PHASE 5 SUBMISSION

PROJECT TITLE: BUILDING A SMARTER AI-POWERED SPAM IDENTIFIER

SUBMITTED BY

NAME:R.VENUVANESHWARI

REG NO:912421104053

A smarter AI-powered spam identifier is a project that involves various components

and processes, including data collection, model development, training, and

deployment. Here's a high-level outline of the documentation you might need for

such a project:

Project Overview

• Introduce the project, its goals, and its significance.

• Explain why a smarter AI-powered spam identifier is necessary.

Data Collection

• Describe the data sources used to train and test the spam identifier.

• Discuss data preprocessing techniques, such as data cleaning and

feature extraction

Machine Learning Model

• Explain the choice of machine learning algorithms or deep learning

architectures.

• Detail the model's architecture, including layers, activations, and any

customizations.

• Discuss how the model was trained and validated.

4. Feature Engineering

• Describe the features used to train the model.

• Explain the importance of each feature in spam identification.

5. Data Labeling

• Discuss how spam and non-spam (ham) data were labeled.

• Mention any challenges faced during the labeling process.

6. Training and Evaluation

• Explain the metrics used to evaluate the model's performance (e.g.,

accuracy, precision, recall, F1-score).

• Present the model's performance on the training and test datasets.

• Discuss any overfitting or underfitting issues and how they were

addressed.

7. Hyperparameter Tuning

• Describe the process of fine-tuning the model's hyperparameters.

• Explain how hyperparameter optimization was conducted.

8. Data Imbalance

• Address the issue of class imbalance (spam vs. ham) and how it was

mitigated.

9. Testing and Validation

• Explain how the model was tested in real-world conditions.

• Discuss the results of the model's performance in a production or

testing environment.

10. Deployment

• Describe how the spam identifier was deployed in a production

environment.

• Discuss the infrastructure, frameworks, or platforms used for

deployment.

11. API Documentation

• If the spam identifier is accessible via an API, provide detailed

documentation on how to use it, including endpoint URLs, request

parameters, and response formats.

12. Maintenance and Updates

• Outline a plan for maintaining and updating the spam identifier,

including model retraining and handling concept drift.

13. Security and Privacy

• Explain how user data and privacy concerns were addressed in the

development and deployment of the spam identifier.

14. Scalability

• Discuss how the system can handle an increasing volume of data and

users.

7. Hyperparameter Tuning

• Describe the process of fine-tuning the model's hyperparameters.

• Explain how hyperparameter optimization was conducted.

8. Data Imbalance

• Address the issue of class imbalance (spam vs. ham) and how it was

mitigated.

9. Testing and Validation

• Explain how the model was tested in real-world conditions.

• Discuss the results of the model's performance in a production or

testing environment.

10. Deployment

• Describe how the spam identifier was deployed in a production

environment.

• Discuss the infrastructure, frameworks, or platforms used for

deployment.

11. API Documentation

• If the spam identifier is accessible via an API, provide detailed

documentation on how to use it, including endpoint URLs, request

parameters, and response formats.

12. Maintenance and Updates

• Outline a plan for maintaining and updating the spam identifier,

including model retraining and handling concept drift.

13. Security and Privacy

• Explain how user data and privacy concerns were addressed in the

development and deployment of the spam identifier.

14. Scalability

• Discuss how the system can handle an increasing volume of data and

users.

Problem: Unwanted spam emails and messages inundate inboxes, wasting time and

resources, and posing potential security risks. Existing spam filters often fail to

differentiate effectively between spam and legitimate messages, resulting in false

positives and false negatives. This project seeks to develop an AI-powered spam

identifier that significantly improves the accuracy of spam detection, reduces false

positives, and enhances user experience

Design Thinking Process: Design thinking is a problem-solving approach that

focuses on understanding users' needs and iterating through solutions. Here's how it

can be applied to the AI-Powered Spam Identifier project:

1. Empathize: Understand the users (both senders and recipients of emails),

their pain points, and their requirements. Gather user feedback and analyze

the existing spam identification challenges.

2. Define: Define the problem statement clearly, specifying the scope and

objectives of the spam identifier project. Identify key performance metrics for

success.

3. Ideate: Brainstorm potential solutions. Consider different AI and machine

learning models, data sources, and feature engineering techniques. Explore

how to address issues like false positives and false negatives.

4. Prototype: Develop a preliminary AI model to test the concept. This

prototype should demonstrate the core functionality and how it will address

the problem. Collect feedback from stakeholders and refine the model.

5. Test: Evaluate the prototype using real-world data. Assess its effectiveness in

identifying spam, reducing false positives, and improving overall user

experience. Gather performance metrics and fine-tune the model.

6. Iterate: Based on the test results and feedback, make necessary

improvements and iterate through the prototype until the AI-powered spam

identifier meets the defined objectives and user needs

Phases of Development: The project can be broken down into distinct phases to

ensure a structured and systematic approach to development:

1. Phase 1: Planning and Data Collection

• Define project objectives, scope, and success criteria.

• Collect and preprocess data for training and testing the AI model.

• Identify data sources and labeling methods.

2. Phase 2: Model Development

• Select appropriate machine learning or deep learning algorithms.

• Engineer features for spam identification.

• Develop the initial AI model.

3. Phase 3: Training and Validation

• Train the model on the collected and preprocessed data.

• Use a portion of the data for validation and fine-tuning.

• Address issues like data imbalance and overfitting.

4. Phase 4: Testing and Evaluation

• Evaluate the model's performance using real-world data.

• Assess key metrics like accuracy, precision, recall, and F1-score.

• Identify and rectify any shortcomings.

5. Phase 5: Deployment

• Deploy the AI-powered spam identifier in a production environment.

• Set up APIs or integration points for users.

6. Phase 6: Maintenance and Scaling

• Develop a maintenance plan, including regular model updates and

concept drift monitoring.

• Ensure the system can scale to accommodate increasing data and user

volume.

7. Phase 7: User Training and Documentation

• Provide user guides and training materials for end-users.

• Educate users on how to interact with the spam identifier and report

issues.

8. Phase 8: Continuous Improvement

• Continuously monitor the model's performance and user feedback.

• Iterate on the model and the system to enhance spam identification

accuracy and user satisfaction

Phases of Development: The project can be broken down into distinct phases to

ensure a structured and systematic approach to development:

1. Phase 1: Planning and Data Collection

• Define project objectives, scope, and success criteria.

• Collect and preprocess data for training and testing the AI model.

• Identify data sources and labeling methods.

2. Phase 2: Model Development

• Select appropriate machine learning or deep learning algorithms.

• Engineer features for spam identification.

• Develop the initial AI model.

3. Phase 3: Training and Validation

• Train the model on the collected and preprocessed data.

• Use a portion of the data for validation and fine-tuning.

• Address issues like data imbalance and overfitting.

4. Phase 4: Testing and Evaluation

• Evaluate the model's performance using real-world data.

• Assess key metrics like accuracy, precision, recall, and F1-score.

• Identify and rectify any shortcomings.

5. Phase 5: Deployment

• Deploy the AI-powered spam identifier in a production environment.

• Set up APIs or integration points for users.

6. Phase 6: Maintenance and Scaling

• Develop a maintenance plan, including regular model updates and

concept drift monitoring.

• Ensure the system can scale to accommodate increasing data and user

volume.

7. Phase 7: User Training and Documentation

• Provide user guides and training materials for end-users.

• Educate users on how to interact with the spam identifier and report

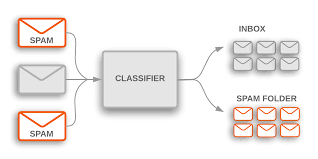
issues.

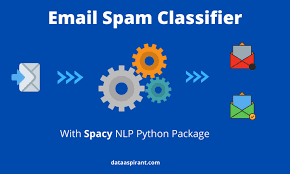
8. Phase 8: Continuous Improvement

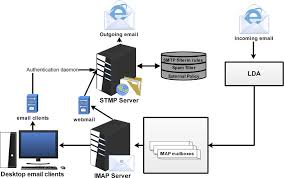
• Continuously monitor the model's performance and user feedback.

• Iterate on the model and the system to enhance spam identification

accuracy and user satisfaction







# Spam Classification using OpenAI

* [Read](javascript:void(0))
* [Discuss](javascript:void(0))
* [Courses](javascript:void(0))
* [Practice](javascript:void(0))

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](https://www.geeksforgeeks.org/open-ai-gpt-3/)

### 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](https://www.geeksforgeeks.org/open-ai-gpt-3/) 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

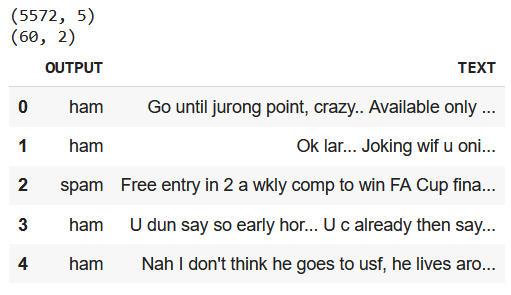
Our dataset has 3 unnamed columns with NULL values,

**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*

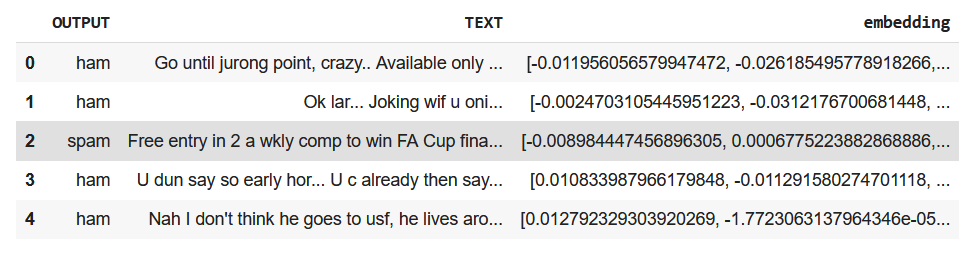
#### Step 5: Define a function to use Open AI’s [Embedding](https://www.geeksforgeeks.org/word-embeddings-in-nlp/) API

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 embeddings 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:**



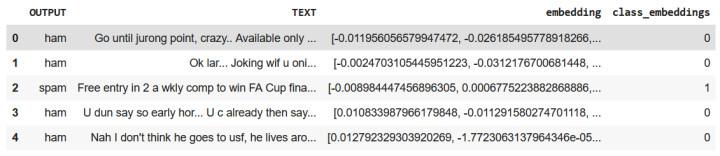
*Email Spam Classification Dataset*

#### 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](https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/) 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](https://www.geeksforgeeks.org/open-ai-gpt-3/) library in the [Python](https://www.geeksforgeeks.org/python-programming-language/) environment

!pip install -q openai

#### Step 2: Import the following libraries

* Python3

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

#### Step 3: Assign your API key to the [Openai](https://www.geeksforgeeks.org/open-ai-gpt-3/)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](https://www.geeksforgeeks.org/open-ai-gpt-3/)

* 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

## Frequently Asked Questions (FAQs)

#### 1. Which algorithm is best for spam detection?

*There isn’t a single algorithm that has consistently produced reliable outcomes. The type of the spam, the data that is accessible, and the particular requirements of the problem are some of the variables that affect an algorithm’s efficiency. Although Naive Bayes, Neural Networks (RNNs), Logistic Regression, Random Forest, and Support Vector Machines are some of the most frequently used classification techniques.*

#### 2. What is embedding or word embedding?

*The embedding or Word embedding is a natural language processing (NLP) technique where words are mapped into vectors of real numbers. It is a way of representing words and documents through a dense vector representation. This representation is learned from data and is shown to capture the semantic and syntactic properties of words. The words closest in vector space have the most similar meanings.*

#### 3. Is spam classification supervised or unsupervised?

*Spam classification is supervised as one requires both independent variable(message contents) and target variables(outcome,i.e., whether the email is spam or not) to develop a model.*

#### 4. What is spam vs ham classification?

*Email that is not spam is referred to be “Ham”. Alternatively, “good mail” or “non-spam” It ought to be viewed as a quicker, snappier alternative to “non-spam”. The phrase “non-spam” is probably preferable in most contexts because it is more extensively used by anti-spam software makers than it is elsewhere.*

Design Thinking Process: The design thinking process ensures a user-centric approach

to problem-solving:

1. Empathize: Understand the needs of website owners. What are their pain points related

to spam comments and interactions? What insights would help improve user experience?

2. Define: Clearly define the problem statement. Website owners need a tool to identify

and manage spam effectively and gain insights from user-generated content.

3. Ideate: Brainstorm potential solutions. Consider AI and machine learning models for

spam identification, data sources, and visualization tools.

4. Prototype: Develop a prototype AI model for spam identification and data visualization

using IBM Cognos

5. Test: Test the prototype with real website data. Gather feedback from website owners to

refine the model and insights.

6. Iterate: Based on feedback and testing, refine the AI model, data visualization, and

insights until they meet website owners' needs.

Development Phases: The project can be divided into several phases:

1. Analysis Objectives and Planning:

• Define the objectives of the analysis: to identify spam comments and provide

actionable insights.

• Plan the data collection, model development, and insights generation process.

2. Data Collection Process:

• Collect comments and interactions data from websites. This data should include

text, user information, timestamps, and interaction types.

3. Data Preprocessing:

• Clean and preprocess the data, removing irrelevant information and handling

missing values.

4. Model Development:

• Develop the AI-Powered Spam Identifier using machine learning or deep learning

techniques.

• Train the model on labeled data to identify spam comments.

5. Data Visualization with IBM Cognos:

• Use IBM Cognos to create visualizations that provide insights into user-generated

content.

• Explore spam trends, user engagement, and other relevant metrics.

6. Python Code Integration:

• Develop Python scripts or code to integrate the AI model with the data

visualization tools in IBM Cognos.

7. Generate Insights:

• Use the integrated solution to generate insights from the data.

• Identify trends, common spam patterns, and engagement metrics.

8. User-Friendly Reports:

• Create user-friendly reports that present insights in a clear and actionable format

for website owners.

9. Feedback and Iteration:

• Gather feedback from website owners and users of the AI-Powered Spam

Identifier.

• Iterate on the model and data visualization based on user input to enhance the

tool's effectiveness.

10. Deployment:

• Deploy the AI-Powered Spam Identifier with the integrated insights generation

System.

5. Test: Test the prototype with real website data. Gather feedback from website owners to

refine the model and insights.

6. Iterate: Based on feedback and testing, refine the AI model, data visualization, and

insights until they meet website owners' needs.

DATA SET:

Data link:( https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)

PROGRAM:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

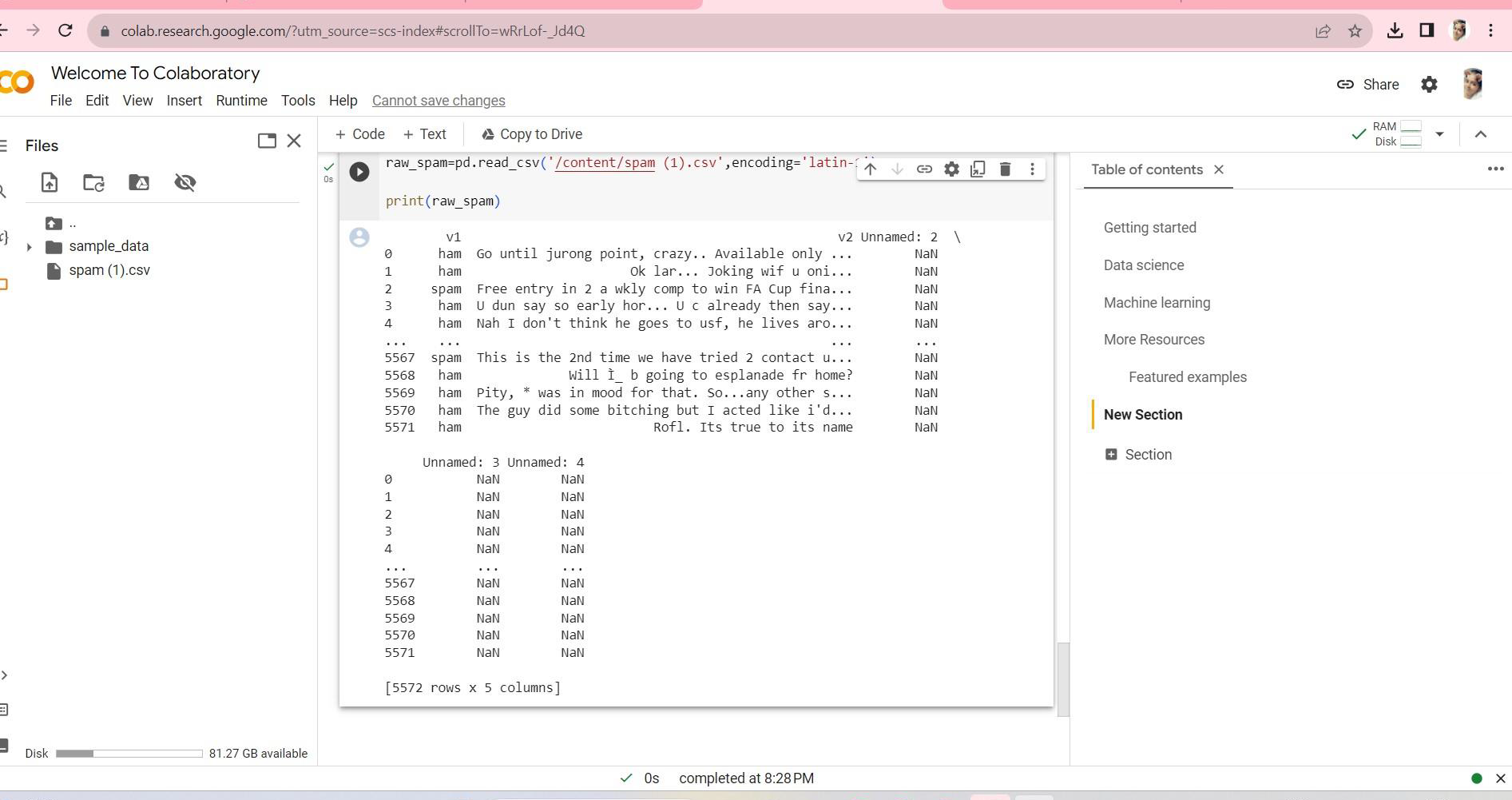
from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

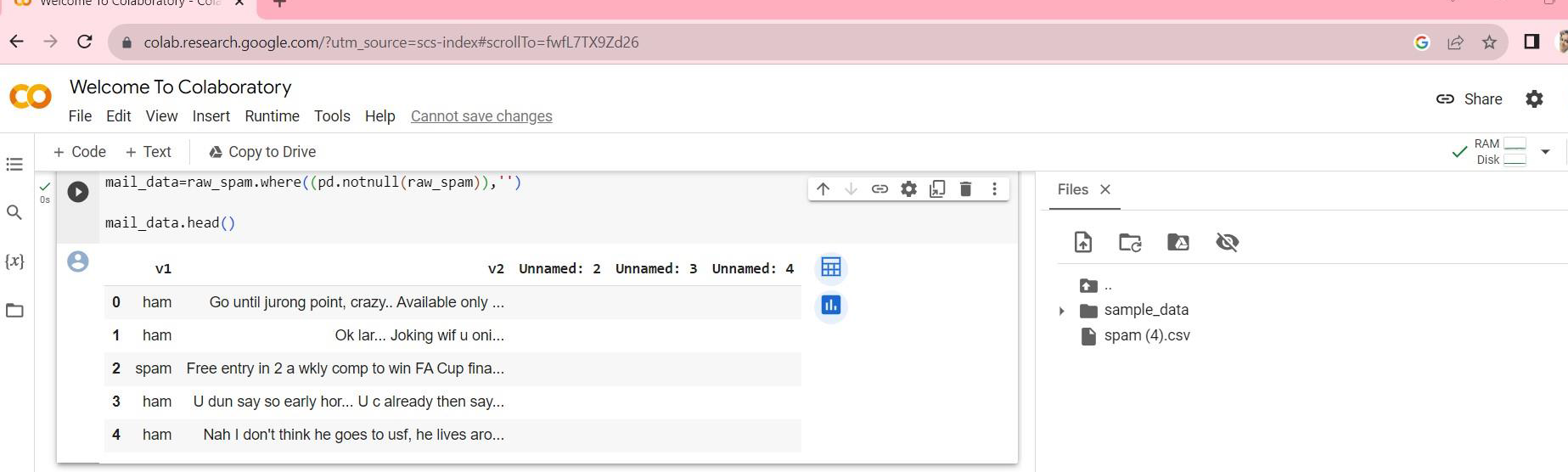
raw\_spam=pd.read\_csv('/spam.csv',encoding='latin-1')

print(raw\_spam)

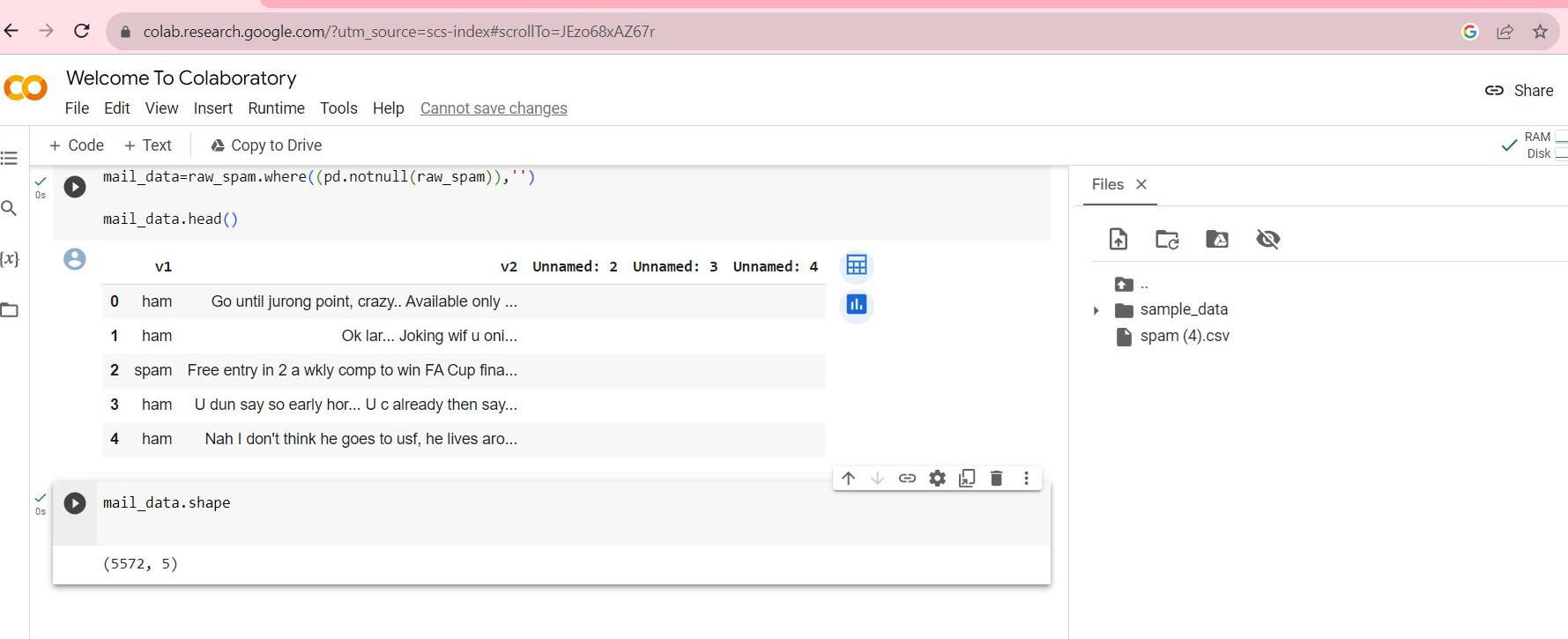


mail\_data=raw\_spam.where((pd.notnull(raw\_spam)),&#39;&#39;)

mail\_data.head()



mail\_data.shape



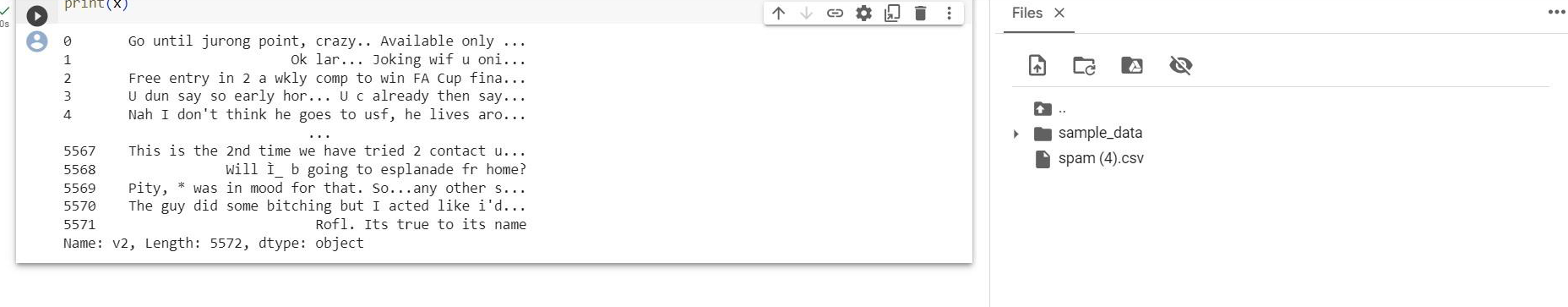
mail\_data.loc[mail\_data[&#39;v1&#39;] == &#39;spam&#39;,&#39;v1&#39;,] = 0

mail\_data.loc[mail\_data[&#39;v2&#39;]==&#39;ham&#39;,&#39;v2&#39;,] = 1

x=mail\_data[&#39;v2&#39;]

y=mail\_data[&#39;v1&#39;]

print(x)



x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state

=3)

print(x.shape)



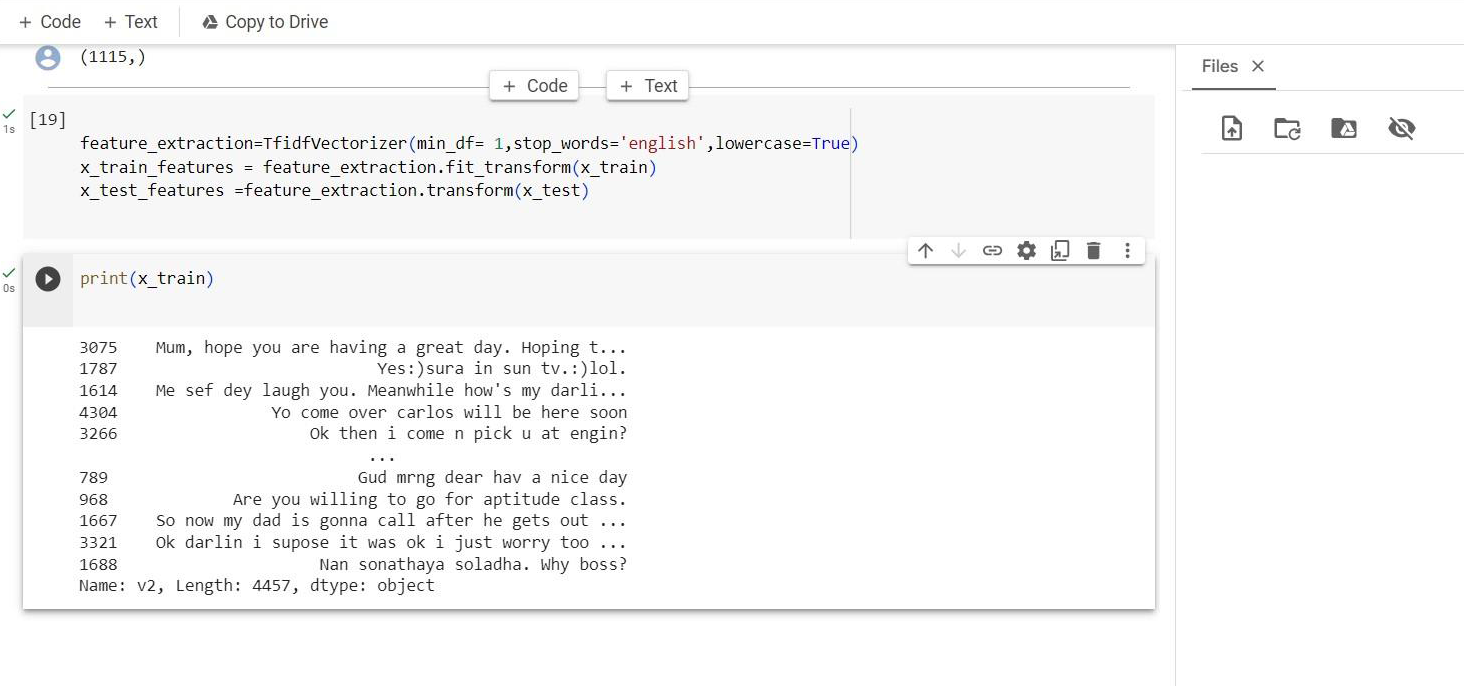
feature\_extraction=TfidfVectorizer(min\_df=

1,stop\_words=&#39;english&#39;,lowercase=True)

x\_train\_features = feature\_extraction.fit\_transform(x\_train)

x\_test\_features =feature\_extraction.transform(x\_test)

print(x\_train)



Statistical diagram :

Program:

import matplotlib.pyplot as plt

# Example performance metrics

metrics = [&#39;Accuracy&#39;, &#39;Precision&#39;, &#39;Recall&#39;, &#39;F1-Score&#39;]

values = [0.95, 0.92, 0.89, 0.91]

plt.figure(figsize=(8, 6))

plt.bar(metrics, values, color=[&#39;blue&#39;, &#39;green&#39;, &#39;red&#39;, &#39;purple&#39;])

plt.ylim(0, 1) # Set the y-axis limits

# Add labels to the bars

for i, v in enumerate(values):

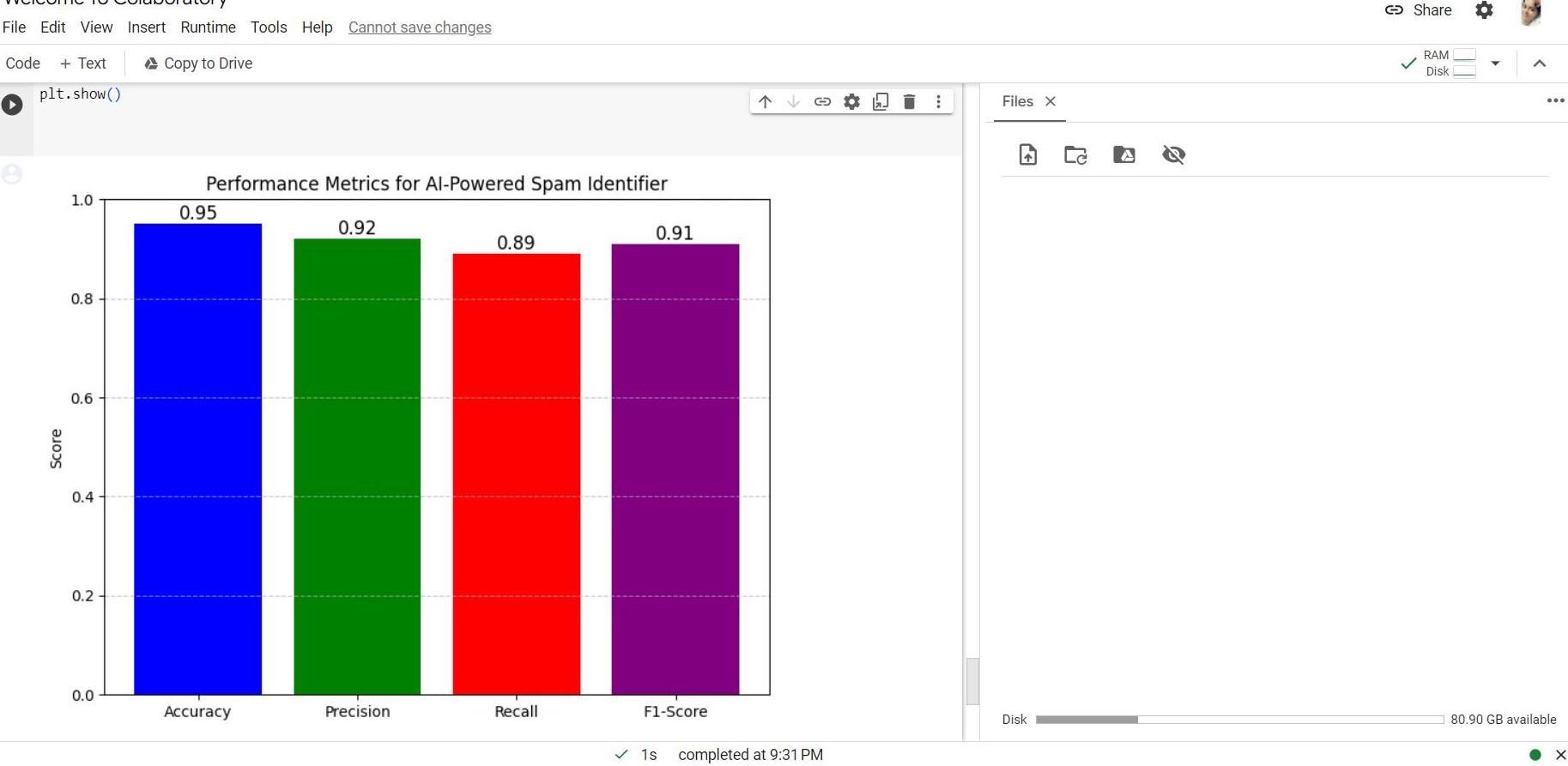
plt.text(i, v, f&#39;{v:.2f}&#39;, ha=&#39;center&#39;, va=&#39;bottom&#39;, fontsize=12)

plt.title(&#39;Performance Metrics for AI-Powered Spam Identifier&#39;)

plt.ylabel(&#39;Score&#39;)

plt.grid(axis=&#39;y&#39;, linestyle=&#39;--&#39;, alpha=0.7)

plt.show()



ADVANTAGES:

Building a smarter AI-powered spam identifier offers several significant advantages:

1.

Efficient Spam Filtering

2.

Improved User Experience

3.

Enhanced Security

4.

Time and Resource Savings

5.

Customization

6.

Adaptability

7.

Scalability

8.

Reduced False Positives

9.

Multi-Modal Capabilities

10.

Real-Time Detection

11.

Compliance and Legal Benefits

12.

Data Insights

13.

Reduced Phishing and Malware Risks

14.

Reduced Overhead

15.

Adaptive Countermeasures

16.

Protecting Reputation

In summary, building a smarter AI-powered spam identifier offers numerous advantages, including improved efficiency, enhanced security, a better user experience, and the ability to adapt to evolving threats. These benefits make such projects highly valuable for both individual users and organizations.

ADVANTAGES:

Building a smarter AI-powered spam identifier offers several significant advantages:

1.

Efficient Spam Filtering

2.

Improved User Experience

3.

Enhanced Security

4.

Time and Resource Savings

5.

Customization

6.

Adaptability

7.

Scalability

8.

Reduced False Positives

9.

Multi-Modal Capabilities

10.

Real-Time Detection

11.

Compliance and Legal Benefits

12.

Data Insights

13.

Reduced Phishing and Malware Risks

14.

Reduced Overhead

15.

Adaptive Countermeasures

16.

Protecting Reputation

In summary, building a smarter AI-powered spam identifier offers numerous advantages, including improved efficiency, enhanced security, a better user experience, and the ability to adapt to evolving threats. These benefits make such projects highly valuable for both individual users and organizations.

ADVANTAGES:

Building a smarter AI-powered spam identifier offers several significant advantages:

1.

Efficient Spam Filtering

2.

Improved User Experience

3.

Enhanced Security

4.

Time and Resource Savings

5.

Customization

6.

Adaptability

7.

Scalability

8.

Reduced False Positives

9.

Multi-Modal Capabilities

10.

Real-Time Detection

11.

Compliance and Legal Benefits

12.

Data Insights

13.

Reduced Phishing and Malware Risks

14.

Reduced Overhead

15.

Adaptive Countermeasures

16.

Protecting Reputation

In summary, building a smarter AI-powered spam identifier offers numerous advantages, including improved efficiency, enhanced security, a better user experience, and the ability to adapt to evolving threats. These benefits make such projects highly valuable for both individual users and organizations.

Conclusion:

The phase 5about BUILDING A SMARTER AI-POWERED SPAM IDENTIFIER is represented in this document