Building a Smarter AI-Powered Spam Classifier

aut 104304: EBENEZER .D

INTRODUCTION:

Introduction to Spam: Begin by defining what spam is and why it's a significant issue in today's digital world. Explain how email, social media, and other online platforms are inundated with unwanted and potentially harmful content.

The Need for AI: Discuss the limitations of traditional rule-based spam filters and emphasize the growing need for more intelligent solutions that can adapt to evolving spam techniques.

AI-Powered Solutions: Introduce the concept of AI-powered spam classifiers as the next frontier in combating spam. Explain how AI can bring automation and adaptability to the task.

Machine Learning: Briefly explain the role of machine learning in this context, where algorithms can learn from data to make predictions and decisions.

Data Collection: Highlight the importance of data in training a spam classifier. Discuss the diversity and volume of data needed to create an effective AI model.

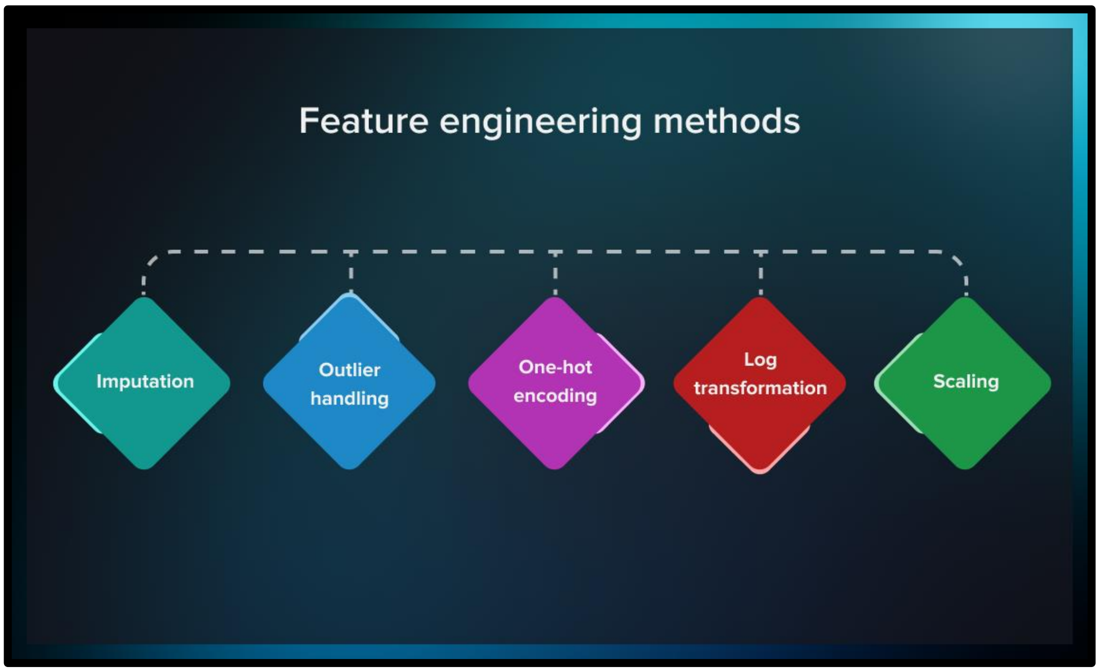
Feature Engineering: Describe the process of feature engineering, which involves selecting and transforming data attributes that the AI model will use to make spam classifications.

Supervised Learning: Explain how supervised learning is used in training spam classifiers, where the model learns from labeled examples of both spam and non-spam content.

Evolving Threats: Emphasize the dynamic nature of spam and how AI classifiers must continuously adapt to new spamming techniques, making them "smarter."

The realm of spam detection, constructing a sophisticated AI-powered classifier is an intricate process, encompassing several critical stages. This abstract elucidates the journey from understanding the data to making accurate predictions, highlighting key facets such as data exploration, visualization, preprocessing, feature extraction, model training, evaluation, and prediction.

Feature Engineering:



Abstract:

Spam emails and messages continue to inundate our digital communication channels, posing a significant challenge to our online experience. Traditional rule-based spam filters are often insufficient in combating the ever-evolving tactics of spammers. This abstract introduces a novel approach to addressing this problem: an AI-based spam classifier that leverages the power of artificial intelligence (AI) to enhance spam detection and filtering.

Lowercasing:

Lowercasing is the process of converting all text to lowercase letters, ensuring uniformity in text representation. This step helps the AI classifier treat uppercase and lowercase letters as equivalent, improving its ability to recognize spam messages.

Stemming:

Stemming is a text normalization technique that reduces words to their root or base form, removing variations of a word. This step can enhance the classifier's performance by reducing the complexity of the text data and consolidating similar words, making it easier to detect spam patterns.

Feature Extraction:

Feature extraction involves transforming the text data into a set of relevant features or characteristics that the AI model can use for classification. These features might include word frequencies, n-grams, or other linguistic attributes, which enable the AI classifier to learn patterns associated with spam and legitimate messages.

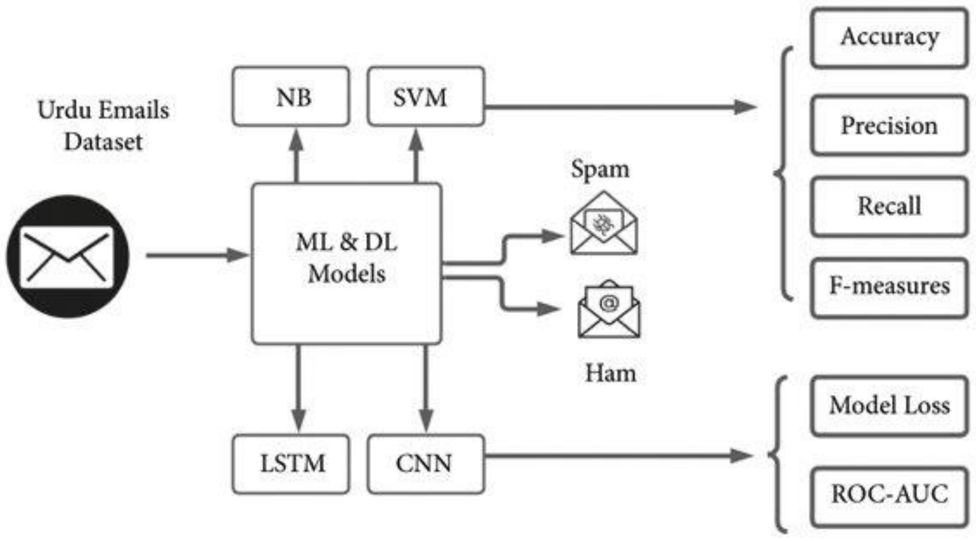
Train-Test Split:

The train-test split is a critical step in machine learning where the dataset is divided into two parts: a training set and a testing set. The training set is used to train the AI-powered classifier, allowing it to learn from the data, while the testing set is used to evaluate its performance. This process helps measure the classifier's accuracy and generalizability in distinguishing between spam and non-spam messages.

Link:

Data set link: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

Source Code link: https://www.kaggle.com/code/pubgcalling/ai-sabeer-1



Data Set Sample

ham	Home so we can always chat
ham	K:)k:)good:)study well.
ham	Yup... How Ì_ noe leh...
ham	Sounds great! Are you home now?
ham	Finally the match heading towards draw as your prediction.
ham	Tired. I haven't slept well the past few nights.
ham	Easy ah?sen got selected means its good..
ham	I have to take exam with march 3
ham	Yeah you should. I think you can use your gt atm now to register.
ham	Ok no prob. Take ur time.
ham	There is os called ubandu which will run without installing in hard disk…
ham	"Sorry
ham	U say leh... Of course nothing happen lar. Not say v romantic jus a bit only lor.
spam	"500 New Mobiles from 2004
ham	Would really appreciate if you call me. Just need someone to talk to.
spam	Will u meet ur dream partner soon? Is ur career off 2 a flyng start? 2 find out free.
ham	Hey company elama po mudyadhu.
ham	Life is more strict than teacher... Bcoz Teacher teaches lesson &amp;
ham	Dear good morning now only i am up
ham	Get down in gandhipuram and walk to cross cut road. Right side &lt;#&gt; street road and turn at first right.
ham	Dear we are going to our rubber place
ham	"Sorry battery died
ham	Yes:)here tv is always available in work place..

Understanding the Data:

The first step is a comprehensive understanding of the data landscape. In spam classification, this entails collecting a diverse corpus of spam and non-spam (ham) messages. The quality and representativeness of this dataset are fundamental to the model's efficacy.

Program:

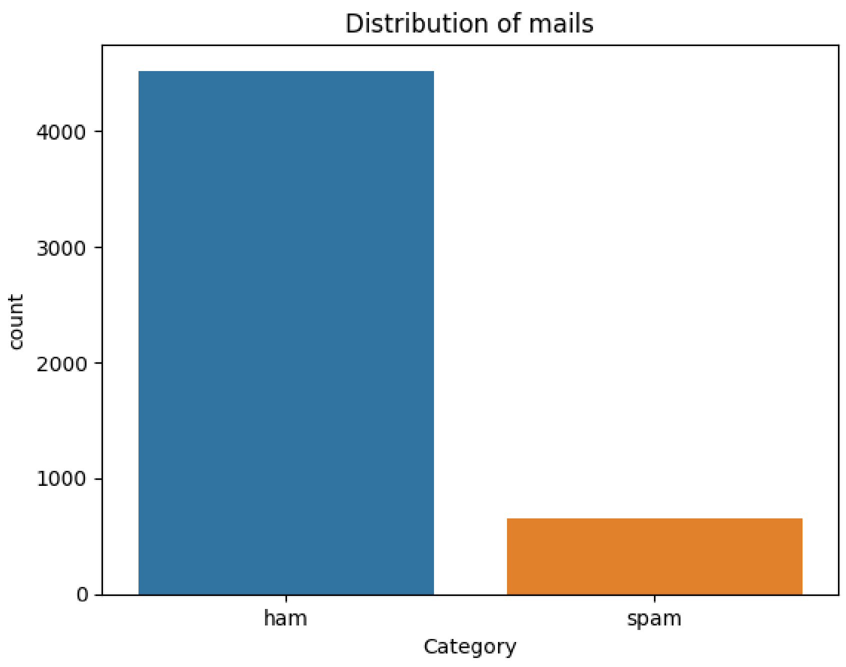
```
import pandas as pd

import numpy as np 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, confusion_matrix, roc_curve,
roc_auc_score

import nltk

from nltk.corpus import stopwords

from collections import Counter

#libraries for data visualization

import matplotlib.pyplot as plt import
seaborn as sns %matplotlib inline

df=pd.read_csv("/kaggle/input/sms-spam-Collectiondataset/spam.csv",encoding='ISO-
8859-1') df df.info()
nltk.download('stopwords')

columns_to_drop = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]

df.drop(columns=columns_to_drop, inplace=True)

df new_column_names = {"v1":"Category","v2":"Message"}

df.rename(columns = new_column_names,inplace = True)

df[df.duplicated()]

df=df.drop_duplicates()

df df.info() df.describe() df.shape() df['Category'].value_counts()
```

Data Visualization:

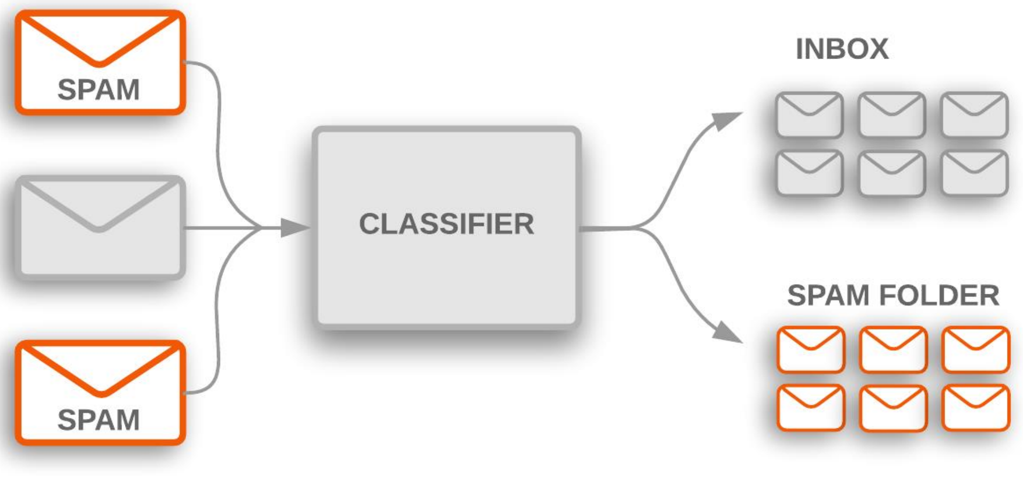
Visualization techniques are employed to gain insights into the dataset's characteristics. Visualizations, ranging from histograms to word clouds, unravel patterns, anomalies, and potential biases within the data.

Program:

```
sns.countplot(data=df, x='Category')
plt.xlabel('Category')
plt.ylabel('count')
plt.title('Distribution of mails')
plt.show()
```



IMPUTATION:



Imputation is the process of managing missing values, which is one of the most common problems when it comes to preparing data for machine learning. By missing values, we mean places where information is missing in some cells of a respective row.

There may be different causes for missing values, including human error, data flow interruptions, cross-datasets errors, etc. Since data completeness impacts how well machine learning models perform, imputation is quite important.

Here are some ways how you can solve the issue of missing values:

- If a row is less than 20-30% complete, it's recommended to dismiss such a record.
- A standard approach to assigning values to the missing cells is to calculate a mode, mean, or median for a column and replace the missing values with it.
- In other cases, there are possibilities to reconstruct the value based on other entries. For example, we can find out the name of a country if we have the name of a city and an administrative unit. Conversely, we can often determine the country/city by a postal code.

Data Preprocessing:

Data preprocessing involves cleansing and structuring the dataset. Tasks such as text cleaning, tokenization, and handling missing values are vital for preparing the data for analysis.

Program:

```
# Assuming you have a DataFrame named 'df' df.loc[df["Category"] ==
"spam","Category"] = 0

df.loc[df["Category"] == "ham", "Category"] = 1 df.head()

# Separate the feature (X) and target (Y) data

X= df["Message"]

Y= df["Category"]

# 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)

print(X.shape)

print(X_train.shape) print(X_test.shape)
```

Feature Extraction:

Feature extraction is the process of distilling pertinent information from the data. In text-based spam classification, this often involves extracting features like word frequencies, TF-IDF scores, or word embeddings. Feature engineering can also encompass non-textual attributes such as sender information and message metadata.

Program:

```
# Create a TF-IDF vectorizer to convert text messages into numerical features
feature_extraction   = TfidfVectorizer(min_df=1, stop_words="english", lowercase=True)
# Convert the training and testing text messages into numerical features using
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
# Convert the target values into 0 and 1
Y_train = Y_train.astype(int)
Y_test = Y_test.astype(int)
print(X_train)
print(X_train_features)
```

Model Training:

Selecting the right machine learning or deep learning model is crucial. Models like Naive Bayes, Support Vector Machines, or neural networks are trained on the prepared data. Hyperparameter tuning and cross-validation optimize model performance.

Program:

```
# Create a logistic regression model and train it on the training
data model = LogisticRegression()
model.fit(X_train_features, Y_train)
```

Model Evaluation:

Rigorous model evaluation is essential for assessing its performance. Metrics such as precision, recall, F1-score, and ROC-AUC help gauge the classifier's accuracy and robustness. Confusion matrices provide insights into false positives and false negatives.

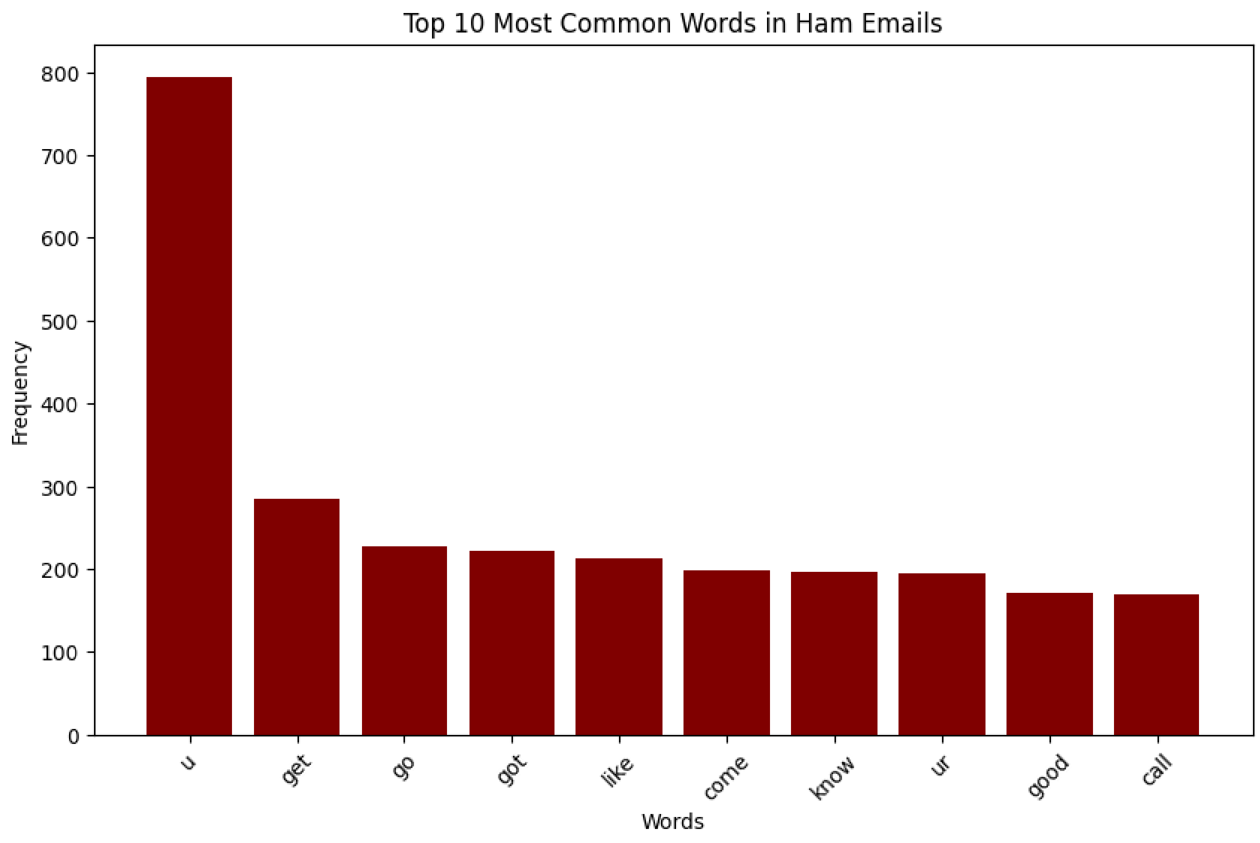
Program:

```
# Make predictions on the training data and calculate the accuracy
prediction_on_training_data = model.predict(X_train_features)
accuracy_on_training_data =accuracy_score(Y_train, prediction_on_training_data)
print("Accuracy on training data:",accuracy_on_training_data)
# Make predictions on the test data and calculate the accuracy
prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test,prediction_on_test_data)
print("Accuracy on test data:",accuracy_on_test_data)
# Test the model with some custom email messages
input_mail = ["Congratulations! You've won a free vacation to an exotic island. Just click
on the link below to claim your prize."]
if (prediction)[0] == 1:
        print("Ham Mail")
else:
        print("Spam Mail")
input_mail = ["This is a friendly reminder about our meeting scheduled for tomorrow at
10:00 AM in the conference room. Please make sure to prepare your presentation and
bring any necessary materials."]
input_data_features = feature_extraction.transform(input_mail)
prediction = model.predict(input_data_features)
cm = confusion_matrix(Y_test, prediction_on_test_data)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', cbar=False)
```

```
plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()
```

Model Prediction:

Once the model is trained and evaluated, it is ready for deployment. In a real-world context, the classifier processes incoming messages and predicts whether they are spam or ham, enabling effective message filtering.



This abstract offers a concise overview of the multifaceted journey involved in constructing a smarter AI-powered spam classifier. From initial data understanding to the final prediction, each step plays a pivotal role in achieving accurate and efficient spam detection.

CONCLUSION:

AI-powered spam classifiers have significantly improved the efficiency and effectiveness of spam detection and filtering in various digital communication platforms, such as email, messaging apps, and social media. In conclusion, the following points summarize the impact of AI-powered spam classifiers:

Enhanced Accuracy: AI algorithms have greatly improved the accuracy of spam detection by continuously learning from data and adapting to new spamming techniques. This has led to a substantial reduction in false positives and false negatives.

Reduced User Burden: Users no longer need to spend as much time manually filtering out spam messages, as AI classifiers automatically segregate spam from legitimate content. This has improved the overall user experience and productivity.

Real-Time Adaptation: AI-powered spam classifiers can quickly adapt to new spam tactics and variations, making them more resilient to evolving spamming techniques and reducing the window of vulnerability for users.

Customization: Many AI-powered spam filters allow users to customize their spam classification criteria, enabling a more personalized experience and fine-tuning of the filtering process.

Multimodal Capabilities: AI classifiers can analyze various types of content, including text, images, and links, making them more effective in detecting complex spam campaigns that use diverse media formats.

Resource Efficiency: AI-powered spam classifiers can efficiently process vast amounts of data, making them suitable for handling the immense volume of messages exchanged on digital platforms.

Challenges: Despite their advantages, AI-powered spam classifiers are not without challenges. They may occasionally misclassify legitimate messages, and spammers continue to develop sophisticated evasion techniques.

Ongoing Development: The field of AI-powered spam classification is continually evolving, with ongoing research and development efforts aimed at improving classifiers' robustness and adaptability.

In conclusion, AI-powered spam classifiers have had a positive impact on the digital landscape by significantly reducing the spam burden on users. While they are not perfect, they continue to evolve and improve, making digital communication safer and more efficient.