# BUILDING A SMARTER AI- POWERED SPAM CLASSIFIER

TEAM MEMBER

622021243020: HARINI B

PHASE 2 SUBMISSION DOCUMENT

PROJECT: BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER



INTRODUCTION:.

* **In today's digital age, where communication channels are flooded with a relentless stream of messages and emails, the need for an intelligent and effective spam classifier has never been more critical. Traditional rule-based filters can no longer keep pace with the ever-evolving tactics of spammers. To address this integration of Artificial Intelligence (AI) has emerged as a powerful solution.**
* **Building a smarter AI-powered spam classifier is not merely about separating spam from legitimate messages; it's about creating a dynamic system capable of adapting to the ingenious techniques employed by spammers. This endeavour combines the realms of data science, machine learning, and natural language processing to develop a robust defence mechanism against unwanted and potentially harmful content.**
* **In this journey, we will explore the key steps and strategies required to construct an AI-powered spam classifier that not only identifies spam with high accuracy but also continually evolves to stay ahead of the spammer's tactics. From data collection and preprocessing to model selection, training, and deployment, we will delve into the intricacies of building an intelligent guardian for your communication channels.**

CONTENT FOR PROJECT PHASE 2:

**Consider Machine Learning techniques like Random Forest classifier for detecting E-mail spam.**

# Email Spam Classifier



**The objective is to develop a machine learning model that can categorize emails into two categories: spam and non-spam (often referred to as "ham").**

DATA SOURCE:

**A good data source for email spam classifier using machine learning should be accurate,complete,covering the geographic area of interest,accessible.**

**Dataset Link :**

(https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)

* Data Collection:

**We will need a containing labeled examples of spam and non spam messages. We can use a kaggal dataset for this purpose .By carefully collecting and preparing a well-labeled and diverse dataset, you set a strong foundation for training an effective spam classifier.**

* Data Preprocessing:

**The text data needs to be cleaned and preprocessed. This involves removing special characters, converting text to lowercase, Text cleaning and tokenizing the text into individual words**

* Feature Extraction:

**We will convert the tokenized words into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).**

* Model Selection:

**We can experiment with various machine learning algorithms such as Naive Bayes, Support Vector Machines, and more advanced techniques like deep learning using neural networks.**

* Evaluation:

**We will measure the model' s performance using metrics like accuracy, precision, recall, and F1-score.**

* Iterative Improvement:

**We will fine-tune the model and experiment with hyper parameters to improve its accuracy.**

## PROGRAM:

## *EMAIL SPAM CLASSIFIER*

## Importing Necessary Libraries

In [1]:

*# Import Libraries*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

## Load and Explore the Dataset

In [2]:

*# Load the dataset*

df = pd.read\_csv("/kaggle/input/sms-spam-collection-dataset/spam.csv", encoding='ISO-8859-1')

In [3]:

*# Display the first few rows of the dataset*

df.head()

Out[3]:

|  | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
| --- | --- | --- | --- | --- | --- |
| 0 | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| 1 | Ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| 2 | Spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| 3 | Ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |
| 4 | Ham | Nah I don't think he goes to usf, he lives aro... | NaN | NaN | NaN |

## Data Preprocessing

In [4]:

*# Display the column names of the DataFrame*

print(df.columns)

Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')

In [5]:

*# Convert 'spam' and 'ham' to binary labels*

df['v1'] = df['v1'].map({'spam': 0, 'ham': 1})

In [6]:

*# Split the data into features (X) and target (Y)*

X = df["v2"]

Y = df["v1"]

In [7]:

*# Split the data into training and test sets*

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.35, random\_state=3)

## Feature Extraction - TF-IDF

In [8]:

*# TF-IDF feature extraction*

tfidf\_vectorizer = TfidfVectorizer(min\_df=1, stop\_words='english', lowercase=True)

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

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

## Model Training (Random Forest)

In [9]:

*# Model training*

model = RandomForestClassifier(n\_estimators=100, random\_state=3)

model.fit(X\_train\_features, Y\_train)

Out[9]:

RandomForestClassifier

RandomForestClassifier(random\_state=3)

## Model Evaluation (Random Forest)

In [10]:

prediction\_on\_training\_data = model.predict(X\_train\_features)

accuracy\_on\_training\_data = accuracy\_score(Y\_train, prediction\_on\_training\_data)

prediction\_on\_test\_data = model.predict(X\_test\_features)

accuracy\_on\_test\_data = accuracy\_score(Y\_test, prediction\_on\_test\_data)

In [11]:

*#Print accuracy*

print('Accuracy on training data: **{:.2f}** %'.format(accuracy\_on\_training\_data \* 100))

print('Accuracy on test data: **{:.2f}** %'.format(accuracy\_on\_test\_data \* 100))

Output:

Accuracy on training data: 100.00 %

Accuracy on test data: 97.49 %

## Confusion Matrix Visualization(Random Forest Classifier)

In [12]:

*# Confusion Matrix Visualization*

conf\_matrix = confusion\_matrix(Y\_test, prediction\_on\_test\_data)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues", cbar=False,

xticklabels=['Spam', 'Ham'], yticklabels=['Spam', 'Ham'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

out[12]:

confusion matrix

|  |  |
| --- | --- |
| 236 | 44 |
| 5 | 1666 |

spam ham

predicted

## 

## Classification Report (Random Forest Classifier)

In [13]:

classification\_rep = classification\_report(Y\_test, prediction\_on\_test\_data, target\_names=['Spam', 'Ham'])

print("Classification Report:")

print(classification\_rep)

Out[13]:

Classification Report:

precision recall f1-score support

Spam 0.98 0.84 0.91 280

Ham 0.97 1.00 0.99 1671

accuracy 0.97 1951

macro avg 0.98 0.92 0.95 1951

weighted avg 0.97 0.97 0.97 1951

## Feature Importance Visualization (Random Forest)

In [14]:

feature\_importance = model.feature\_importances\_

feature\_names = tfidf\_vectorizer.get\_feature\_names\_out()

sorted\_idx = np.argsort(feature\_importance)[-20:] *# Top 20 important features*

plt.figure(figsize=(4,7))

plt.barh(range(len(sorted\_idx)), feature\_importance[sorted\_idx], align="center")

plt.yticks(range(len(sorted\_idx)), [feature\_names[i] for i **in** sorted\_idx])

plt.xlabel("Feature Importance")

plt.ylabel("Feature")

plt.title("Top 4 Important Features (Random Forest)")

plt.show()

Out[14]:

Random Forest

## Make Predictions on New Input (Random Forest Classifier)

In [15]:

input\_your\_mail = "Keep yourself safe for me because I need you and I miss you already and I envy everyone that see's you in real life"

input\_data\_features = tfidf\_vectorizer.transform([input\_your\_mail])

prediction = model.predict(input\_data\_features)

if prediction[0] == 1:

print("Ham Mail")

else:

print("Spam Mail")

Out[15]:

Ham Mail

CONCLUSION AND FUTURE WORK(PHASE 2):

PROJECT CONCLUSION:

* **In conclusion, building an email spam classifier using the Random Forest model is a promising approach to effectively identify and filter spam messages from legitimate ones. Random Forest, an ensemble learning technique, combines the strength of multiple decision trees to achieve higher accuracy and robustness in classification tasks.**
* **Incorporating Random Forest as the model choice, combined with thoughtful feature engineering and a systematic iterative improvement approach, enables the development of an efficient and accurate email spam classifier. The classifier helps in enhancing email security, ensuring that users receive legitimate emails while minimizing the impact of spam in their inbox.**
* **In the realm of email spam classification, several exciting avenues for future work and improvements can be explored to enhance the accuracy, efficiency, and adaptability of spam detection systems.**
* **Continuously exploring these future directions will contribute to the evolution of email spam classification systems, leading to more efficient, accurate, and adaptive spam detection mechanisms that cater to evolving spamming techniques and user needs.**