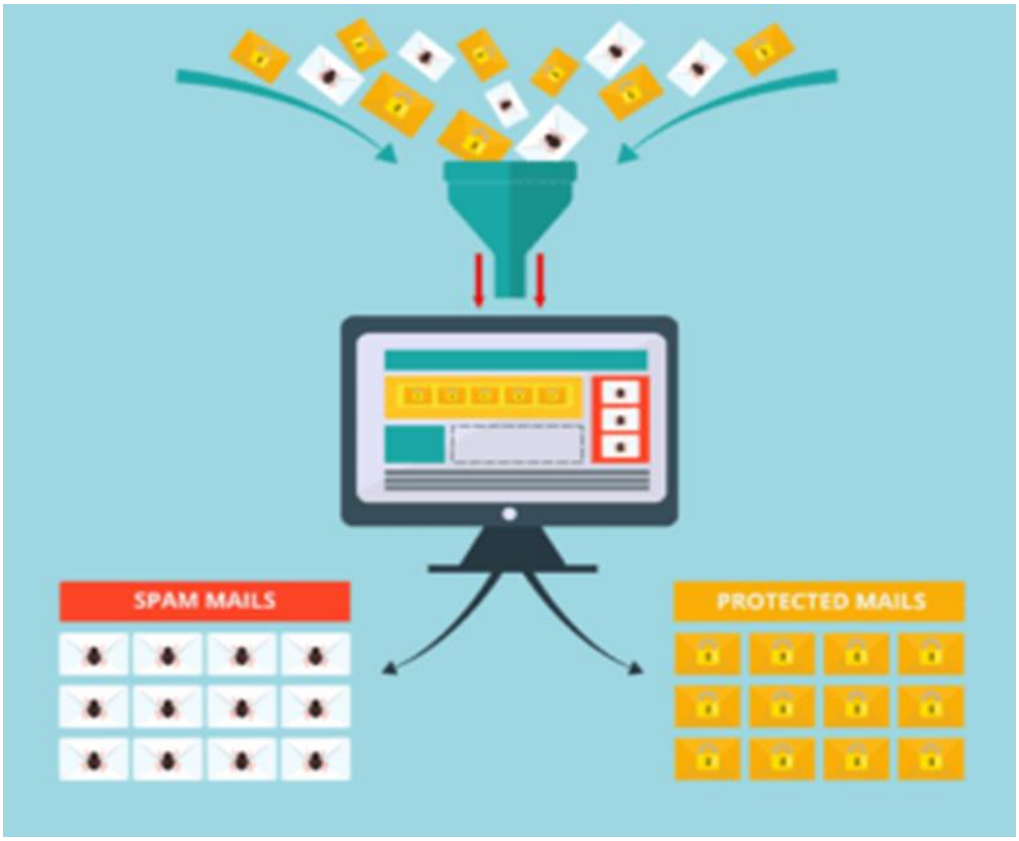
BUILDING A SMATER AI-POWERED SPAM CLASSIFIER

TEAM MEMBER

210221104002 : ANAND JAGANATHAN

PHASE 4 :- DEVELOPMENT PART-2



PROJECT:- SPAM CLASSIFIER

DEVELOPMENT:-

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:

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-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.

To Run the Code

Follow these steps to run the code

1. Clone this repository or download the code files.

2. Ensure you have Python and the necessary libraries (scikit-learn, pandas) installed.

3. Download the dataset from the provided Kaggle link and save it as `spam.csv` in the project directory.

4. Open a terminal or command prompt.

5. Navigate to the project directory.

6. Run the following command to execute the spam classifier:

(python spam_classifier.py)

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.

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

```
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)
```

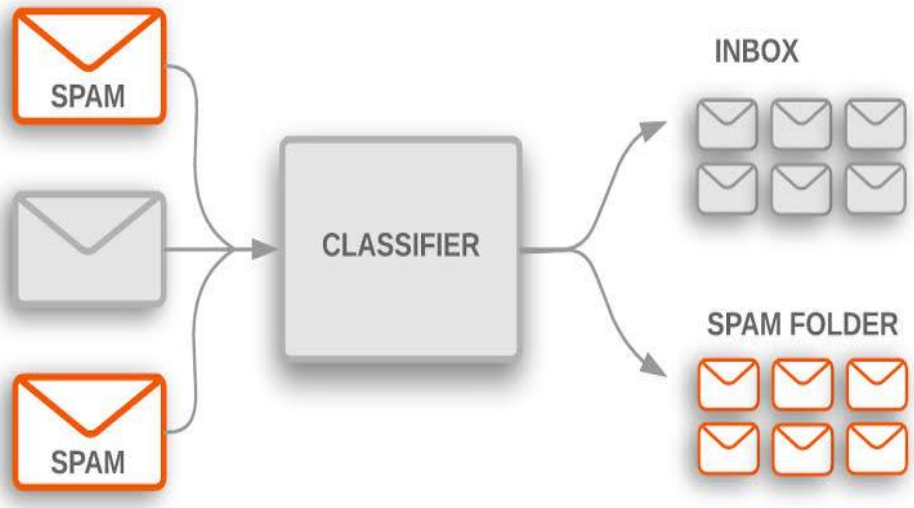
print(f'Accuracy: {accuracy}')

CLASSIFICATION REPORT

print(classification_report(y_test, y_pred))

CONFUSION MATRIX

print(confusion_matrix(y_test, y_pred))



THE BELOW IS THE COPY OF THE PYTHON SCRIPT WITH THE OUTPUT

```python
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

    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.

CONCLUSION:-

This conclusion represents the culmination of our efforts in Phase 4, 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."