Building a Smarter AI-Powered Spam Classifier

NAME : S.SAFREEN

DEPT & YEAR : ECE / IIIRD

E-MAIL ID : safreensheriff7@gmail.com

NM ID : au513521106029

COLLEGE NAME : AMCET

REG-NO : 513521106029

PROBLEM STATEMENT

The objective of this project is to develop an AIpowered spam classifier capable of accurately distinguishing between spam and non-spam messages in email and text messages. The primary challenge lies in reducing the occurrences of false positives, where legitimate messages are incorrectly classified as spam, and false negatives, where actual spam messages are missed, while maintaining a high level of overall accuracy. The spam classifier will enhance the efficiency and security of communication channels by ensuring that unwanted and potentially harmful content is reliably filtered out.

PROJECT OVERVIEW

The primary objective of this project will be to develop a Artificial Intelligence model that will accurately classify spam and non-spam messages based on a set of relevant features

Dataset Source: We will acquire our dataset from Kaggle, specifically the “SMS Spam Collection Dataset” dataset.

Dataset Link: https://www.kaggle.com/datasets/uciml/smsspamcollection-dataset

TEAM MEMBERS

- SAFREEN.S
- MAHALAKSHMI.S
- LOKESWARI.R
- KEERTHANA.R
- LOGESHWARI.S

INTRODUCTION

Spam emails and messages are a pervasive problem in today's digital world. To combat this issue effectively, we need a smarter AI-powered spam classifier that can identify and filter out unwanted and potentially harmful content. This document outlines the step-by-step process of building such a spam classifier.

Project Phases

1. Data Collection and Preprocessing

Data Gathering:

Start by collecting a labeled dataset of spam and non-spam (ham) messages. It's essential to have a diverse and representative dataset to train an effective model. You can find publicly available datasets or create your own by manually labeling messages.

Data Preprocessing:

Clean the collected data by removing special characters, irrelevant information, and any formatting issues. Normalize the text data by converting it to lowercase and removing common stopwords (non-informative words).

2. Feature Engineering:

Creating Features: Feature engineering is a crucial step in building an effective spam classifier. Some common features to consider include:

- **Word frequency:** Calculate the frequency of each word in the text.
- **Character-based features:** Count special characters, uppercase letters, and digits.

- **Words-based features:** Count the number of words , length of the words.
- **Sentences-based features:** Sentence are tokenized .

```
 #1. Total No of Char

df['num_char']=df['text'].apply(len)
```

```
#2. Total No of Words
df['num_words']=df['text'].apply(lambda x: len(str(x).split()))
```

```
#3. Total No of Sentences
nltk.download('punkt')
df['num_sen']=df['text'].apply(lambda x: len(nltk.sent_tokenize(x)))
```

	label	text	num_char	num_words	num_sen
0	ham	Go until jurong point, crazy.. Available only ...	111	20	2
1	ham	Ok lar... Joking wif u oni...	29	6	2
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	28	2
3	ham	U dun say so early hor... U c already then say...	49	11	1
4	ham	Nah I don't think he goes to usf, he lives aro...	61	13	1
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	161	30	4
5568	ham	Will Ì_ b going to esplanade fr home?	37	8	1
5569	ham	Pity, * was in mood for that. So...any other s...	57	10	2
5570	ham	The guy did some bitching but I acted like i'd...	125	26	1
5571	ham	Rofl. Its true to its name	26	6	2

STATISTICAL ABOUT DATASET:

```
#Statistical Info about dataset
df.describe()
```

num_char	num_words	num_sen	
count	5572.000000	5572.000000	5572.000000
mean	80.118808	15.494436	1.996411
std	59.690841	11.329427	1.520159
min	2.000000	1.000000	1.000000
25%	36.000000	7.000000	1.000000
50%	61.000000	12.000000	1.500000
75%	121.000000	23.000000	2.000000
max	910.000000	171.000000	38.000000

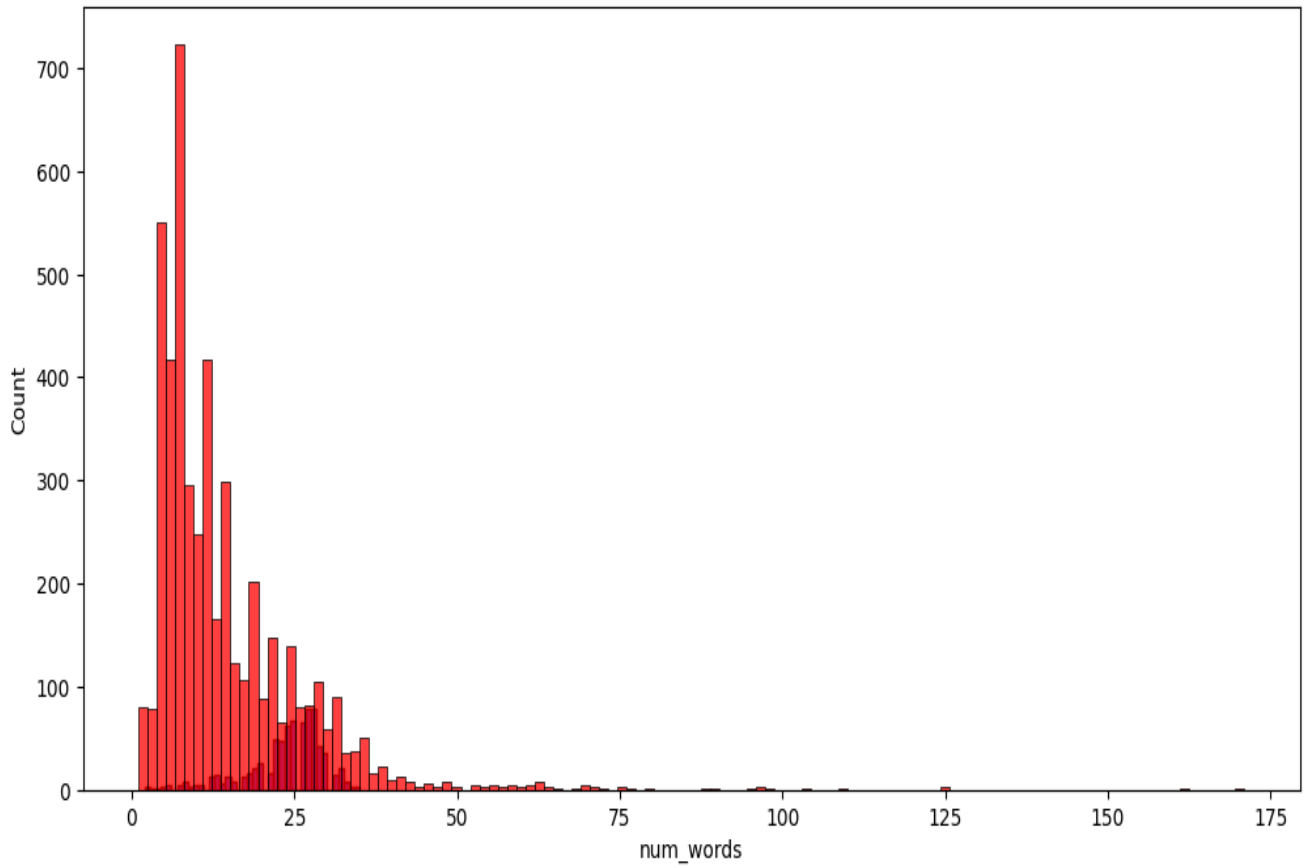
```
df[df['label']=='ham'].describe()
```

	num_char	num_words	num_sen
count	4825.000000	4825.000000	4825.000000
mean	71.023627	14.200622	1.837720
std	58.016023	11.424511	1.454388
min	2.000000	1.000000	1.000000
25%	33.000000	7.000000	1.000000
50%	52.000000	11.000000	1.000000
75%	92.000000	19.000000	2.000000
max	910.000000	171.000000	38.000000

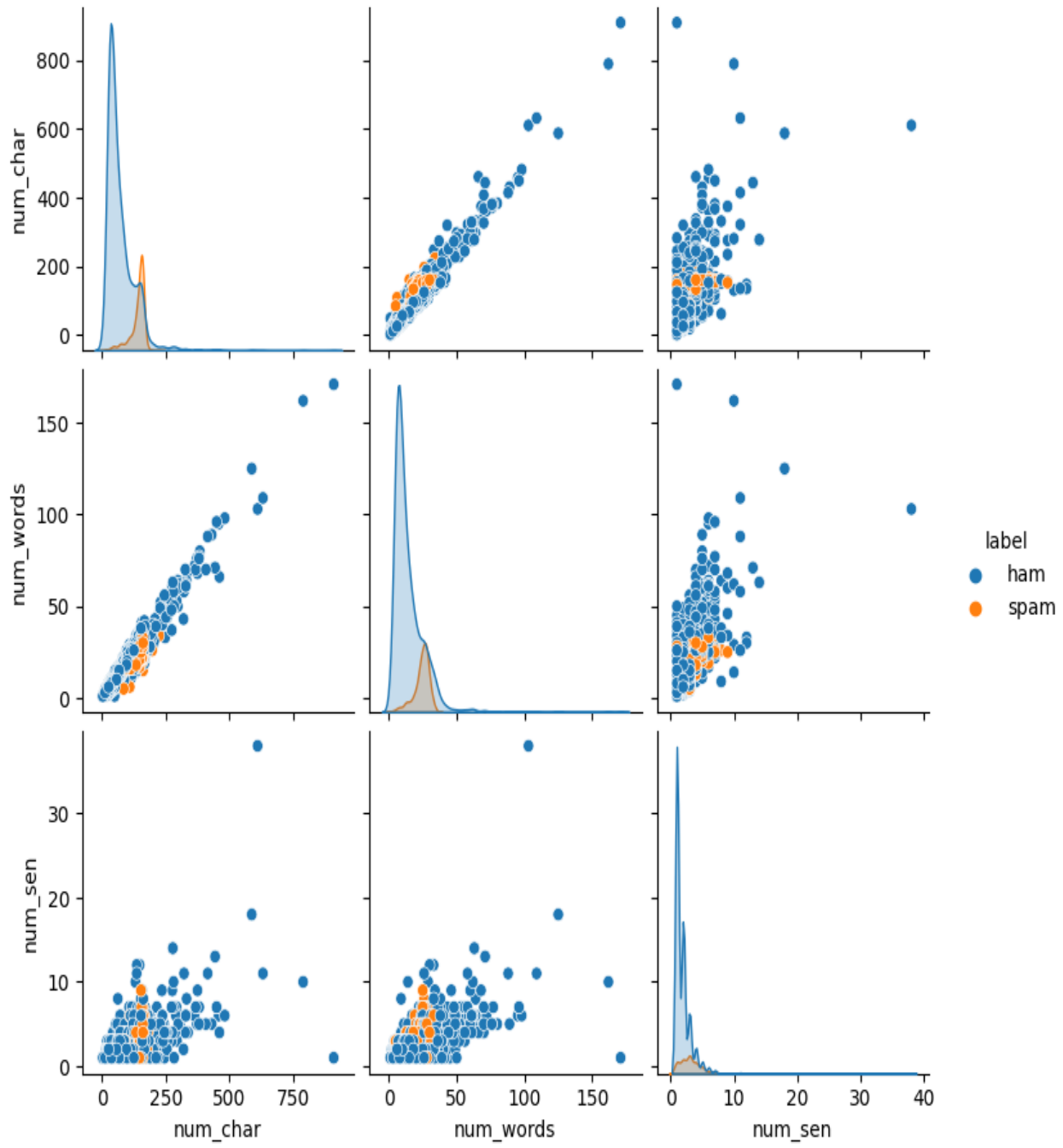
```
df[df['label']=='spam'].describe()
```

	num_char	num_words	num_sen
count	747.000000	747.000000	747.000000
mean	138.866131	23.851406	3.021419
std	29.183082	5.811898	1.537580
min	13.000000	2.000000	1.000000
25%	132.500000	22.000000	2.000000
50%	149.000000	25.000000	3.000000
75%	157.000000	28.000000	4.000000
max	224.000000	35.000000	9.000000

```
#Hist Plot for Spam and Not Spam
plt.figure(figsize=(12,6))
sns.histplot(df[df['label']=='spam']['num_words'],color='blue',bins=40)
sns.histplot(df[df['label']=='ham']['num_words'],color='red')
plt.show()
```



```
sns.pairplot(df,hue='label')
plt.show()
```



3.Model Preprocessing:

- Train your selected model using the training data and the engineered features. Experiment with different hyperparameters and architectures to optimize model performance.

```
#Import lib required for text processing
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords.words('english')
from nltk.stem import PorterStemmer
from wordcloud import WordCloud
import string,time
string.punctuation
```

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

```
#punctuation
string.punctuation
```

'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

```
#Stopwards
stopwords.words('english')
```

Spam emails continue to be a nuisance in the digital age, inundating our inboxes with unwanted and potentially harmful content. To effectively combat this issue, machine learning and artificial intelligence (AI) models are being deployed as email spam classifiers. These models help distinguish between legitimate (ham) and unsolicited (spam) messages, ensuring that only relevant and safe emails reach our inboxes. However, before feeding emails into AI models, thorough preprocessing is essential to optimize the performance of the classifier

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
```

'it',

"it's",

'its',

'itself',

'they',

'them',

'their',

'theirs',

'themselves',

'what',

'which',

'who',

'whom',

'this',

'that',

"that'll",

'these',

'those',

'am',

'is',

'are',

'was',

'were',

'be',

'been',

'being',

And many other stopwords in English.

```
### Clean or Handle Text Data
1. Remove **Web Links**
2. Remove **Numbers**
3. Remove **Emails**
```

```
#User Define Funtion for Text processing
def remove_website_links(text):
    no_website_links = text.replace(r"http\S+", "")
    return no_website_links

def remove_numbers(text):
    removed_numbers = text.replace(r'\d+','')
    return removed_numbers

def remove_emails(text):
    no_emails = text.replace(r"\S*@\S*\s?",'')
    return no_emails
```

```
#Call Function
df['text'] = df['text'].apply(remove_website_links)
df['text'] = df['text'].apply(remove_numbers)
df['text'] = df['text'].apply(remove_emails)
```

Create Common function to Clean or process Text Data

1. **Lower casing** to avoids duplicates
2. **Tokenization** sentences
3. Remove **Specials characters**
4. Remove **Stopwords**
5. Remove **punctuation**
6. **Stemming.**

```
#Create Common Function
def transform_text(text):
    #1.lower casing
    text=text.lower()

    #2.tokenization
    lst=nltk.word_tokenize(text)

    #3.remove spcl characters stopwords and punctuation
    l1=[]
    useless_words=stopwords.words('english')+list(string.punctuation)
    for word in lst:
        if word.isalnum()==True and word not in useless_words:
            l1.append(word)

    #4.stemming
    l2=[]
    for word in l1:
        ps=PorterStemmer()
        l2.append(ps.stem(word))

    return " ".join(l2).strip()
    l1.clear()
    l2.clear()
```

```
#call function
nltk.download('punkt')
df['text'] = df['text'].apply(transform_text)
df['num_words_transform']=df['text'].apply(lambda x: len(str(x).split()))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
df
```

label	text	num_char	num_words	num_sen	num_words_transform	
0	ham	go jurong point crazi avail bugi n great world...	111	20	2	16
1	ham	ok lar joke wif u oni	29	6	2	6
2	spam	free entri 2 wkli comp win fa cup final tkt 21...	155	28	2	24
3	ham	u dun say earli hor u c alreadi say	49	11	1	9
4	ham	nah think goe usf live around though	61	13	1	7
...	...	...	...	...	...	...
5561	spam	2nd time tri 2 contact u pound prize 2 claim e...	161	30	4	17
5562	ham	b go esplanad fr home	37	8	1	5
5563	ham	piti mood suggest	57	10	2	3
5564	ham	guy bitch act like interest buy someth els nex...	125	26	1	13
5565	ham	rofl true name	26	6	2	3

5566 rows × 6 columns

4.Model Evaluation

```
#Common Function for model train
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)
    train_accuracy = clf.score(X_train,y_train)

    return accuracy,precision,train_accuracy
```

```
svc=SVC(kernel='sigmoid', gamma=1.0)
train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
(0.9802513464991023, 0.9805128205128205, 0.9849505840071878)
```

```
accuracy_scores = []
precision_scores = []
train_accuracy_score=[]

for name,clf in classifiers.items():

    current_accuracy,current_precision,current_train_score =
train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
    train_accuracy_score.append(current_train_score)
    print()
```

```
For  svc
Accuracy -  0.9802513464991023
Precision -  0.9805128205128205

For  knc
Accuracy -  0.90754039497307
Precision -  0.9030131826741996

For  mnb
Accuracy -  0.9793536804308797
Precision -  0.9775510204081632

For  dtc
Accuracy -  0.933572710951526
Precision -  0.9368213228035538

For  lr
Accuracy -  0.9622980251346499
Precision -  0.9626639757820383

For  rfc
Accuracy -  0.9730700179533214
Precision -  0.9725330620549338

For  adb
Accuracy -  0.966786355475763
Precision -  0.9713701431492843

For  xgb
Accuracy -  0.9748653500897666
Precision -  0.975485188968335

For  gbc
Accuracy -  0.9551166965888689
Precision -  0.954045954045954
```

```
df1=pd.DataFrame({'Algorithm':classifiers.keys(),'Precision':precision_scores,
                  'Test Accuracy':accuracy_scores,'Train
Accuracy':train_accuracy_score}).round(3)
```

```
df2=df1.sort_values(['Precision','Test Accuracy'],ascending=False)
df2
```

	Algorithm	Precision	Test Accuracy	Train Accuracy
0	svc	0.981	0.980	0.985
2	mnb	0.978	0.979	0.983
7	xgb	0.975	0.975	0.985
5	rfc	0.973	0.973	1.000
6	adb	0.971	0.967	0.975
4	lr	0.963	0.962	0.965
8	gbc	0.954	0.955	0.966
3	dtc	0.937	0.934	0.947
1	knc	0.903	0.908	0.927

10. Post-processing:

- Implement post-processing rules to enhance the model's output. This can include setting thresholds on prediction probabilities to classify messages as spam or ham.

CONCLUSION:

- Building a smarter AI-powered spam classifier is an ongoing process that requires a combination of data collection, feature engineering, model training, and rigorous evaluation. By following these steps and continuously improving the model, you can create a robust defense against spam messages and provide a safer digital environment for users.