1.Building a Smarter AI Powered Spam Classifier

AIM :

To Implement Building Smarter AI Powered Spam Classiffier

Algorithm

Step 1: E-mail Data Collection. The dataset contained in a corpus plays a crucial role in assessing the performance of any spam filter.
Step 2: Pre-processing of E-mail content. ...
Step 3: Feature Extraction and Selection. ...
Step 4: Implementation. ...
Step 5: Performance Analysis.

Code

```
# necessary libraries
import openai
import pandas as pd
import numpy as np

# replace "YOUR API KEY" with your generated API key
openai.api_key = "YOUR API KEY"

# while loading the csv, we ignore any encoding errors and skip any bad line
df = pd.read_csv('spam.csv', encoding_errors='ignore', on_bad_lines='skip')
print(df.shape)
# we have 3 columns with NULL values, to remove that we use the below line
df = df.dropna(axis=1)
# we are taking only the first 60 rows for developing the model
df = df.iloc[:60]
# rename the columns v1 and v2 to Output and Text respectively
df.rename(columns = {'v1':'OUTPUT', 'v2': 'TEXT'}, inplace = True)
print(df.shape)
df.head()
```

```
(5572, 5)
(60, 2)
```

	OUTPUT	TEXT
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...

Email Spam Classification Dataset

```
# function to generate vector for a string
def get_embedding(text, model="text-embedding-ada-002"):
return openai.Embedding.create(input = , model=model)['data'][0]['embedding']

# applying the above funtion to generate vectors for all 60 text pieces
df["embedding"] = df.TEXT.apply(get_embedding).apply(np.array) # convert string to array
df.head()
```

	OUTPUT	TEXT	embedding
0	ham	Go until jurong point, crazy.. Available only ...	[-0.011956056579947472, -0.026185495778918266,...
1	ham	Ok lar... Joking wif u oni...	[-0.0024703105445951223, -0.0312176700681448, ...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	[-0.008984447456896305, 0.0006775223882868886,...
3	ham	U dun say so early hor... U c already then say...	[0.010833987966179848, -0.011291580274701118, ...
4	ham	Nah I don't think he goes to usf, he lives aro...	[0.012792329303920269, -1.7723063137964346e-05...

```
class_dict = {'spam': 1, 'ham': 0}
df['class_embeddings'] = df.OUTPUT.map(class_dict)
df.head()
```

	OUTPUT	TEXT	embedding	class_embeddings
0	ham	Go until jurong point, crazy.. Available only ...	[-0.011956056579947472, -0.026185495778918266,...	0
1	ham	Ok lar... Joking wif u oni...	[-0.0024703105445951223, -0.0312176700681448, ...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	[-0.008984447456896305, 0.0006775223882868886,...	1
3	ham	U dun say so early hor... U c already then say...	[0.010833987966179848, -0.011291580274701118, ...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	[0.012792329303920269, -1.7723063137964346e-05...	0

```
# split data into train and test
X = np.array(df.embedding)
y = np.array(df.class_embeddings)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# train random forest classifier
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train.tolist(), y_train)
preds = clf.predict(X_test.tolist())

# generate a classification report involving f1-score, recall, precision and accuracy
report = classification_report(y_test, preds)
print(report)
```

Output:

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90         9
           1       1.00      0.33      0.50         3

    accuracy                           0.83        12
   macro avg       0.91      0.67      0.70        12
weighted avg       0.86      0.83      0.80        12
```

```
print("accuracy: ", np.round(accuracy_score(y_test, preds)*100,2), "%")
```

Output:

```
accuracy:  83.33 %
```

```
confusion_matrix(y_test, preds)
```

```
array([[9, 0],
       [2, 1]])
```

```
#pip install -q openai

import openai

# replace "YOUR API KEY" with your generated API key
openai.api_key = "YOUR API KEY"

def spam_classification(message):
response = openai.Completion.create(
model="text-davinci-003",
prompt=f"Classify the following message as spam or not spam:\n\n{message}\n\nAnswer:",
temperature=0,
max_tokens=64,
top_p=1.0,
frequency_penalty=0.0,
presence_penalty=0.0
)
return response['choices'][0]['text'].strip()
```

Example1

```
out = spam_classification("""Congratulations! You've Won a $1000 gift card from walmart.
Go to https://bit.ly to claim your reward.""")
print(out)
```

Output

Spam

Example 2

```
out = spam_classification("Hey Alex, just wanted to let you know tomorrow is an off. Thank you")
print(out)
```

Output

Not Spam