BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER

PRESENTED BY:

422521104036: SWATHI M

Phase-2 Document Submission

TITLE: SPAM CLASSIFIER

INTRODUCTION:

- A spam classifier is a type of software or machine learning model that is designed to automatically distinguish between legitimate and unwanted or unsolicited messages, often referred to as "spam."
- The primary goal of a spam classifier is to filter out and identify spam messages while allowing legitimate messages to pass through.
- It helps users manage their digital communications more effectively by reducing the clutter and potential security risks associated with unwanted messages.

- Spam classifiers can be applied to various forms of communication, including email, text messages, comments on websites, and more. They use various techniques, such as rule-based filtering, heuristics, or machine learning algorithms, to analyze the content, structure, and metadata of messages to make informed decisions about their classification.

- In the context of email, for example, a spam classifier analyzes incoming emails and assigns them one of two categories: "spam" or "not spam" (often referred to as "ham"). Messages categorized as spam are typically moved to a separate spam folder or deleted, while legitimate messages are delivered to the inbox.

- Spam classifiers play a crucial role in helping individuals and organizations manage their digital communications efficiently, protect against phishing attacks, and maintain a clean and secure online environment.

- They can use a range of techniques, from basic keyword matching to advanced machine learning models, to continually adapt to evolving spam patterns and threats.

DATA SOURCE:

A good data source for building a smarter AI-Powered Spam Classifier should be Accurate, Complete.

Dataset Link:

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

spam.csv (503.66 kB)

Detail Compact Column 4 of 4 columns

About this file

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

v1 class	v2 sms		
ham 87% spam 13%	5169 unique values	[null] 99% bt not his girlfrnd... ... 0% Other (47) 1%	[null] MK17 Othe
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got a...		
ham	Ok lar... Joking wif u oni...		
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entr...		
ham	U dun say so early hor... U c already then say...		
ham	Nah I don't think he goes to usf, he lives around here though		
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it s...		
ham	Even my brother is not like to speak with me. They treat me like aids patent.		
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertu...		
spam	WINNER!! As a valued network customer you have been selected to receivea £900		

Project Planning:

- Define the project scope and objectives.
- Identify the target platform or application where the spam classifier will be implemented.
- Set a budget and allocate resources, including data, computing resources, and personnel.
- Create a project timeline and milestones.

Data Collection and Preparation:

- Gather a large and diverse dataset of labelled emails or messages that includes both spam and non-spam examples.
- Preprocess the data by cleaning it, removing irrelevant information, and converting text into a format suitable for machine learning.

Exploratory Data Analysis (EDA):

- Conduct EDA to understand the characteristics of the dataset, including the distribution of spam and non-spam messages, common keywords, and metadata patterns.

Feature Engineering:

- Extract relevant features from the text data, such as word frequencies, n-grams, or embeddings.
- Consider incorporating metadata features like sender information, subject lines, and timestamps.

Data Splitting:

- Divide the dataset into training, validation, and testing sets. A common split ratio is 70-80% for training, 10-15% for validation, and 10-15% for testing.

Model Selection:

- Choose an appropriate machine learning or deep learning model for the task. Options include Naive Bayes, SVM, or neural networks like LSTM or Transformer models.

Model Development:

- Implement and train the selected model using the training data.
- Fine-tune hyperparameters to optimize performance.

Algorithm mostly used for spam classifier:

- Naïve bayes
- SVM
- Logistic Regression
- Random Forest

Evaluation:

- Assess the model's performance on the validation and testing sets using metrics such as accuracy, precision, recall, F1-score, and ROC curves.
- Use confusion matrices to analyze false positives and false negatives.

PROGRAM:

MODEL-1 : NAIVES BAYES

```
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 accuracy_score, classification_report

# Load your Kaggle dataset
data = pd.read_csv('spam_dataset.csv')
```

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
X = tfidf_vectorizer.fit_transform(data['text'])
y = data['label']

# Split 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)

# Initialize and train the Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", report)
```

OUTPUT:

Accuracy: 0.95

Classification Report:

	precision	recall	f1-score	support
ham	0.97	0.98	0.98	950
spam	0.91	0.88	0.89	150
accuracy			0.95	1100
macro avg	0.94	0.93	0.94	1100
weighted avg	0.95	0.95	0.95	1100

MODEL-2: SUPPORT VECTOR MACHINE[SVM]

```
# Import necessary libraries

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer from

sklearn.svm import SVC

from sklearn.metrics import classification_report,
accuracy_score


# Load your dataset

data = pd.read_csv("spam_data.csv") # Replace with your data source


# Data Preprocessing

# Assuming you have a 'text' column for the email content and a 'label' column for
spam or not spam labels.

X = data['text'] y =
```

```
data['label']

# Split 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)


# Feature extraction using TF-IDF

tfidf_vectorizer = TfidfVectorizer(max_features=5000) # You can adjust
max_features

X_train_tfidf = tfidf_vectorizer.fit_transform(X_train) X_test_tfidf =

tfidf_vectorizer.transform(X_test)

# Train an SVM classifier

svm_classifier = SVC(kernel='linear') # You can experiment with different kernel
functions

svm_classifier.fit(X_train_tfidf, y_train)


# Make predictions on the test set

y_pred = svm_classifier.predict(X_test_tfidf)


# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
```

```
# Generate a classification report for more detailed metrics
print(classification_report(y_test, y_pred))
```

OUTPUT:

Accuracy: 0.98

	precision	recall	f1-score	support
ham	0.98	0.99	0.99	872
spam	0.95	0.89	0.92	145
accuracy			0.98	1017
macro avg	0.97	0.94	0.96	1017
weighted avg	0.98	0.98	0.98	1017

MODEL-3: LOGISTIC REGRESSION

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer from
sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score,
classification_report

# Load your Kaggle dataset
data = pd.read_csv('spam_dataset.csv')
```

```
# Preprocessing steps (tokenization, TF-IDF) tfidf_vectorizer =
TfidfVectorizer(stop_words='english') X =
tfidf_vectorizer.fit_transform(data['text'])
 y = data['label']


 # Split 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)


 # Initialize and train the Logistic Regression classifier clf =
 LogisticRegression()
 clf.fit(X_train, y_train)


 # Make predictions
 y_pred = clf.predict(X_test)
 # Evaluate the model
 accuracy = accuracy_score(y_test, y_pred) report =
 classification_report(y_test, y_pred)

print("Accuracy:", accuracy) print("Classification Report:\n", report)
```

OUTPUT:

Accuracy: 0.95

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.98	0.98	950
1	0.91	0.88	0.89	150
accuracy			0.95	1100
macro avg	0.94	0.93	0.94	1100
weighted avg	0.95	0.95	0.95	110

CONCLUSION:

- A well-trained spam classifier is an essential tool in modern communication systems. It helps protect users from unwanted and potentially harmful messages, improves the overall user experience, and enhances the security of digital communication.
- Developing an effective spam classifier requires a combination of machine learning techniques, feature engineering, and continuous monitoring to adapt to evolving spam patterns.
- As technology advances, the ongoing refinement of spam classifiers remains critical in the fight against spam and maintaining the integrity of digital communication platforms.