**Phase 5**

Building a Smarter AI-Powered Spam Classifier



**Team Members: -**

Santhiya M

Sahana S

Remgis Ezhil Belsi I

Problem Statement:

The goal of this project is to build a AI powered spam classifier .This is a common problem in natural language processing and text classification. The project aims to create an effective spam filter that can be used to protect users from unwanted and potentially harmful text messages.

Phases of Development:

1. Data Collection and Exploration
2. Data Preprocessing
3. Feature Extraction
4. Model Selection
5. Model Training
6. Model Evaluation

Design Thinking Process:

Data Collection

* Objective: Gather a dataset containing labeled examples of spam and non-spam messages.
* Approach: Utilize a Kaggle dataset or similar reliable sources that provide a diverse and representative dataset of both spam and non-spam messages. The dataset forms the foundation for training and evaluating the spam classifier.

Data Preprocessing

* Objective: Prepare the text data for modeling by cleaning and standardizing it.
* Description: This step involves various data cleaning tasks, including the removal of special characters, conversion of text to lowercase to ensure consistency, and tokenization to break the text into individual words or phrases. Additionally, stemming or lemmatization can be applied to reduce words to their base forms for uniformity.

Feature Extraction

* Objective: Convert the tokenized text data into numerical features for machine learning models.
* Description: To enable machine learning algorithms to work with the text data, feature extraction is essential. The chosen technique, TF-IDF (Term Frequency-Inverse Document Frequency), assigns numerical values to words based on their importance in the text. Alternatively, word embeddings like Word2Vec or GloVe can capture semantic relationships between words for more advanced models.

Model Selection

* Objective: Choose an appropriate machine learning algorithm for spam classification.
* Description: The model selection phase involves experimenting with various machine learning algorithms, starting with baseline models like Naive Bayes, Support Vector Machines, and logistic regression. Advanced techniques, such as deep learning using neural networks (e.g., LSTM or CNN), can also be explored to capture complex patterns in text data. Ensemble methods like Random Forest or Gradient Boosting may be considered to combine multiple models for improved performance.

Evaluation

* Objective: Assess the performance of the spam classifier using relevant evaluation metrics.
* Metrics: The evaluation phase includes measuring the model's performance using metrics such as accuracy, precision, recall, and F1-score. The confusion matrix provides insights into true positives, true negatives, false positives, and false negatives, offering a comprehensive view of classifier performance.
* Iterative Improvement
* Objective: Continuously refine and enhance the spam classifier's performance.
* Description: The iterative improvement stage involves several strategies for model enhancement. This includes fine-tuning model hyper parameters through techniques like grid search or random search, exploring feature engineering to extract more informative features, addressing class imbalance issues if present, implementing cross-validation to ensure model robustness, and regularly updating the model with new data to adapt to evolving spam tactics.

Dataset Description:

The dataset used for this project is available on Kaggle at the following link: [SMS Spam Collection Dataset](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset). It contains a collection of SMS messages labeled as spam or ham. The dataset is in CSV format and includes two columns: "v1" for the label (spam or ham) and "v2" for the text of the SMS messages.

Data Preprocessing Steps:

1. Removing any duplicate SMS messages.
2. Handling missing or null values, if any.
3. Text cleaning, including lowercasing, removing punctuation, and tokenization.
4. Stop word removal to filter out common words.
5. Lemmatization or stemming to reduce words to their base form.

Feature Extraction Techniques:

Text data will be converted into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This process represents each SMS message as a vector of numerical values.

Choice of Machine Learning Algorithm:

A common choice for text classification problems like this is the support vector Classification algorithm. It is known to perform well with text data.

Model Training:

The selected algorithm will be trained on the preprocessed dataset using techniques like cross-validation to avoid overfitting.

Evaluation Metrics:

The performance of the model will be evaluated using standard classification metrics such as accuracy, precision, recall, F1-score, and the ROC-AUC score.

Innovative Techniques or Approaches: I

To improve the model's performance, techniques such as hyper parameter tuning, feature selection, and ensembling (e.g., using multiple classifiers) may be explored.

Coding:

import numpy as np

import pandas as pd

import string

from sklearn.feature\_extraction.text import TfidfVectorizer

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

from sklearn.feature extraction.text import fidfVectorizer

import nltk

import re

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import SnowballStemmer

import seaborn as sns

from plotly import graph\_objs as go

import matplotlib as plt

from sklearn.nalve bayes Import Multicol

from sklearn.metrics Import accuracy score, classification report

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

from PIL import Image

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

from sklearn.preprocessing import LabelEncoder

message\_data=pd.read\_csv("/kaggle/input/sms-spam-collectiondataset/spam.csv",encoding = "latin")message\_data.head()

df = pd.read\_csv('spam.csv", encoding="ISO-8859-1")

print(df.head())

D & df = pd.read\_csv(spam.csv.encoding-150-8859-1")

x\_train, x\_test, y train, y test train\_test\_split(dfv1"), df[TEXT], test size-8.2, random\_state-42)

print(df)

data = pd.DataFrame({

"text":[

“This is a sample text for feature extraction.”

“Feature extraction is an Important step in NEP.”

“Extracting features helps in text classification.”],

'label': [1, 0, 1]

})

tfidf\_vectorizer=TfidfVectorizer()

tfidf\_features=tfidf\_vectorizer.fit\_transform(data["text"])

tridf\_uf =pd.DataFrame(data-tfid\_features.toarray(), columns-tfidf\_vectorizer.get\_feature\_names\_out())

print(tridf df)

X = df['v1']

y=df['text']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size-0.2, random\_state=42)

vectorizer = CountVectorizer()

X\_train\_counts vectorizer.fit\_transform(X\_train)

X\_test\_counts vectorizer.transform(X\_test)

clf = MultinomialNB()

clf.fit(X\_train\_counts, y\_train)

y\_pred = clf.predict(X\_test\_counts)

accuracy (y\_pred == y\_test).mean()

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

print(df)

nltk.download('punkt')

nltk.download('stopwords')

def preprocess\_text(text):

text = text.lower()

text = re.sub(r'[^a-zA-Z\s]', '' text)

tokens = word\_tokenize (text)

tokens = [word for word in tokens if word not in stopwords.words ('english')]

cleaned\_text = '.join(tokens)

return cleaned text

df['text'] = df['text'].apply(preprocess\_text)

print (df)

X=data['vi']

y = data['text']

label\_encoder = LabelEncoder()

y = label\_encoder.fit\_transform(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size-0.2, random\_state=42)

vectorizer = CountVectorizer()

X\_train\_counts = vectorizer.fit\_transform(X\_train) X\_test\_counts = vectorizer.transform(X\_test) 26

classifier = MultinomialNB()

classifier.fit(x train counts, y train)

y\_pred = classifier.predict(X\_test\_counts)

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

precision precision\_score(y\_test, y\_pred, average 'weighted')

recall = recall\_score(y\_test, y\_pred, average="weighted")

f1\_score (y\_test, y\_pred, average='weighted')

metrics = ['Accuracy', 'Precision', 'Recall, 'F1-Score']

values [accuracy, precision, recall, fl]

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

plt.bar(metrics, values, color=['blue', 'green', 'red', 'purple'])

plt.title('Spam Classifier Performance Metrics')

plt.xlabel('Metric')

plt.ylabel('Value')

plt.ylim(0, 1.0) # Set y-axis limits to the range [0, 1]

plt.show()

Conclusion:

This project showcases AI's efficacy in combating email spam. Using the Naive Bayes classifier, along with data preprocessing and feature extraction, a strong spam detector is created. High accuracy, precision, recall, and F1-score metrics underscore the effectiveness. Spam classification is vital for email security. Future work may involve advanced models, larger datasets, and real-time filtering. This project lays the foundation for enhancing email communication security.