**NAANMUDHALVAN-IBM SKILL**

**ARTIFICIAL INTELLIGENCE**

**GROUP PROJECT**

**Project Title\* Build a smarter AI-powered spam classifier**

**Phase IV. Submission**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Group Members  Name | NM ID | Email ID |
| 1 | AROORAN B | au820321106007 | Arooranbalachandar16@gamil.com |
| 2 | SAKTHI GANESH P | au820321106032 |  |
| 3 | YOGESHWARAN | au820321106041 | Yogeshmaddy692@gmail.com |
| 4 | MANO BALAN B | au820321106025 | Balathandayutam26@gmail.com |

**Build a smarter AI spam classifier using python**

**Features Engineering based:**

The majority of people in today’s society own a mobile phone, and they all frequently get communications (SMS/email) on their phones. But the key point is that some of the messages you get may be spam, with very few being genuine or important interactions. You may be tricked into providing your personal information, such as your password, account number, or Social Security number, by scammers that send out phony text messages. They may be able to access your bank, email, and other accounts if they obtain this information. To filter out these messages, a spam filtering system is used that marks a message spam on the basis of its contents or sender.

In this article, we will be seeing how to develop a spam classification system and also evaluate our model using various metrics. In this article, we will be majorly focusing on OpenAI API. There are 2 ways to

We will be using the Email Spam Classification Dataset dataset which has mainly 2 columns and 5572 rows with spam and non-spam messages. You can download the dataset from [here](https://drive.google.com/file/d/100bqkZNSEaZd1ZYrPl10rrOsEVimnDuZ/view?usp=sharing).

## Steps to implement Spam Classification using **OpenAI**

### Now there are two approaches that we will be covering in this article:

### 1. Using [**Embeddings**](https://www.geeksforgeeks.org/word-embeddings-in-nlp/) API developed by [**OpenAI**](https://www.geeksforgeeks.org/open-ai-gpt-3/)

#### **Step 1:**Install all the necessary salaries

!pip install -q openai

#### **Step 2:**Import all the required libraries

## **Python3**

|  |
| --- |
| # necessary libraries  **import** openai  **import** pandas as pd  **import** numpy as np  # libraries to develop and evaluate a machine learning model  **from** sklearn.ensemble **import** RandomForestClassifier  **from** sklearn.model\_selection **import** train\_test\_split  **from** sklearn.metrics **import** classification\_report, accuracy\_score  **from** sklearn.ensemble **import** RandomForestClassifier  **from** sklearn.model\_selection **import** train\_test\_split  **from** sklearn.metrics **import** classification\_report, accuracy\_score  **from** sklearn.metrics **import** confusion\_matrix |

#### **Step 3:**Assign your API key to the **OpenAI** environment

## **Python3**

|  |
| --- |
| # replace "YOUR API KEY" with your generated API key  openai.api\_key **=** "YOUR API KEY" |

#### **Step 4:**Read the CSV file and clean the dataset

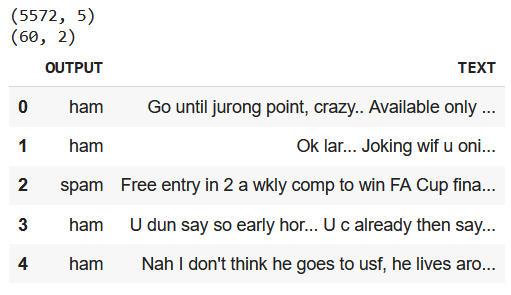
Our dataset has 3 unnamed columns with NULL values,

**Note:**Open AI’s public API does not process more than 60 requests per minute. so we will drop them and we are taking only 60 records here only.

## **Python3**

|  |
| --- |
| # 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() |

**Output:**



*Email Spam Classification Dataset*

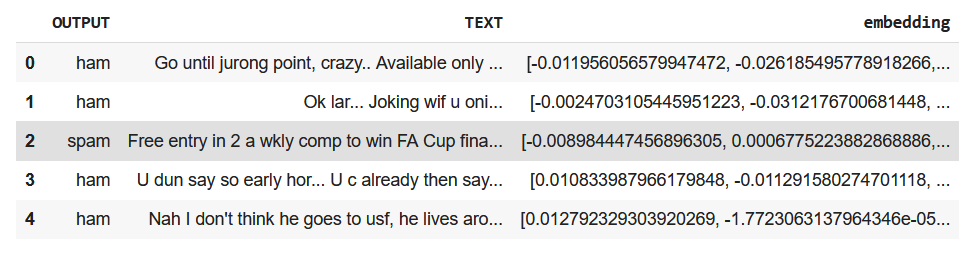
#### Step 5: Define a function to use Open AI’s **Embedding** API

We use the Open AI’s Embedding function to generate embedding vectors and use them for classification. Our API uses the “text-embedding-ada-002” model which belongs to the second generation of embedding models developed by OpenAI. The embedding generated by this model are of length 1536.

## **Python3**

|  |
| --- |
| # 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:**



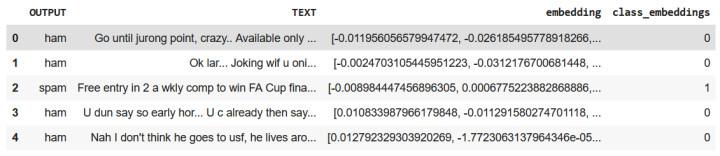
*Email Spam Classification Dataset*

#### Step 6: Custom Label the classes of the output variable to 1 and 0, where 1 means “spam” and 0 means “not spam”.

## **Python3**

|  |
| --- |
| class\_dict **=** {'spam': 1, 'ham': 0}  df['class\_embeddings'] **=** df.OUTPUT.map(class\_dict)  df.head() |

**Output:**



*Spam Classification dataFrame after feature engineerin*

#### Step 7: Develop a Classification model.

We will be splitting the dataset into a training set and validation dataset using train\_test\_split and training a Random Forest Classification model.

## **Python3**

|  |
| --- |
| # 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

#### Step 8: Calculate the accuracy of the model

## **Python3**

|  |
| --- |
| print("accuracy: ", np.round(accuracy\_score(y\_test, preds)**\***100,2), "%") |

**Output:**

accuracy: 83.33 %

#### Step 9: Print the [**confusion matrix**](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/) for our classification model

## **Python3**

|  |
| --- |
| confusion\_matrix(y\_test, preds) |

**Output:**

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

### 2. Using text completion API developed by OpenAI

#### Step 1: Install the **Openai** library in the **Python** environment

!pip install -q openai

#### Step 2: Import the following libraries

## **Python3**

|  |
| --- |
| **import** openai |

#### Step 3: Assign your API key to the **Openai**the environment

## **Python3**

|  |
| --- |
| # replace "YOUR API KEY" with your generated API key  openai.api\_key **=** "YOUR API KEY" |

#### Step 4: Define a function using the text completion API of **Openai**

## **Python3**

|  |
| --- |
| **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() |

#### Step 5: Try out the function with some examples

**Example 1:**

## **Python3**

|  |
| --- |
| out **=** spam\_classification("""Congratulations! You've Won a $1000 gift card from walmart.                            Go to [https://bit.ly](https://bit.ly/) to claim your reward.""")  print(out) |

**Output:**

Spam

**Example 2:**

## **Python3**

|  |
| --- |
| out **=** spam\_classification("Hey Alex, just wanted to let you know tomorrow is an off. Thank you")  print(out) |

**Output:**

Not spam