BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER

INTRODUCTION:

Artificial Intelligence (AI) and Machine Learning (ML) are becoming the new tool for developers to create a more efficient and life-changing models brining an intelligence into machines to perform various tasks into business operations and household without help of humans.

And to develop the AI and ML model, a precise training data is required that help algorithms to understand the certain patterns or series of outcomes comes to a given question. And training data can consist texts, images or videos which are mainly labeled to make it recognizable to computer vision and understandable to machines.

TRAINING DATA:

Training data is basically a type of data used for training a new application, model or system through various methods depending on the project's feasibility and requirements. And **training data for AI** or ML is slightly different, as they are labeled or annotated with certain techniques to make it recognizable to computer that helps machines to understand the objects.

Types of Training Data for Machine Learning

In machine learning training data is the key factor to make the machines recognize the objects or certain patterns and make the right prediction when used in real-life. Basically, there are three types of training data used in machine learning model development and each data has its own importance and role in building a ML model.

majority of training data contains the pair of input gathered from the various resources and then organized and annotated with certain techniques with accuracy

STRUCTURED DATA:

Structured data is data that has a standardized format for efficient access by software and humans alike. It is typically tabular with rows and columns that clearly define data attributes. Computers can effectively process structured data for insights due to its quantitative nature.

Structured data examples

Here are examples of structured data systems:

- Excel files
- SQL databases
- Point-of-sale data
- Web form results
- Search engine optimization (SEO) tags
- Product directories
- Inventory control
- Reservation systems

UNSTRUCTURED DATA:

Unstructured data is information that is not arranged according to a preset data model or schema, and therefore cannot be stored in a traditional relational database or RDBMS. Text and multimedia are two common types of unstructured content.

Examples of unstructured data

Unstructured data can be created by people or generated by machines.

Here are some examples of the *human-generated variety*:

- Email: Email message fields are unstructured and cannot be parsed by traditional analytics tools. That said, email metadata affords it some structure, and explains why email is sometimes considered semi-structured data.
- Text files: This category includes word processing documents, spreadsheets, presentations, email, and log files.
- Social media and websites: data from social networks like Twitter, LinkedIn, and Facebook, and websites such as Instagram, photo-sharing sites, and YouTube.
- Mobile and communications data: For this category, look no further than text messages, phone recordings, collaboration software, chat, and instant messaging.
- Media: This data includes digital photos, audio, and video files.

SEMI STRUCTURED DATA:

Semi-structured data is a form of structured data that does not obey the tabular structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Therefore, it is also known as self-describing structure.

Examples of structured data

- Dates and times.
- Cell phone numbers.
- Social security numbers.
- Banking/transaction information.
- Customer names, postal addresses, and email addresses.
- Product prices.
- Serial numbers.

-
    - **Begin Building smarter AI power spam loading and preprocessing the dataset**

```
In[1]:
import numpy as np
 # linear algebra
```

```
import pandas as pd
 # data processing, CSV file I/O (e.g. pd.read_csv)

from nltk.corpus import stopwords import nltk
nltk.download('stopwords')

from sklearn.pipeline import Pipeline

from sklearn.naive_bayes import BernoulliNB , MultinomialNB , GaussianN
B

from sklearn.metrics import accuracy_score

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
for filename in filenames:
print(os.path.join(dirname, filename))
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
/kaggle/input/sms-spam-collection-dataset/spam.csv
```

Understand the spam collection data !

In [2]:

```
filepath = '/kaggle/input/sms-spam-collection-dataset/spam.csv'
data_import = pd.read_csv(filepath , encoding = 'ISO-8859-1')
data_import.head()
```

Out[2]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

- **Preprocessing !**

In [3]:

```
df =  data_import.drop(['Unnamed: 2' , 'Unnamed: 3' , 'Unnamed: 4'] , a
xis1)
df.head()
```

Out[3]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [4]:

```
### Removing stopwords from the feature column.
sw = stopwords.words('english')

def stopword(text) :
txt = [word.lower()
for word in text.split()
if word.lower() not in sw]
return txt

df['v2'] = df['v2'].apply(stopword)
```

```
df.head()
```

Out[4]:

	v1	v2
0	ham	[go, jurong, point,, crazy.., available, bugis...
1	ham	[ok, lar..., joking, wif, u, oni...]
2	spam	[free, entry, 2, wkly, comp, win, fa, cup, fin...
3	ham	[u, dun, say, early, hor..., u, c, already, sa...
4	ham	[nah, think, goes, usf,, lives, around, though]

- **Stemming**

In [5]:

```
from nltk.stem.snowball
import SnowballStemmer
ss = SnowballStemmer("english")
def stemming(text) :
text = [ss.stem(word)
for word in text
if word.split()]
return "".join(text)

df['v2'] = df['v2'].apply(stemming)
```

In [6]:

```
df.head()
```

Out[6]:

	v1	v2
0	ham	gojurongpoint,crazy..availbugingreatworldlaebu...
1	ham	oklar...jokewifuoni...
2	spam	freeentri2wklicompwinfacupfinaltkts21stmay2005...
3	ham	udunsayearlihor...ucalreadisay...
4	ham	nahthinkgoeusf,livearoundthough

In [7]:

```
### TF-IDF { Term Frequency , Inverse Document Frequency }

from sklearn.feature_extraction.text
import TfidfVectorizer

tfid_vect = TfidfVectorizer()

# Extract the tfid representation matrix of the test data.
tfid_matrix = tfid_vect.fit_transform(df['v2'])

print(f"Type :{type(tfid_matrix)} , Matrix at 0 : {tfid_matrix[0]} , Sh
ape : {tfid_matrix.shape}")
```

```
Type :<class 'scipy.sparse._csr.csr_matrix'> , Matrix at 0 :   (0, 1
827)     0.5056391989470028
  (0, 1030)     0.5056391989470028
  (0, 2166)     0.48268727087494234
  (0, 3635)     0.5056391989470028 , Shape : (5572, 12124)
```

In [8]:

```
# Collect sparse matrix into dense

array = tfid_matrix.todense()
```

In [9]:

```
df1 = pd.DataFrame(array)
df1[df1[10]  != 0].head()
```

```
Out[9]:
```

	0	1	2	3	4	5	6	7	8	9	...	12114	12115	12116	12117	12118	12119	12120	12121	12122	12123
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5285	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2 rows × 12124 columns

```
In [10]:
```

```python
df1['v1'] = df['v1']
```

```
In [11]:
```

```python
df1.head()
```

```
Out[11]:
```

	0	1	2	3	4	5	6	7	8	9	...	12115	12116	12117	12118	12119	12120	12121	12122	12123	v1
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	ham
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	ham
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	spam

	0	1	2	3	4	5	6	7	8	9	...	12115	12116	12117	12118	12119	12120	12121	12122	12123	v1
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	ham
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	ham

5 rows × 12125 columns

In [12]:

```
from sklearn.model_selection
    import train_test_split

    features = df1.drop('v1' , axis = 1)
    label = df1['v1']
    x_train , x_test , y_train , y_test = train_test_split(features , label
     , test_size = 0.3)
    print(f"X train shape : {x_train.shape}\nY train shape : {y_train.shape
    }\nX test shape : {x_test.shape}\nY test shape : {y_test.shape}")
    X train shape : (3900, 12124)
    Y train shape : (3900,)
    X test shape : (1672, 12124)
    Y test shape : (1672,)
```

In [13]:

```
    ber_pipe = Pipeline(steps = [( 'ber_model' , BernoulliNB())])

    multi_pipe = Pipeline(steps = [('multi_model' , MultinomialNB())])

 guass_pipe = Pipeline(steps = [('guass_model' , GaussianNB())])
```

In [14]:

```
 def model_evaluation(model) :
        model.fit(x_train , y_train)
y_pred_model = model.predict(x_test)

        acc_score = accuracy_score(y_test , y_pred_model)
    print(f"Accuracy Score of {model[0]} : {acc_score}")
```

```
model_evaluation(ber_pipe)
   model_evaluation(multi_pipe)
model_evaluation(guass_pipe)
   Accuracy Score of BernoulliNB() : 0.8947368421052632
Accuracy Score of MultinomialNB() : 0.9204545454545454
   Accuracy Score of GaussianNB() : 0.46411483253588515
```