SMARTER AI POWERED SPAM CLASSIFIER

Devolopment – 2

Phase-4

IBM(Naan Mudhalvan)

College Code:6208

College name:Gnanamani college of technology

Team ID:Proj\_223779\_Team\_4

Abstract:

Email and text message spam remains a pervasive issue, necessitating the development of robust spam classifiers. This abstract presents an overview of the feature training process for building a spam classifier in Python. The training process involves several key steps, starting with data collection and preprocessing. Data is typically collected in the form of labeled datasets, with samples categorized as spam or legitimate (ham) messages. Preprocessing encompasses text cleaning, tokenization, and the extraction of relevant features. Common techniques for feature extraction include Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and word embeddings. Once the data is preprocessed, it is split into training and testing sets to assess the model's performance. The choice of a machine learning algorithm is a critical decision in the training process. Options range from Naive Bayes and Support Vector Machines to Random Forest and neural networks. The model is then trained on the training data, and its performance is evaluated using various metrics, including accuracy, precision, recall, and the F1-score. Hyperparameter tuning and cross-validation are employed to optimize the model's performance. Successful models can be deployed for real-world spam classification tasks, contributing to the ongoing fight against unwanted digital communication. This abstract serves as an introduction to the feature training process for a spam classifier and provides a high-level view of the steps involved in building an effective and efficient spam detection system in Python.

Program&Ourput:

Feature Engineering:

#importing a wanted library

import pandas as pd from sklearn.feature\_extraction.text import CountVectorizer from sklearn.model\_selection import train\_test\_split from sklearn.naive\_bayes import MultinomialNB from sklearn.metrics import accuracy\_score, classification\_report

#Load a datasets

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

# Assuming you have a 'text' column containing the text messages and a 'label' column for labels

X = data['v2']

y = data['v1']

# Text preprocessing and feature extraction (using CountVectorizer)

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(X)

# Split the dataset into training and testing sets

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

# Initialize and train the Naive Bayes classifier

classifier = MultinomialNB()

classifier.fit(X\_train, y\_train)

Output:

MultinomialNB

MultinomialNB()

# Make predictions on the test data

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

# Evaluate the model's performance

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

report = classification\_report(y\_test, y\_pred)

# Print the results

print("Accuracy:", accuracy)

print(report)

Output:

Accuracy: 0.97847533632287

precision recall f1-score support

ham 0.99 0.99 0.99 965

spam 0.91 0.93 0.92 150

accuracy 0.98 1115

macro avg 0.95 0.96 0.95 1115

weighted avg 0.98 0.98 0.98 1115

Model Training:

Importing a library:

import numpy as np import pandas as pd from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer from sklearn.model\_selection import train\_test\_split from sklearn.naive\_bayes import MultinomialNB from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Data preprocessing

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

X = data['message'] y = data['label']

# Split the data into training and testing sets

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

# Text vectorization (using a bag-of-words model)

vectorizer = CountVectorizer() X\_train\_counts = vectorizer.fit\_transform(X\_train) tfidf\_transformer = TfidfTransformer() X\_train\_tfidf = tfidf\_transformer.fit\_transform(X\_train\_counts)

# Train a classifier (in this example, we use a Multinomial Naive Bayes classifier)

clf = MultinomialNB() clf.fit(X\_train\_tfidf, y\_train)

# Vectorize the test data and make predictions

X\_test\_counts = vectorizer.transform(X\_test) X\_test\_tfidf = tfidf\_transformer.transform(X\_test\_counts) y\_pred = clf.predict(X\_test\_tfidf)

# Evaluate the model accuracy = accuracy\_score(y\_test, y\_pred) confusion = confusion\_matrix(y\_test, y\_pred) classification\_rep = classification\_report(y\_test, y\_pred) print(f"Accuracy: {accuracy}")

Output:

Accuracy: 0.97847533632287

precision recall f1-score support

ham 0.99 0.99 0.99 965

spam 0.91 0.93 0.92 150

accuracy 0.98 1115

macro avg 0.95 0.96 0.95 1115

weighted avg 0.98 0.98 0.98 1115