TITLE: BUILDING SMARTER AI POWERED SPAM CLASSIFIER

ABSTRACT:

Email spam continues to be a pervasive problem, necessitating the development of smarter AI-powered spam classifiers. This one-page abstract provides an overview of key strategies and considerations for building a more intelligent spam classifier.

* Data Collection and Preprocessing: Gather diverse email data and preprocess it by removing HTML tags, tokenizing text, eliminating stop words, and extracting relevant features.
* Feature Engineering: Explore various features, including word embeddings , to capture nuanced spam patterns.
* Model Selection: Choose a suitable machine learning or deep learning model (e.g., neural networks) for text classification.
* Model Training and Validation: Split data into training, validation, and test sets. Fine-tune models, adjust hyperparameters, and employ techniques like cross-validation.
* Imbalanced Data Handling: Address class imbalance by oversampling or using SMOTE.
* Ensemble Methods: Combine multiple models using ensemble techniques to enhance performance.
* Hyperparameter Tuning: Optimize model hyperparameters through grid search or random search.
* Regularization and Dropout: Apply regularization to prevent overfitting and dropout in deep learning models.
* Feature Importance Analysis: Analyze feature or word importance using feature scores or attention mechanisms.
* Evaluation Metrics: Utilize precision, recall, F1-score, and ROC-AUC to assess classifier performance, focusing on false positives and false negatives.
* Continuous Monitoring and Updates: Retrain the model regularly with new data and monitor its performance in a production environment

Project Overview:

The goal of this project is to develop an advanced spam classifier using artificial intelligence (AI) techniques to enhance email and message filtering systems. The AI-powered spam classifier will intelligently distinguish between legitimate messages and spam, ensuring that users receive only relevant and non-malicious content in their inboxes.

Objectives:

•Design and implement a robust AI-powered spam classification system.

•Achieve a high level of accuracy in spam detection.

•Continuously adapt and improve the classifier's performance through machine learning techniques.

•Enhance user experience by reducing false positives and false negatives.

•Develop a user-friendly interface for configuring and managing spam filtering rules.

Methodology:

1. Data Collection: Gather a variety of spam and non-spam messages.

2. Data Cleaning: Clean and organize the data, removing any unnecessary information.

3. Feature Extraction: Identify important characteristics from the messages, like keywords or sender information.

4. Model Building: Create a spam classifier using machine learning techniques, like decision trees or neural networks.

5. Training: Train the model using labeled data (spam vs. non-spam).

6. Testing: Evaluate the model's accuracy on a separate set of data not used during training.

7. Real-time Integration: Implement the model into email or messaging systems to classify incoming messages.

8. User Feedback: Allow users to report spam or false positives to improve the model.

9. User Interface: Develop a user-friendly interface for users to manage their spam settings.

10. Continuous Improvement: Regularly update the model based on user feedback and emerging spam patterns.

program:

-\*- coding: utf-8 -\*-

# coding: utf-8

#Naive Bayes

import os

import io

import numpy

from pandas import DataFrame

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

#Function to read files (emails) from the local directory

def readFiles(path):

for root, dirnames, filenames in os.walk(path):

for filename in filenames:

path = os.path.join(root, filename)

inBody = False

lines = []

f = io.open(path, 'r', encoding='latin1')

for line in f:

if inBody:

lines.append(line)

elif line == '\n':

inBody = True

f.close()

message = '\n'.join(lines)

yield path, message

def dataFrameFromDirectory(path, classification):

rows = []

index = []

for filename, message in readFiles(path):

rows.append({'message': message, 'class': classification})

index.append(filename)

return DataFrame(rows, index=index)

#An empty dataframe with 'message' and 'class' headers

data = DataFrame({'message': [], 'class': []})

#Including the email details with the spam/ham classification in the dataframe

data = data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBreak/Data Science with Python & R/DataScience/DataScience-Python3/emails/spam', 'spam'))

data = data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBreak/Data Science with Python & R/DataScience/DataScience-Python3/emails/ham', 'ham'))

data = data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBreak/emails/spam', 'spam'))

data = data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBreak/emails/ham', 'ham'))

#Head and the Tail of 'data'

data.head()

print(data.tail())

vectoriser = CountVectorizer()

count = vectoriser.fit\_transform(data['message'].values)

print(count)

target = data['class'].values

print(target)

classifier = MultinomialNB()

classifier.fit(count, target)

print(classifier)

exampleInput = ["Hey. This is John Cena. You can't see me", "Free Viagra boys!!", "Please reply to get this offer"]

excount = vectoriser.transform(exampleInput)

print(excount)

prediction = classifier.predict(excount)

print(prediction)

output:

(0, 20104) 1 [0->1st sentence; 20104->word id; 1-> no. of times that the word occurs in the sentence]

(0, 15629) 1

(0, 30882) 1

(0, 50553) 1

(0, 36099) 1

(0, 44217) 1

(0, 58467) 1

(0, 51216) 1

(0, 10966) 1

(0, 47038) 1

(0, 46816) 1

(0, 54656) 1

(0, 43219) 2

(0, 16635) 1

(0, 38953) 1

(0, 14434) 1

(0, 16777) 1

(0, 36134) 1

(0, 35030) 1

(0, 46819) 1

(0, 12870) 1

(0, 58727) 1

(0, 22787) 1

(0, 22197) 2