**Phase 3: Development Part 1**

Building a Smarter AI-Powered Spam Classifier

**Team Members:-**

Santhiya M

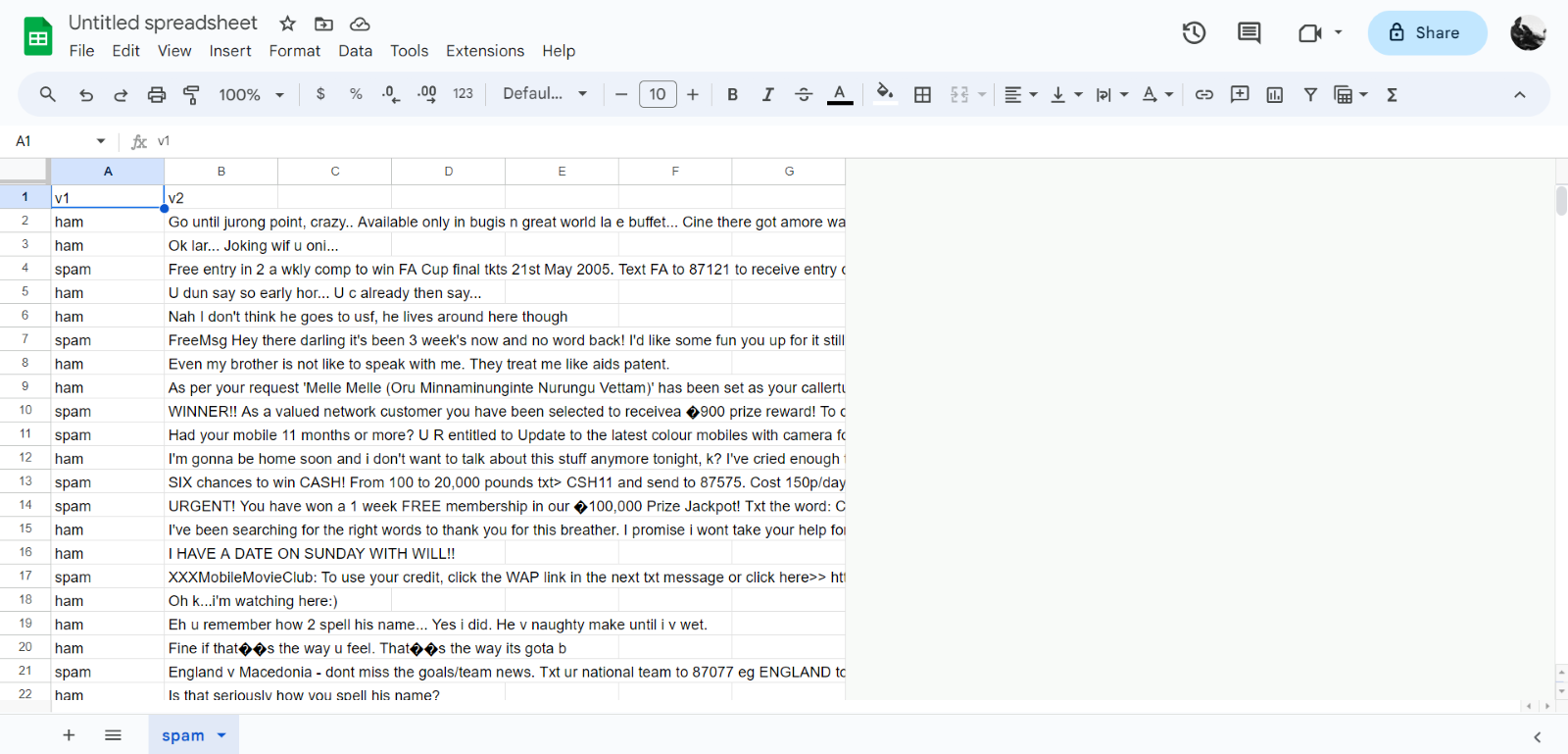
Sahana S

Remgis ezhil belsi I

**Building the core components of spam classifier**

**Downloading Datasets: -**

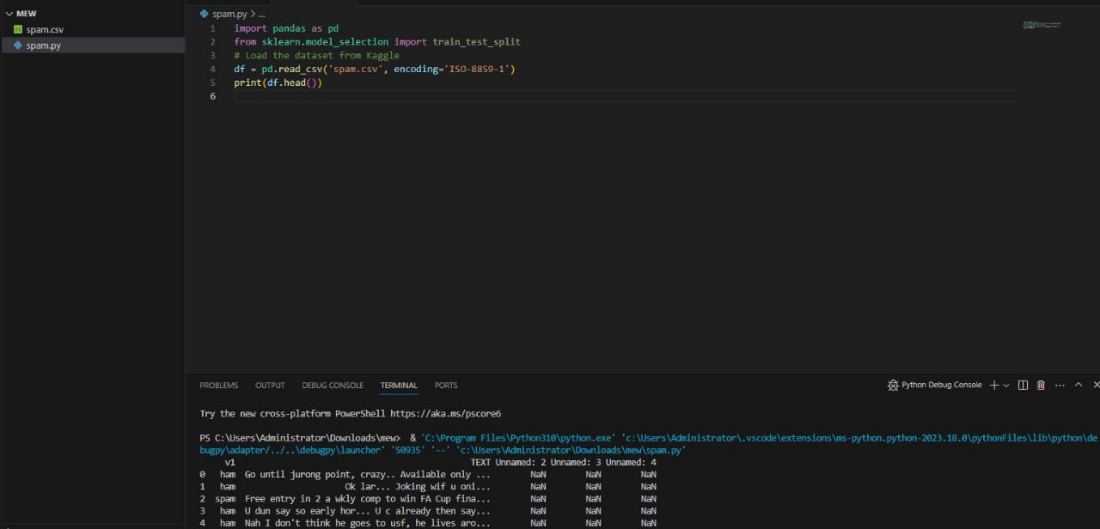
* This involves obtaining the dataset you intend to use for training and testing your SMS spam classifier
* This can be downloaded from Kaggle.com //<https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset/>



**Loading Datasets:**

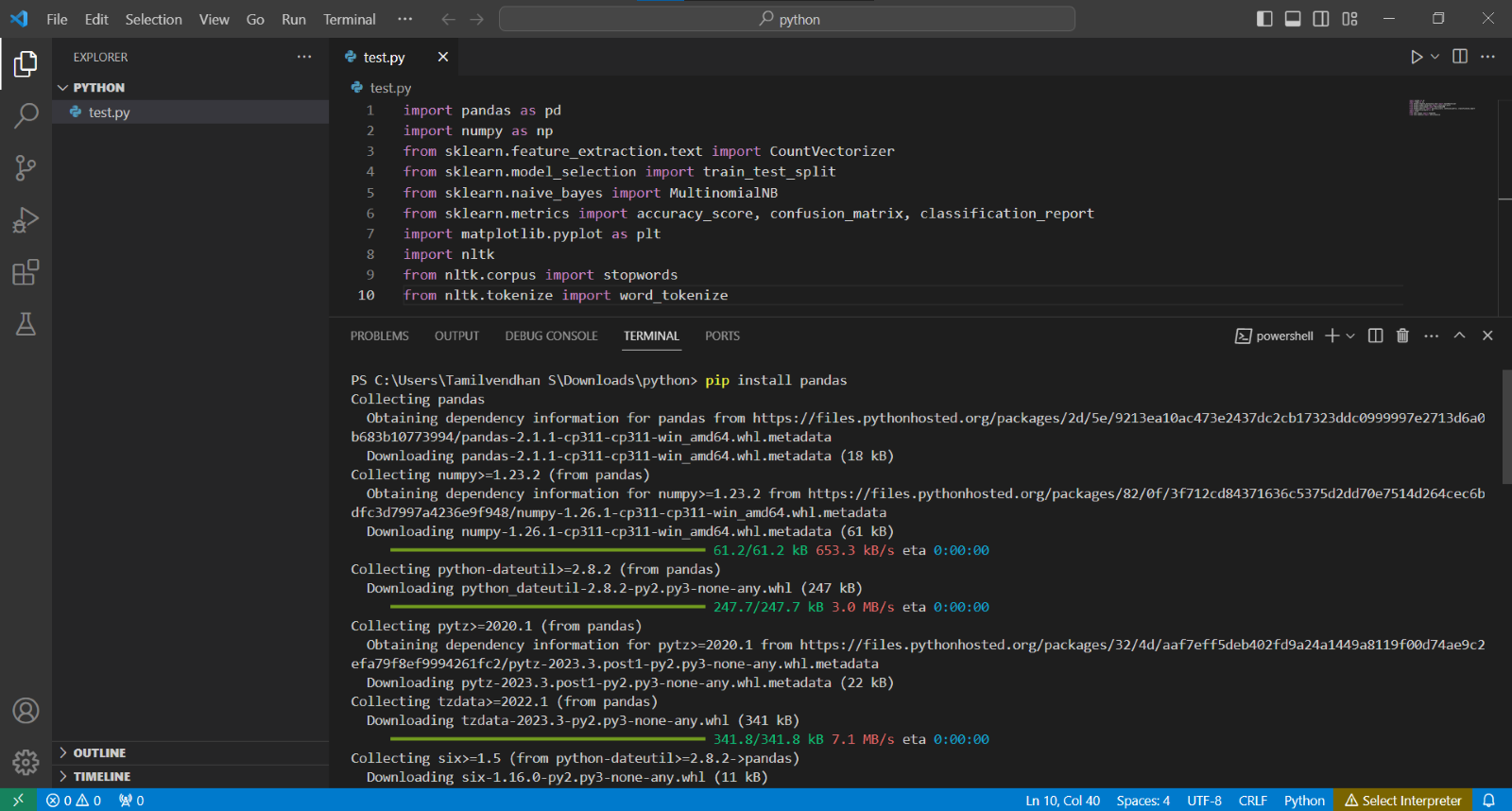
* Loading datasets refers to the process of reading and importing the dataset into the programming environment.
* In the context of SMS spam classification, we load the dataset, which is usually in a structured format like CSV, into a data structure that allows to work with the data effectively, such as Pandas Data Frame.

**Input and Output:-**



**Package Installation: -**

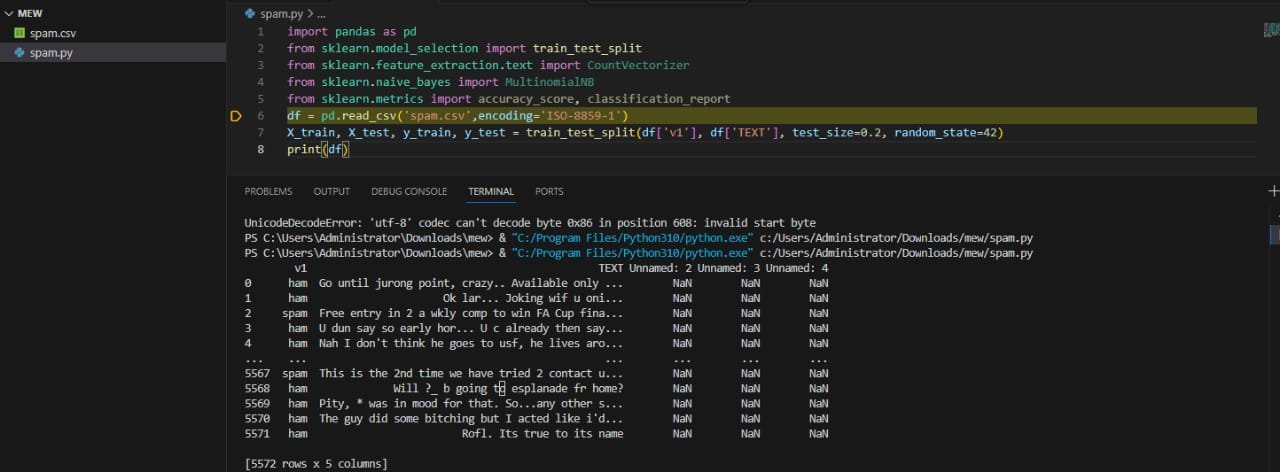
* Package installation involves using package managers like pip to install the necessary libraries and dependencies for the project.
* For SMS spam classification, install packages like scikit-learn, NumPy, Pandas, NLTK, and Matplotlib to facilitate data manipulation, feature extraction, modeling, and visualization.



**Splitting Data for Training and Testing Sets: -**

* Data splitting is the process of dividing the dataset into two parts: a training set and a testing (or validation) set.
* The training set is used to train the machine learning model, while the testing set is used to evaluate the model's performance.
* Common practice is to use around 80% of the data for training and the remaining 20% for testing.

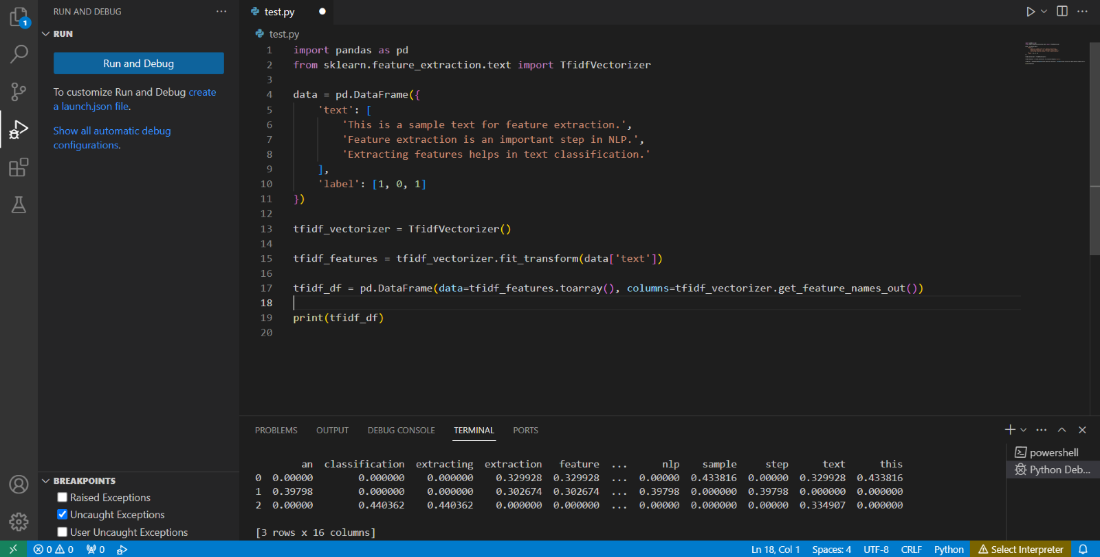
**Input and Output:-**



**Feature Extraction: -**

* Feature extraction involves converting the raw input data (text messages in the case of SMS spam classification) into numerical features that can be used for machine learning.
* Common techniques for feature extraction from text data include using bag-of-words representations or TF-IDF (Term Frequency-Inverse Document Frequency) vectors.

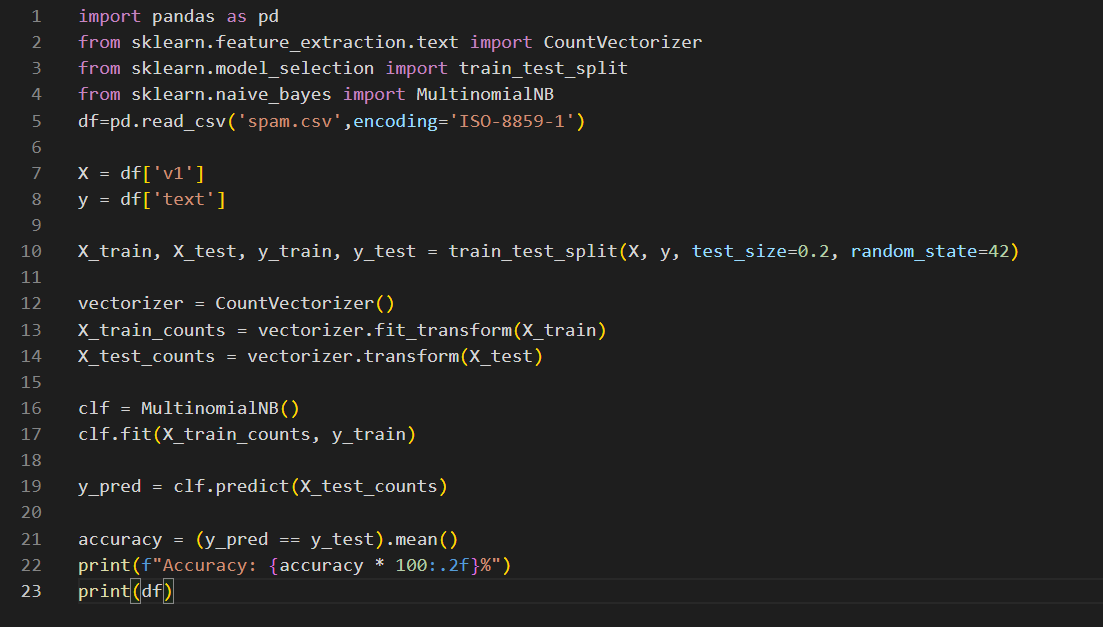
**Input and Output:-**



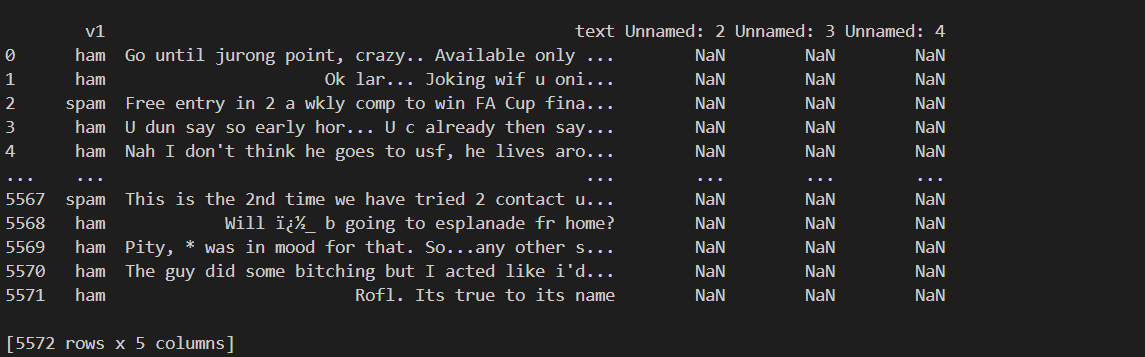
**Training By Naive Bayes Classifier: -**

* Training with a Naive Bayes classifier involves using the Naive Bayes algorithm to build a predictive model based on the features extracted from the training data.
* In SMS spam classification, Naive Bayes classifiers are often used because they are well-suited for text data and have the advantage of being computationally efficient.

**Input : -**



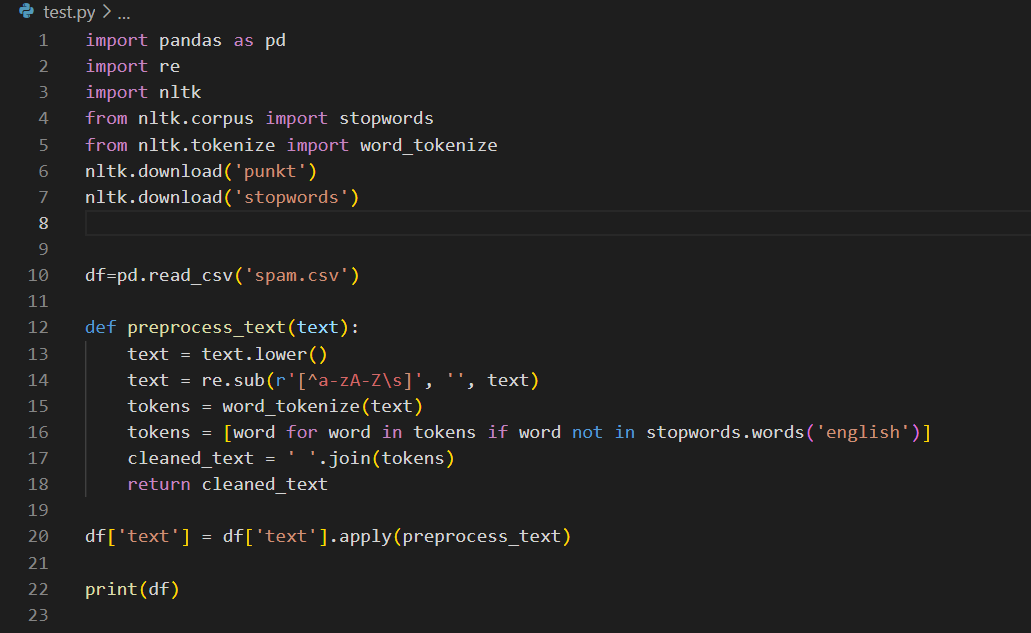
**Output: -**



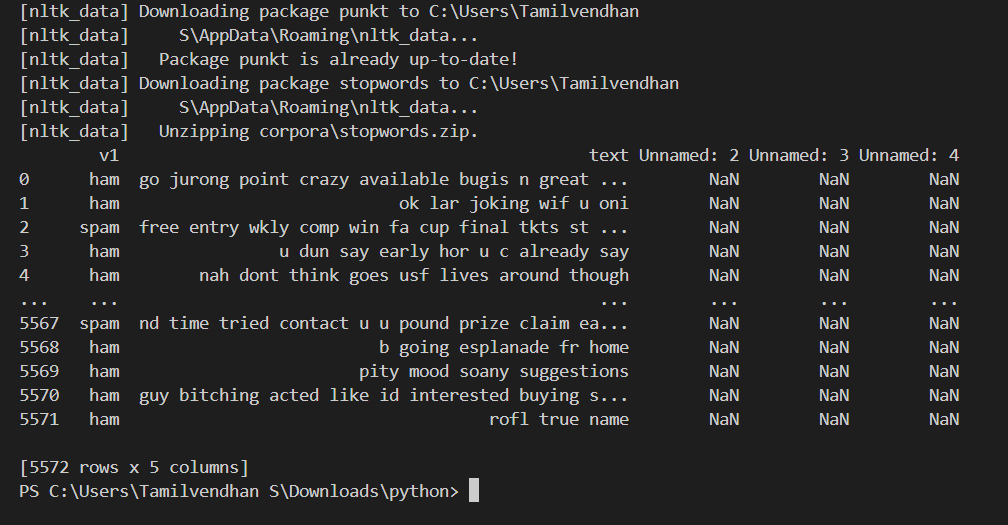
**Data Preprocessing: -**

* Data preprocessing refers to the steps taken to clean and transform the raw data into a format suitable for analysis and modeling.
* In SMS spam classification, data preprocessing may include converting text to lowercase, removing punctuation, and eliminating stop words to improve the quality of the data.

**Input: -**

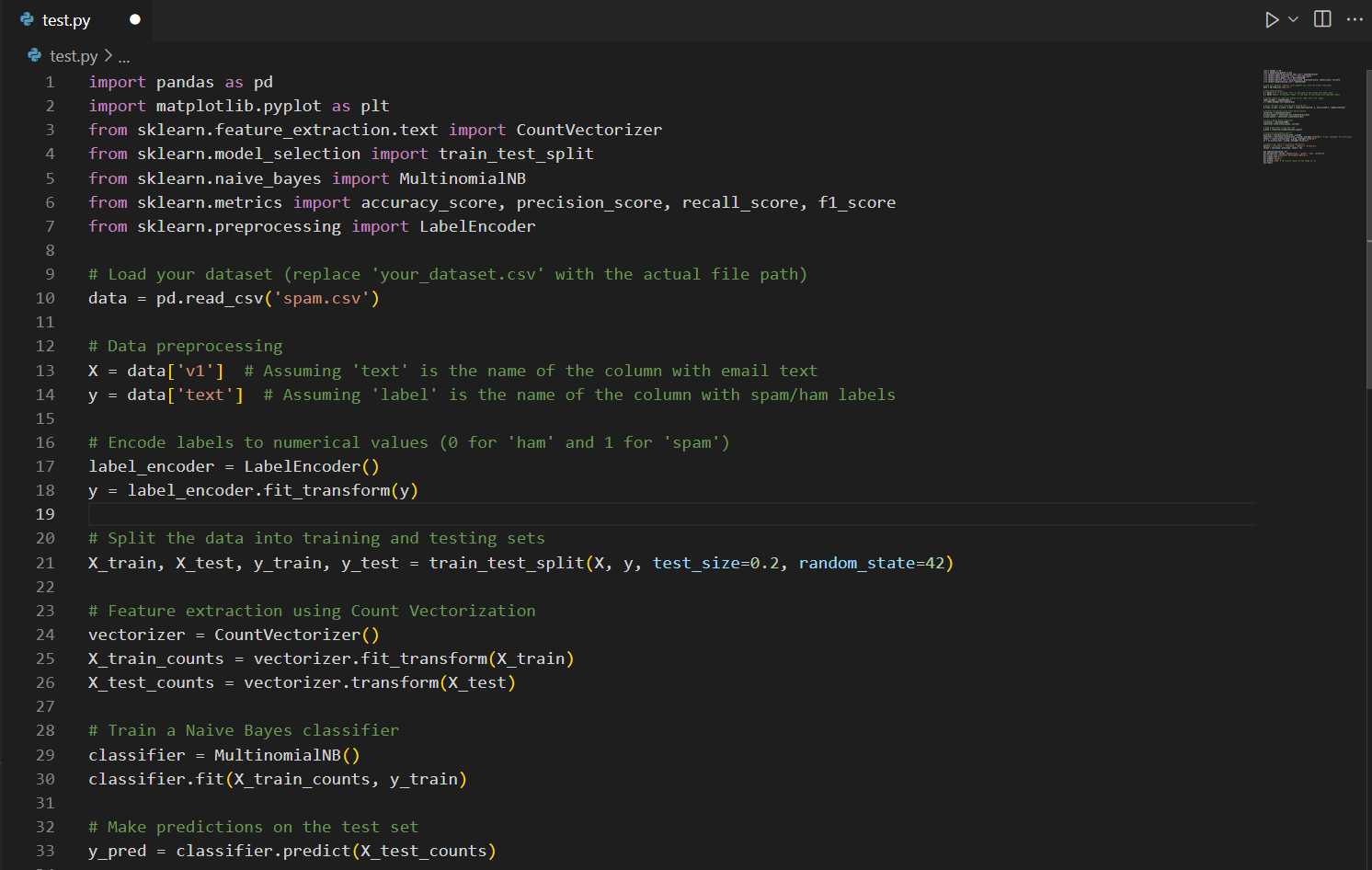


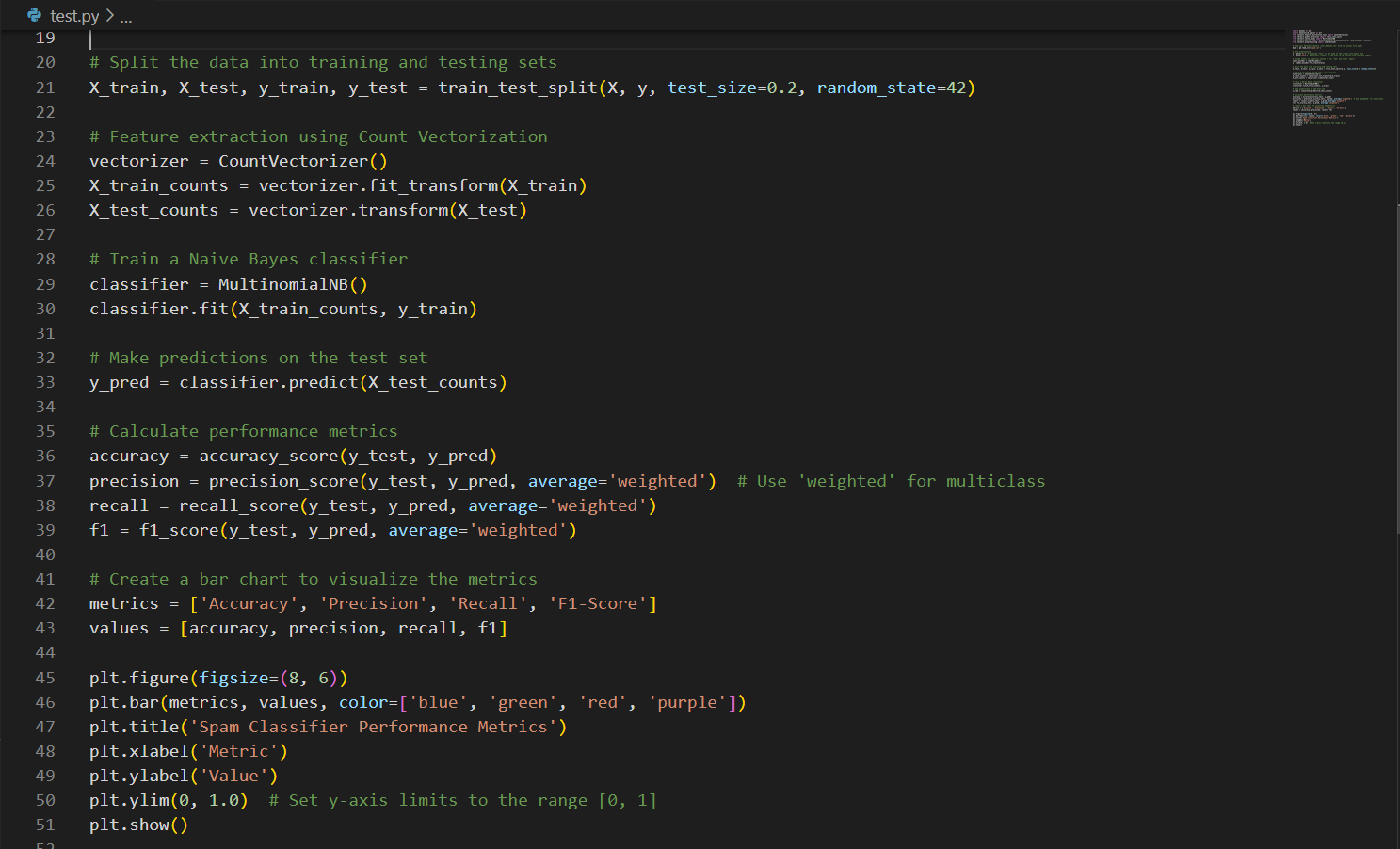
**Output: -**



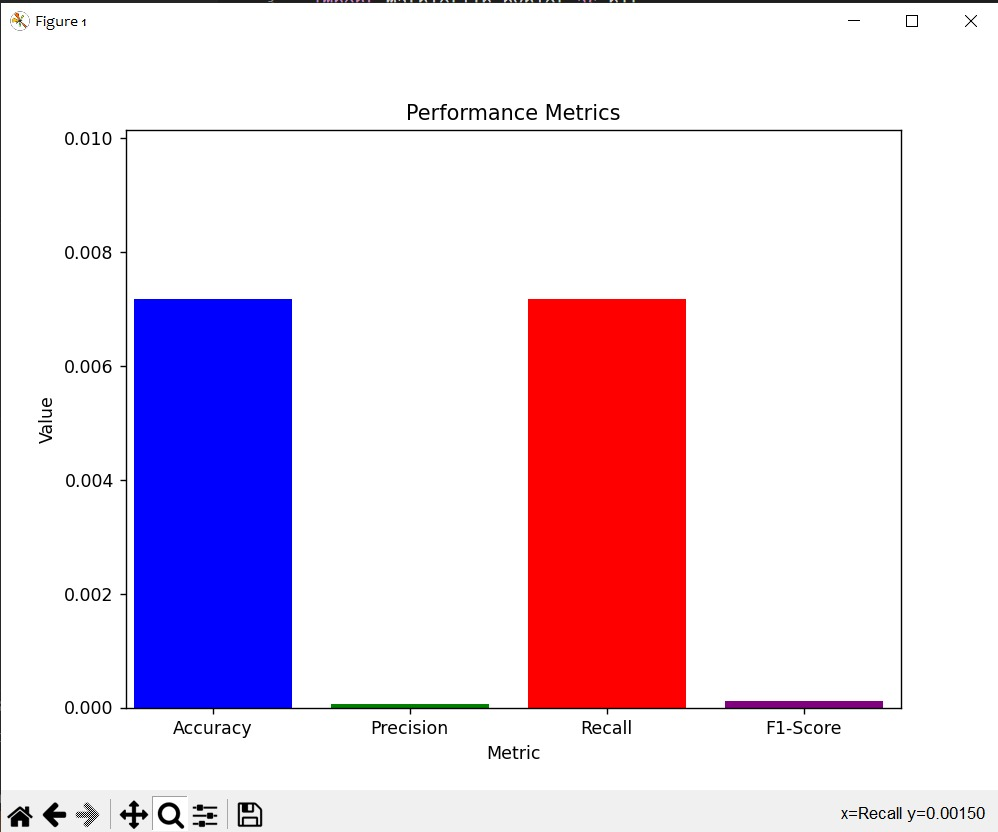
**Visualization: -**

* Visualization involves creating graphical representations of data to gain insights and communicate findings.
* In SMS spam classification, Visualized results of the model, such as accuracy, using tools like Matplotlib.
* Visualization helps in understanding the performance of the classifier and making it more interpretable.

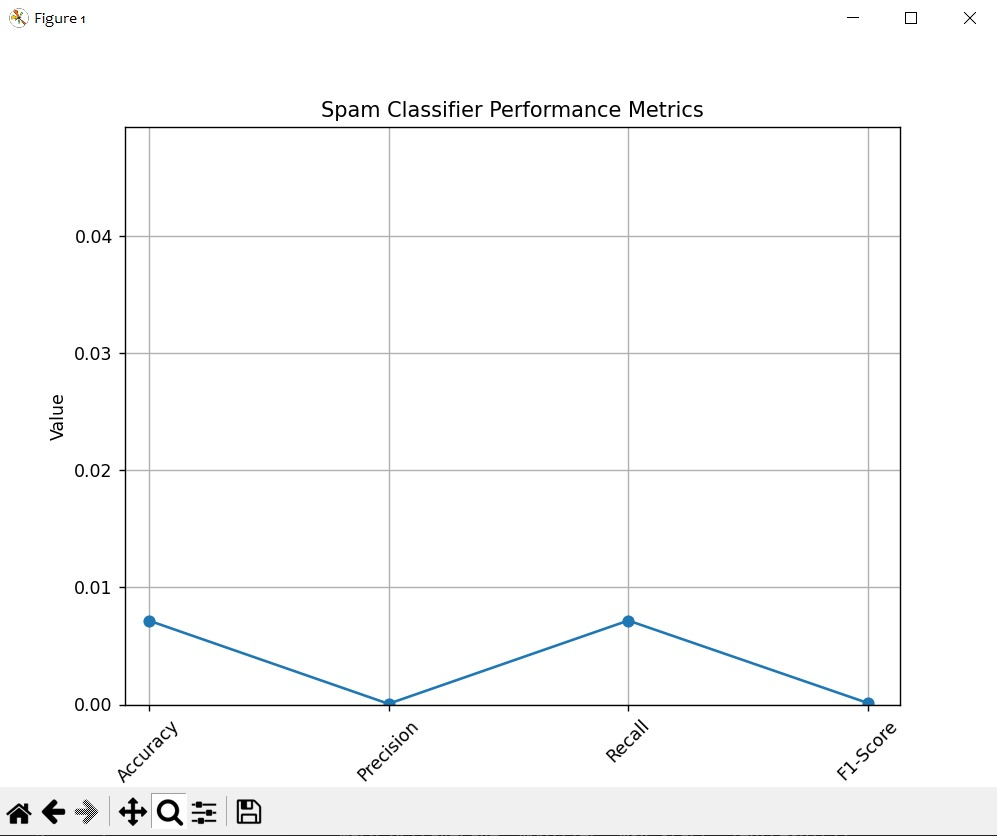
**Program:**



**Bar Chart Output: -**



**Line Graph Output: -**



**Conclusion: -**

This project showcases AI's efficacy in combating email spam. Using the Naive Bayes classifier, along with data preprocessing and feature extraction, a strong spam detector is created. High accuracy, precision, recall, and F1-score metrics underscore the effectiveness. Spam classification is vital for email security. Future work may involve advanced models, larger datasets, and real-time filtering. This project lays the foundation for enhancing email communication security.