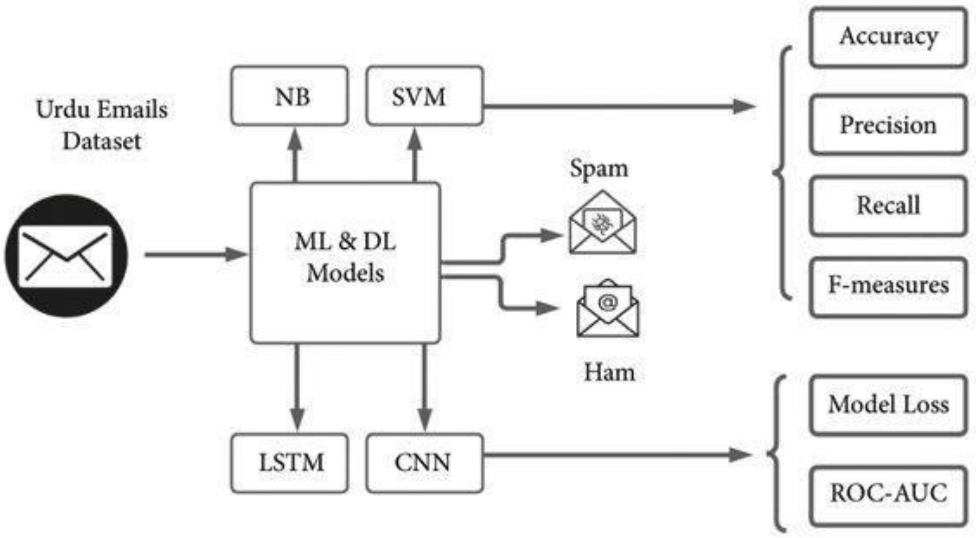
Building a Smarter AI-Powered Spam Classifier

952721104023: SIVARANJAN. R

In 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.



Link:

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

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 &
ham	Dear good morning now only i am up
ham	Get down in gandhipuram and walk to cross cut road. Right side <#> 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

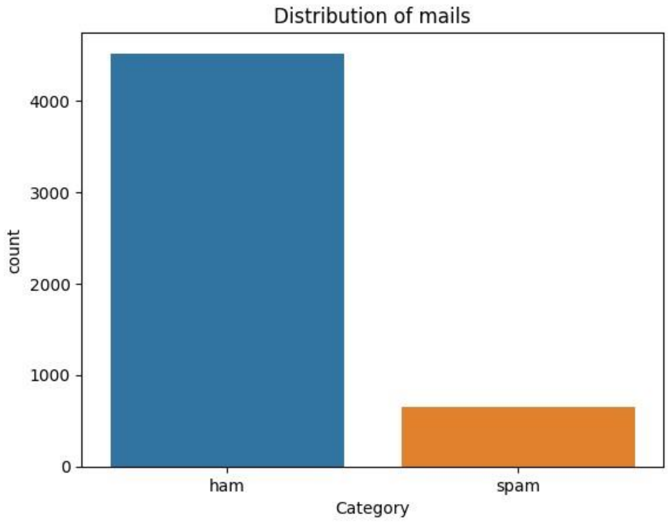
```
import matplotlib.pyplot as plt

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()
```



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"]

X
Y

# 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
TF-IDF

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.

Model Prediction:

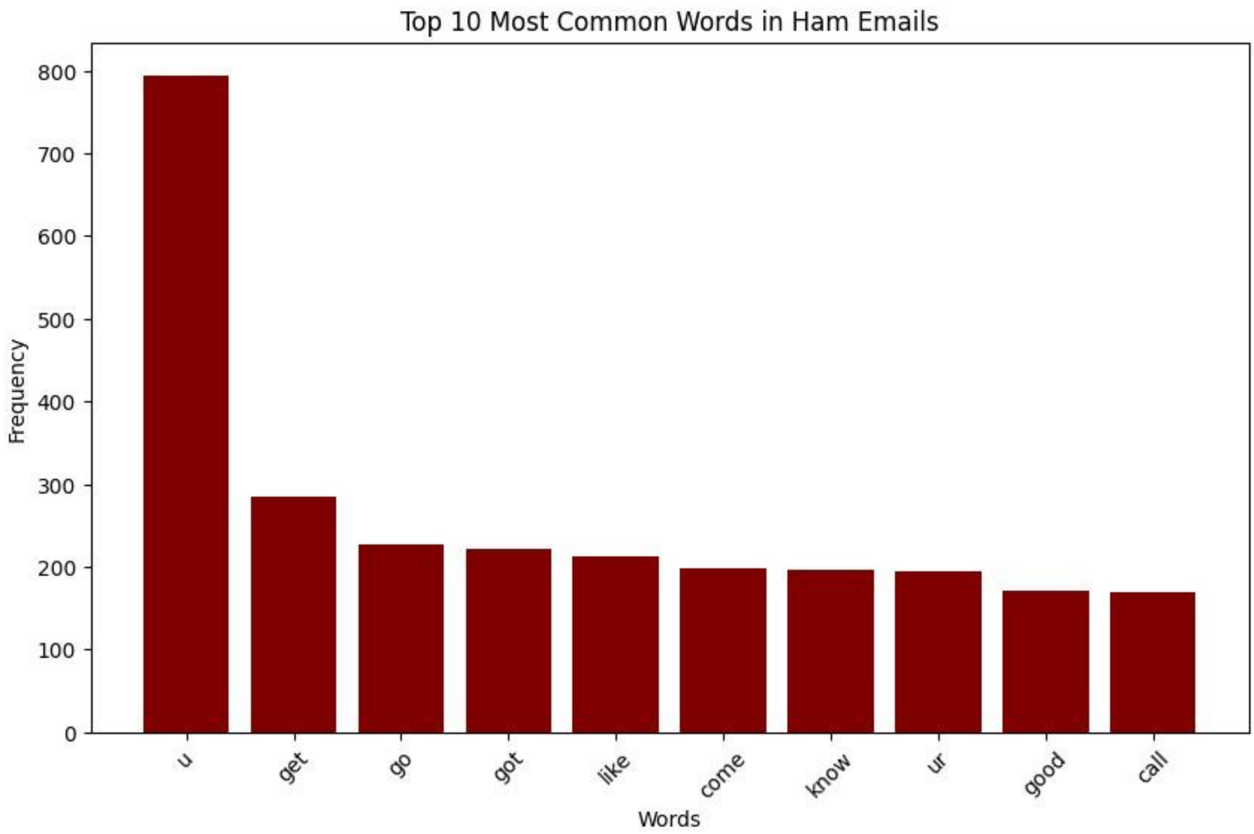
Once the model is trained and evaluated, it is ready for deployment. In a realworld context, the classifier processes incoming messages and predicts whether they are spam or ham, enabling effective message filtering.

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."] input_data_features =
```

```
feature_extraction.transform(input_mail) prediction = model.predict(input_data_features)

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)

if (prediction)[0] == 1:
print("Ham Mail") else:
   print("Spam Mail")
# Data visualization - Confusion Matrix

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()
# Data visualization - Top 10 Most Common Words in Spam Emails

stop_words = set(stopwords.words('english')) spam_words = ''
''.join(df[df['Category'] == 0]['Message']).split() ham_words = ''
''.join(df[df['Category'] == 1]['Message']).split()
```

```
spam_word_freq = Counter([word.lower() for word in spam_words if
word.lower() not in stop_words and word.isalpha()])


plt.figure(figsize=(10, 6))

plt.bar(*zip(*spam_word_freq.most_common(10)), color='g')

plt.xlabel('Words') plt.ylabel('Frequency') plt.title('Top 10

Most Common Words in Spam Emails')

plt.xticks(rotation=45) plt.show()

# Data visualization - Top 10 Most Common Words in Ham Emails


ham_word_freq = Counter([word.lower() for word in ham_words if

word.lower() not in stop_words and word.isalpha()])
plt.figure(figsize=(10, 6))

plt.bar(*zip(*ham_word_freq.most_common(10)), color='maroon')

plt.xlabel('Words') plt.ylabel('Frequency')

plt.title('Top 10 Most Common Words in Ham Emails')

plt.xticks(rotation=45) plt.show()
```



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.