UNIVERSITY COLLEGE OF ENGINEERING

VILLUPURAM

NAAN MUDHALVAN

PHASE-3 DOCUMENT SUBMISSION

BUILDING A SMARTER AI–POWERED SPAM CLASSIFIER

IBM-ARTIFICIAL INTELLIGENCE

SWATHI M

422521104036

Phase 3 - Development part

CONTENT:

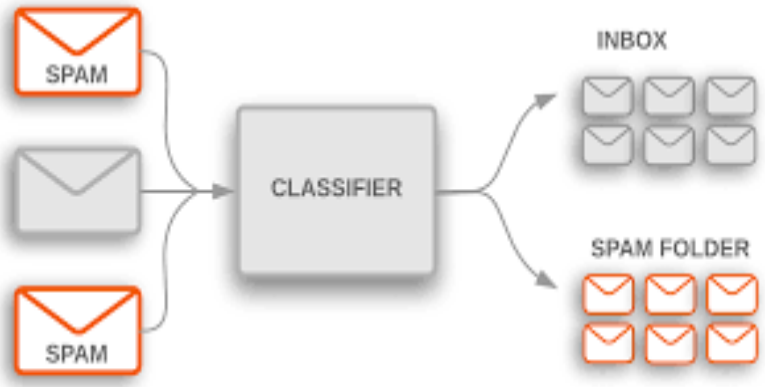
- In this phase,I am building my project by loading and pre-processing the dataset.
- And also start building the core components of spam classifier.
- By using the Kaggle dataset which contains a collection of SMS messages labelled as spam or non-spam.

Dataset Link:

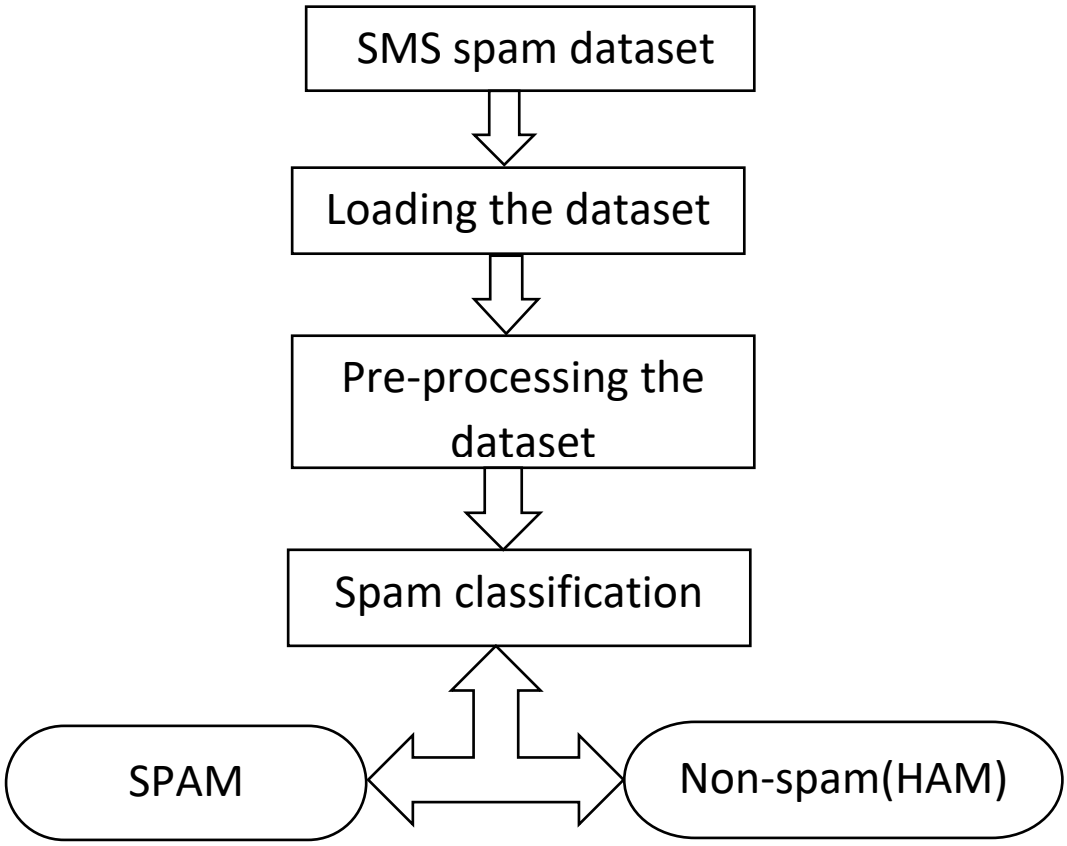
https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

INTRODUCTION:

- SMS is a text-based communication media that allows mobile phone users to share a short text**.**
- Along with the widespread use and popularity as the most important communications media, there are plenty of those who use it for commercial purposes such as advertising media and even fraud.
- The reduced SMS rate is one of the causes of increasing SMS spam as well, as in China, the SMS tariff is well under than $ 0.001.
- Moreover, based on the Korea Information Security (KISA), this amount exceeds the email spam.
- For instance, mobile users in the US gains 1.1 billion SMS spams, and Chinese users receives 8,29 billion SMS spams weekly .
- There is a solution that could be performed to solve the above problems. It is by filtering SMS based on the text classification.
- There are some popular text classifications techniques, including decision trees, Naive Bayes, rule induction, neural network, nearest neighbors, and Support Vector Machine.



OVERVIEW:



SMS Spam Dataset:

An SMS spam dataset is a collection of text messages (short message service, or SMS) that have been categorized as either spam or non-spam (ham).

The main components of an SMS spam dataset include:

1.Text Messages

2.Labels

3.Dataset Size

4.Data Distribution

5.Preprocessing

6.Training and Testing Sets

7.Metadata (Optional)

- ✓ SMS spam datasets are valuable for developing and evaluating spam detection algorithms, which can help filter unwanted or potentially harmful text messages from reaching users' inboxes.

- ✓ Researchers and developers use these datasets to create and test machine learning models, including algorithms based on techniques like natural language processing and classification.

Loading the dataset:

Steps to load the SMS spam dataset:

1.Obtain the Dataset:

First, you need to obtain the SMS spam dataset. I can find SMS spam datasets from kaggle dataset.

Ensure that you have the dataset file in a compatible format (e.g., CSV, TSV, Excel, or plain text).

2. Import Necessary Libraries:

- ✓ pandas for data manipulation,
- ✓ numpy for numerical operations,
- ✓ matplotlib or seaborn for data visualization

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

3.Load the dataset:

To load the dataset into the data structure, there are many methods. The method you use depends on the format of your dataset. For example, if your dataset is in a CSV file, use pd.read_csv() in pandas .

```
dataset = pd.read_csv("sms_spam_dataset.csv")
```

4.Explore the Data:

Once the dataset is loaded, you can explore it to understand its structure and content.

Common exploratory data analysis (EDA) steps include:

- ✓ Displaying the first few rows of the dataset,
- ✓ Checking data statistics,
- ✓ Checking the distribution of spam and ham messages.

```
print(dataset.head())
print(dataset.describe())
print(dataset['label'].value_counts())
```

Preprocessing the dataset:

- ❖ Pre-processing is an essential step when working with SMS spam datasets, especially when preparing the text data for analysis or machine learning.
- ❖ The specific pre-processing steps you need to perform depend on the quality and format of your dataset.

Here are common preprocessing steps for SMS spam datasets:

1.Text cleaning:

- ❖ Remove any irrelevant characters or symbols, such as special characters and punctuation marks.
- ❖ Convert all text to lowercase to ensure uniformity.
- ❖ Remove any extra white space

```
dataset['text'] = dataset['text'].str.replace('[^a-zA-Z\s]', '').str.lower()
```

2.Tokenization:

Tokenization involves splitting text into individual words or tokens. This step is crucial for text analysis.

```
dataset['text'] = dataset['text'].apply(lambda x: x.split())
```

3.Stop Word Removal:

Depending on your analysis goals, you may want to remove common stop words (e.g., "the," "and," "in") from the text data to focus on more meaningful words.

```
stop_words = set(stopwords.words('english'))
dataset['text'] = dataset['text'].apply(lambda x:
[word for word in x if word not in stop_words])
```

4. Stemming or Lemmatization:

- ✓ Reducing words to their base or root form can be helpful for text analysis. Stemming and lemmatization are techniques to achieve this.
- ✓ Choose one based on your preference and the specific requirements of your analysis.

```
stemmer = PorterStemmer()
dataset['text'] = dataset['text'].apply(lambda x:
[stemmer.stem(word) for word in x])
```

5.Vectorization:

- ✓ To use text data in machine learning models, you need to convert it into numerical form.

- ✓ Common methods for text vectorization include Count Vectorization and TF-IDF Vectorization.

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(dataset['text'].apply(lambda x: ' '.join(x)))
```

6.Data Split for Model Building:

- ✓ If the goal is to build a spam classifier, it's important to split the preprocessed data into training and testing sets.
- ✓ This separation allows you to train the model on one portion of the data and evaluate its performance on another, providing an accurate assessment of its effectiveness in identifying spam.

```
X_train, X_test, y_train, y_test = train_test_split(X, dataset['label'], test_size=0.2, random_state=42)
```

Program:

```
import pandas as pd
# Load the SMS Spam Collection Dataset
df = pd.read_csv('spam.csv', encoding='latin-1')
# Keep only the relevant columns (text and label)
df = df[['v1', 'v2']]
# Rename the columns to be more descriptive
df.columns = ['label', 'text']
# Convert labels to binary values (ham: 0, spam: 1)
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
```

 # Preprocessing text data

Example: You can perform text cleaning, such as removing punctuation, converting to lowercase, and tokenization.

 # Here's a simple example of text cleaning and lowercasing

df['text'] = df['text'].str.replace('[^\w\s]', '').str.lower()

 # Print the first few rows of the preprocessed dataset

print(df.head())

OUTPUT:

S.NO	LABEL	TEXT
0	0	go until jurong point crazy available only in ...
1	0	ok lar joking wif u oni
2	1	free entry in 2 a wkly comp to win fa cup fina...
3	0	u dun say so early hor u c already then say
4	0	nah i dont think he goes to usf he lives aroun...

CONCLUSION:

- In summary, loading and preprocessing the dataset are essential steps in developing an effective spam classifier.
- The preprocessing steps are designed to clean and prepare the text data for machine learning, ensuring that the model can effectively

differentiate between spam and non-spam messages.

- Properly preprocessed data is key to building accurate and reliable spam classification models that help protect users from unwanted and potentially harmful messages.