SPAM MAIL DETECTION USING NAÏVE BAYES CLASSIFIER

A PROJECT REPORT

Submitted by

ABISHEAK A (202109004)
KARPAGA GANESH B (202109027)
MANOJ S (202109032)

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK COLLEGE OF ENGINEERING,
SIVAKASI-626123

(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATASCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **“AbisheakA(95172021090), karpaga Ganesh (95172021090), Manoj S(9517202109032)”** for the mini project titled **“SPAM MAIL DETECTION USING NAÏVE BAYES CLASSIFIER”** in 19AD451 – Data Analytics Laboratory and 19AD452- Artificial Intelligence Laboratory during the fourth semester January 2023 – May 2023 under my supervision.

SIGNATURE	**SIGNATURE**
Dr.P.Thendral, M.E.,Ph.D,	**Dr.J.AngelaJennifaSujana, M.E.,Ph.D,**
Assistant Professor(SG),	**Associate Professor(SG) & Head,**
Artificial Intelligence and Data Science,	Artificial Intelligence and Data Science ,
MepcoSchlenk Engineering College,	MepcoSchlenk Engineering College,
Sivakasi- 626 005,Virudhunagar.	Sivakasi- 626 005,Virudhunagar.

Submitted for the project viva-voice examination to be held on ………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATASCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of "**AbisheakA(95172021090), karpaga Ganesh (95172021090), Manoj S(9517202109032)**" for the mini project titled **"SPAM MAIL DETECTION USING NAÏVE BAYES CLASSIFIER"** in 19AD451 – Data Analytics Laboratory and 19AD452- Artificial Intelligence Laboratory during the fourth semester January 2023 – May 2023 under my supervision.

SIGNATURE

Dr.A.Shenbagarajan, M.E.,Ph.D
Assistant Professor(SG),
Artificial Intelligence and Data Science,
MepcoSchlenk Engineering College,
Sivakasi- 626 005,Virudhunagar.

SIGNATURE

Dr.J.AngelaJennifaSujana, M.E.,Ph.D,
Associate Professor(SG) & Head,
Artificial Intelligence and Data Science ,
MepcoSchlenk Engineering College,
Sivakasi- 626 005,Virudhunagar.

Submitted for the project viva-voice examination to be held on ……….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost we **praise and thank "The Almighty",** the lord of all creations, who by his abundant grace has sustained us and helped us to work on this project successfully.

We really find unique pleasure and immense gratitude in thanking our respected management members**,** who is the backbone of our college.

A deep bouquet of thanks to respected Principal **Dr.S.Arivazhagan M.E.,Ph.D.,** for having provided the facilities required for our mini project.

We sincerely thank our Head of the Department **Dr. J. Angela JennifaSujana M.E.,Ph.D.,** Associate Professor(SG) & Head, Department of Artificial Intelligence and Data Science, for her guidance and support throughout the mini project .

We also thank our guide **Dr.A.Shenbagarajan.,M.E.,Ph.D.,** Assistant Professor(SG), **Mrs.P.Thendral., M.E(Ph.D)**, Assistant Professor(SG) Department of Artificial Intelligence and Data Sciencefor their valuable guidance and it is great privilege to express our gratitude to them.

We extremely thank our project coordinator **Dr.A.Shenbagarajan.,M.E.,Ph.D.,** Assistant Professor(SG),**Mrs.P.Thendral, M.E(Ph.D),** Assistant Professor(SG)Department of Artificial Intelligence and Data Science, who inspired us and supported us throughout the mini project.

We extend our heartfelt thanks and profound gratitude to all the faculty members of Artificial Intelligence and Data Science department for their kind help during our mini project work.

We also thank our parents and our friends who had been providing us with constant support during the course of the mini project work.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Spam mail detection is a critical task in ensuring the security and efficiency of email communication systems. This project aims to develop a spam mail detection system using the Naive Bayes classifier. The Naive Bayes algorithm is a popular and efficient probabilistic model that leverages the principles of conditional probability to classify emails as spam or non-spam.

The project begins by creating a dataset comprising a collection of labeled emails, including both spam and non-spam examples. The emails are preprocessed to extract relevant features such as the frequency of specific words, presence of certain patterns, and structural characteristics. These features serve as inputs to the Naive Bayes classifier.

The Naive Bayes classifier is trained on the preprocessed dataset to learn the underlying probability distribution of features associated with spam and non-spam emails. During the training process, the classifier calculates the conditional probabilities of each feature given the class labels. This information is then utilized to classify new, unseen emails as either spam or non-spam.

To evaluate the effectiveness of the spam mail detection system, various performance metrics such as accuracy, precision, recall, and F1 score are computed. The trained classifier is tested on a separate evaluation dataset to measure its ability to correctly classify emails. The results of the evaluation provide insights into the system's performance and can guide further improvements.

The project also explores techniques to enhance the accuracy of spam mail detection. Feature selection methods and parameter tuning are investigated to optimize the performance of the Naive Bayes classifier. Additionally, the project examines the impact of different preprocessing techniques, such as stemming and stop-word removal, on the overall performance of the system.

The experimental results demonstrate the efficacy of Naive Bayes classifier in detecting spam mails. The system achieves high accuracy and exhibits robustness against different types of spam emails. The project contributes to field of email security by providing a practical and efficient approach for identifying and filtering out spam mails, thereby improving the overall user experience and reducing the risks associated with malicious email content.

CHAPTER 1

INTRODUCTION

Spam mail detection is a crucial task for ensuring the security and efficiency of email communication systems. This project aims to develop a spam mail detection system using the Naive Bayes classifier, a popular and efficient probabilistic model that leverages conditional probability principles to classify emails as either spam or non-spam.

The project commences with the creation of a dataset consisting of labeled emails, encompassing both spam and non-spam examples. These emails undergo preprocessing to extract pertinent features, such as word frequency, pattern presence, and structural characteristics. These extracted features are utilized as inputs for the Naive Bayes classifier. Subsequently, the Naive Bayes classifier is trained on the preprocessed dataset to learn the underlying probability distribution of features associated with spam and non-spam emails. During the training process, the classifier calculates the conditional probabilities of each feature given the class labels. This information is then used to classify new, unseen emails as either spam or non-spam.

To evaluate the effectiveness of the spam mail detection system, various performance metrics, including accuracy, precision, recall, and F1 score, are computed. The trained classifier is tested on a separate evaluation dataset to assess its ability to correctly classify emails. The evaluation results provide insights into the system's performance and can guide further improvements.

The project also explores techniques to enhance the accuracy of spam mail detection. It investigates feature selection methods and parameter tuning to optimize the performance of the Naive Bayes classifier. Additionally, the impact of different preprocessing techniques, such as stemming and stop-word removal, on the overall system performance is examined.

The experimental results demonstrate the effectiveness of the Naive Bayes classifier in detecting spam emails. The system achieves high accuracy and exhibits robustness against various types of spam. The project contributes to the field of email security by providing a practical and efficient approach to identify and filter out spam mails, thereby improving the user experience and reducing the risks associated with malicious email content.

.

1.1 SCOPE OF THE PROJECT

The scope of the spam detection project involves the development and evaluation of a spam mail detection system Preprocessing the emails to extract relevant features, such as word frequency, pattern presence, and structural characteristics. Techniques like tokenization, stop-word removal, stemming, and pattern matching may be employed

1.2 OBJECTIVE OF THE PROJECT

The objective of the spam mail detection project is to develop a robust and efficient system for accurately identifying and classifying spam emails. To create a system that can effectively distinguish between spam and non-spam emails.

1.3 BLOCK DIAGRAM

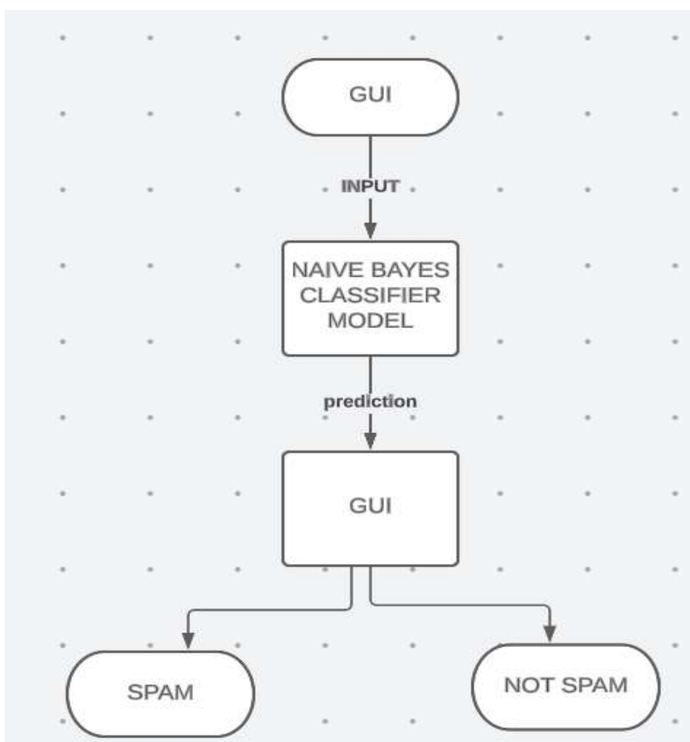

Fig.1.3.1 Block diagram

1.4 MODULE DESCRIPTION

1.4.1GRADIO

Gradio is an open-source Python package that allows you to quickly create easy-to-use, customizable UI components for your ML model, any API, or even an arbitrary Python function using a few lines of code. You can integrate the Gradio GUI directly into your Jupyter notebook or share it as a link with anyone.

1.4.2 COUNTVECTROZIER

Scikit-learn'sCountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

1.4.3 MULTINOMIALNB

MultinomialNB implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification where the data are typically represented as word vector counts.

1.4.4 MATPLOTLIB

Matplotlib is a comprehensive library for creating static, animated,and interactive visualizations in Python. Matplotlib makes easy thingseasy and hard things possible. Createpublication quality plots. Makeinteractive figuresthat can zoom, pan, update. Customizevisual styleandlayout.

1.4.5 SEABORN

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack

1.4. PANDAS

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
import seaborn as sns
import gradio as gr
```

Fig.1.4.1 Importing Module

1.5 GRAPHICAL USER INTERFACE

A graphical user interface (GUI) is an interface that is drawn on the screen for the user to interact with. User interfaces have some common components: Main window. Menu.

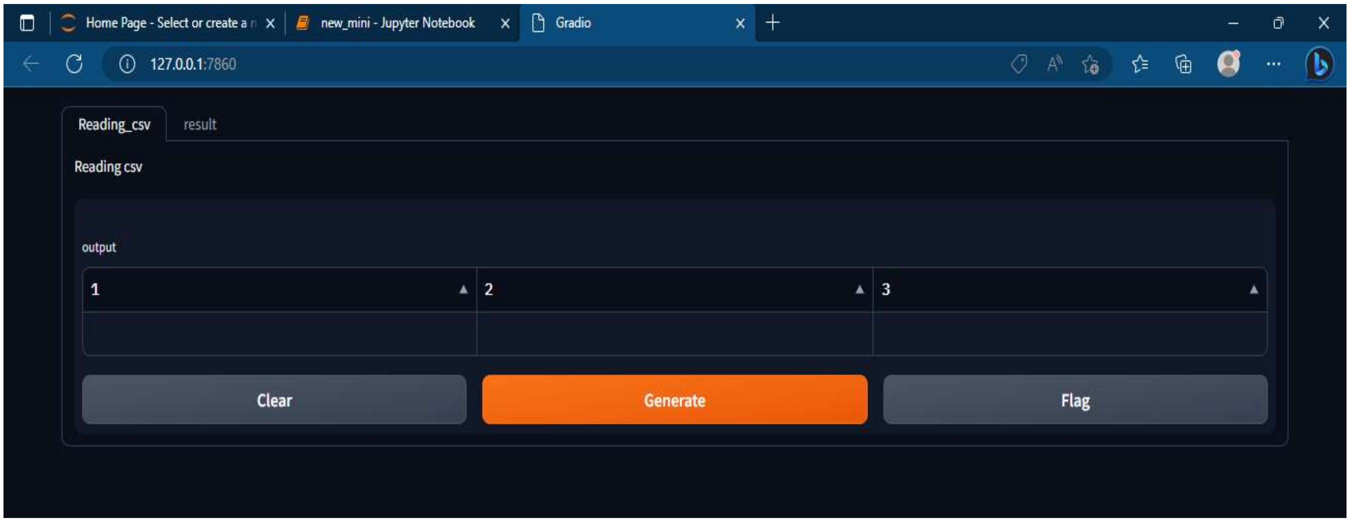

Fig.1.5.1 GUI for reading the csv file

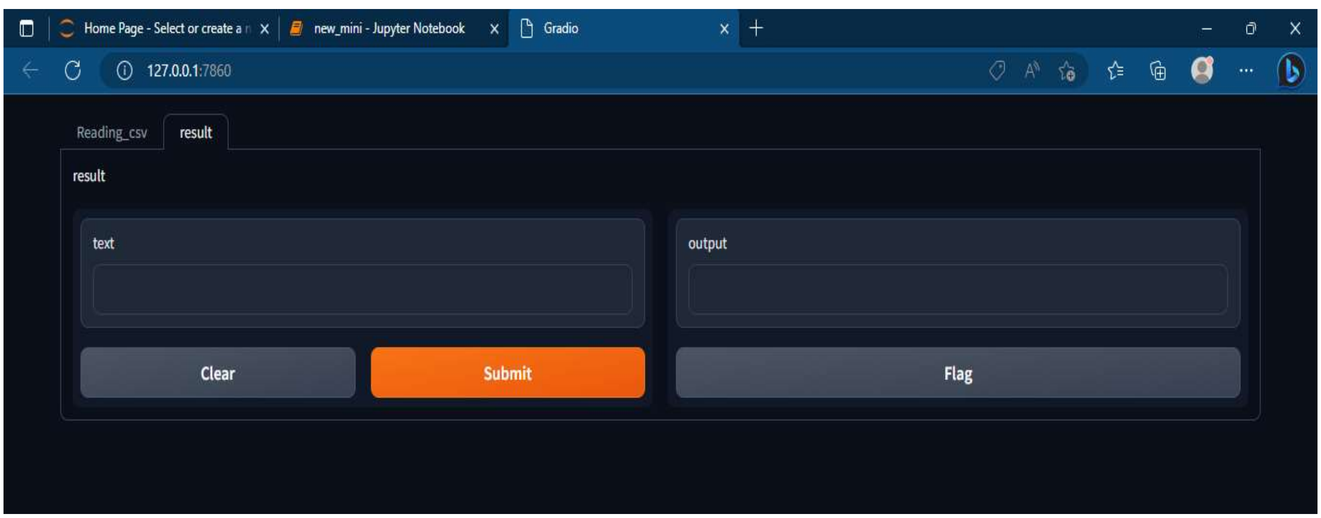

Fig.1.5.2 GUI for displaying the result

CHAPTER 2
PROPOSED MODEL

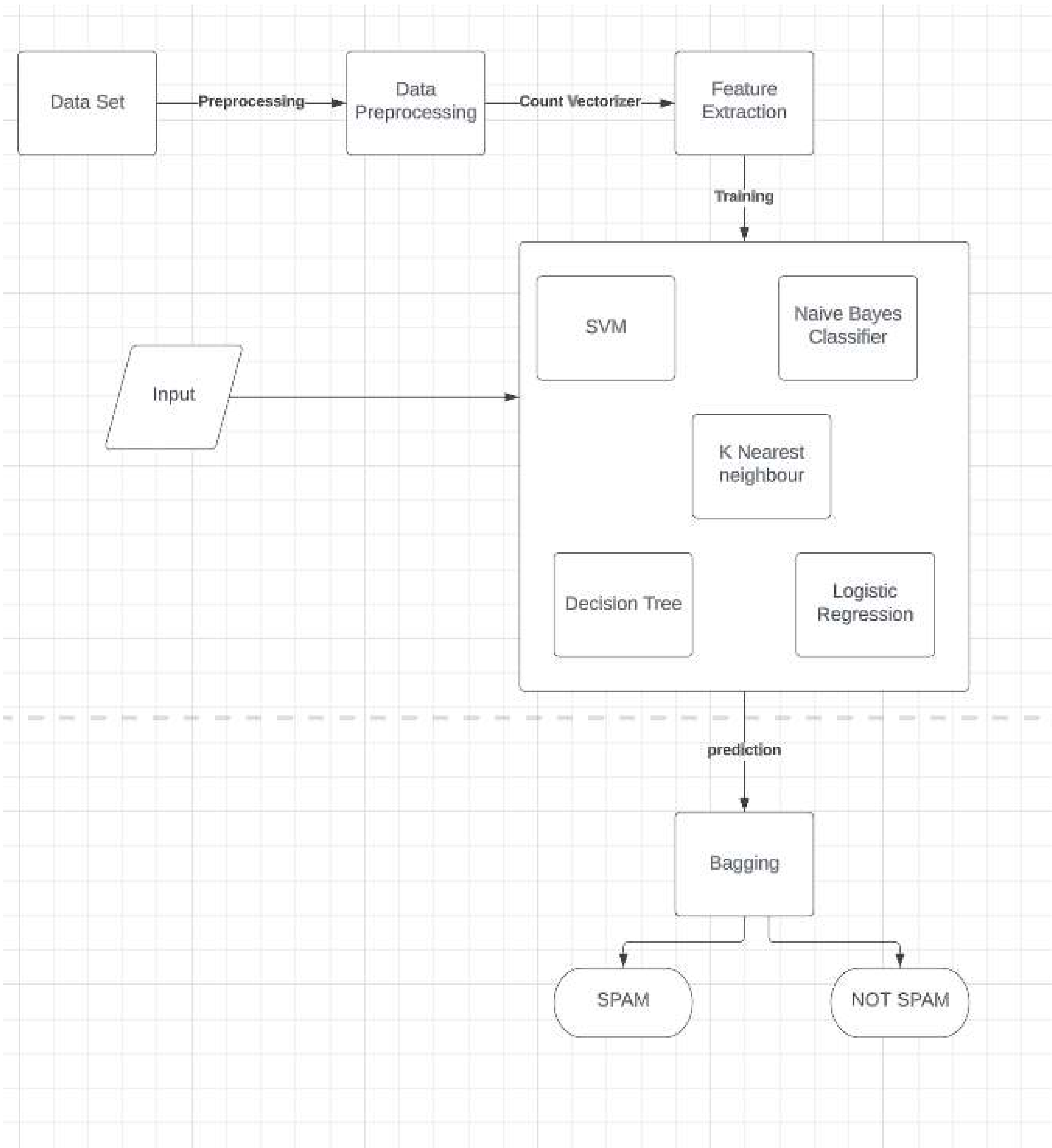

Fig. 2.1 Model Diagram

2.1 DATASET DESCRIPTION

Context

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,171 messages, tagged according being ham (legitimate) or spam.

Content

- Untitled: Notation of ham or spam
- Text: Sample mails
- Label_num: class variable (0 or 1)

Number of instances: 5171

Number of attributes: 2 + class attribute

Missing Attribute Values: No

Class Distribution : class value 1 is interpreted as “SPAM” and class value 0 is interpreted as “NOT SPAM”

2.2 DATA ANALYTICS FUNCTIONAITIES

2.2.1 Reading CSV:

The code reads a CSV file named 'spam_ham_dataset.csv' using the Pandas library's read_csv() function.

2.2.2 Exploring Data:

The code displays the shape of the dataset (data.shape), the first few rows (data.head()), the last few rows (data.tail()), and the summary information of the dataset data.info() and data.describe().

```
In [5]: data.shape
Out[5]: (5171, 3)
```

Fig.2.2.2.1 Shape

```
In [6]: data.head()
Out[6]:
```

	label	text	label_num
0	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	spam	Subject: photoshop , windows , office . cheap ...	1
4	ham	Subject: re : indian springs\r\nthis deal is t...	0

Fig.2.2.2.2 head()

In [7]: `data.tail()`

Out[7]:

	label	text	label_num
5166	ham	Subject: put the 10 on the ft\r\nthe transport...	0
5167	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...	0
5168	ham	Subject: calpine daily gas nomination\r\n>\r\n...	0
5169	ham	Subject: industrial worksheets for august 2000...	0
5170	spam	Subject: important online banking alert\r\ndea...	1

Fig.2.2.2.3 Tail()

In [8]: `data.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   label      5171 non-null   object
 1   text       5171 non-null   object
 2   label_num  5171 non-null   int64 
dtypes: int64(1), object(2)
memory usage: 121.3+ KB
```

Fig.2.2.2.4 info()

In [9]: `data.describe()`

Out[9]:

	label_num
count	5171.000000
mean	0.289886
std	0.453753
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

Fig.2.2.2.5 describe()

2.2.3 Data Visualization:

The code creates a histogram of the 'label' column using Matplotlib's hist() function. It also uses Seaborn'sheatmap() function to visualize the confusion matrices generated by different classifiers.

```
In [22]: plt.hist(data['label'])
         plt.show()
```

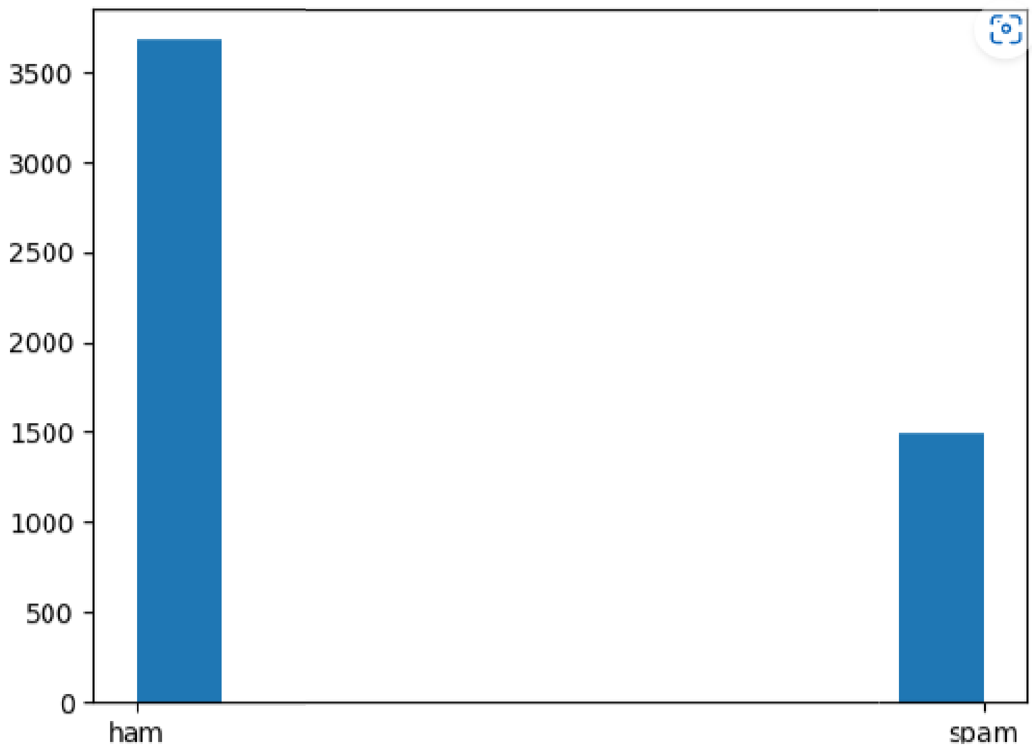

Fig.2.2.3 Histogram

CHAPTER 3

ARTIFICIAL INTELLIGENCE FUNCTIONALITIES

3.1 Text Vectorization:

The code uses Scikit-learn'sCountVectorizer() to convert text data into numerical feature vectors (spamham_countVectorizer).

```
fromsklearn.feature_extraction.textimportCountVectorizer
vectorizer=CountVectorizer()
spamham_countVectorizer=vectorizer.fit_transform(data['text'])
```

3.2 Splitting Data:

The code splits the dataset into training and testing sets using Scikit-learn'strain_test_split() function.

```
fromsklearn.model_selectionimporttrain_test_split,cross_val_score
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.2)
```

3.3 Naive Bayes Classifier:

The code trains a Multinomial Naive Bayes classifier (MultinomialNB()) on the training data and makes predictions on the testing data.

```
#Naive Bayes

NB_classifier=MultinomialNB()
NB_classifier.fit(X,y)
Y_pred=NB_classifier.predict(X_test)
cm=confusion_matrix(Y_test,Y_pred)
print(classification_report(Y_test,Y_pred))
print("The Accuracy is",accuracy_score(Y_test,Y_pred))
sns.heatmap(cm,annot=True)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       734
           1       0.99      0.97      0.98       301

    accuracy                           0.99      1035
   macro avg       0.99      0.98      0.99      1035
weighted avg       0.99      0.99      0.99      1035

The Accuracy is 0.9893719806763285
<Axes: >
```

Fig 3.3.1 Naïve Bayes Classifier Result

```
<Axes: >
```

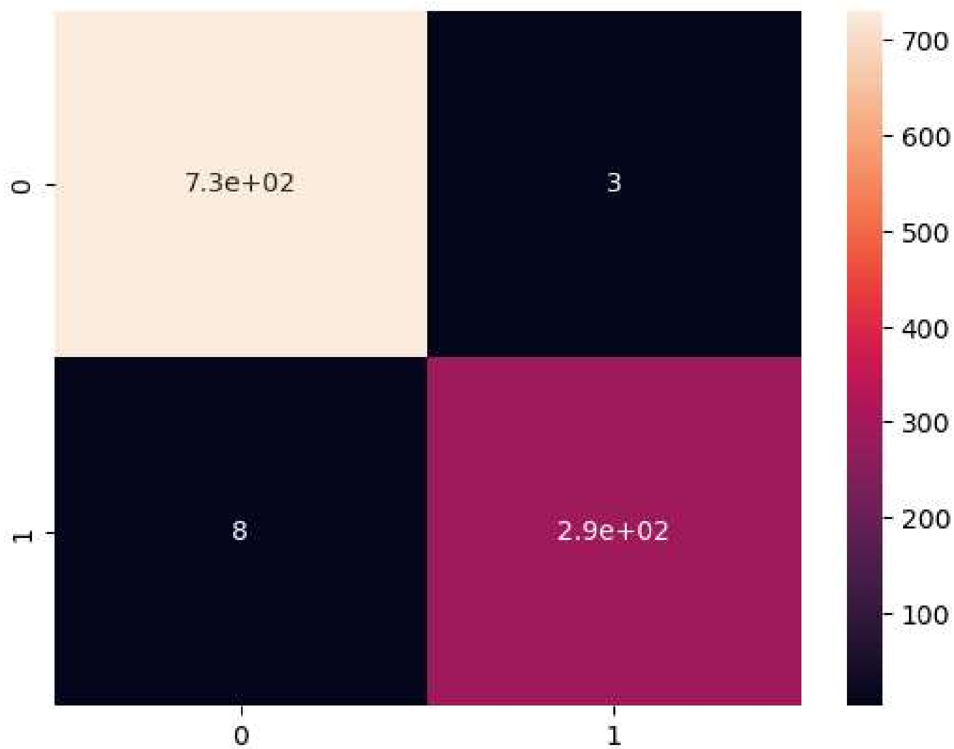

Fig.3.3.2 Naïve bayes Classifier Confusion Matrix

3.4 Logistic Regression:

The code trains a Logistic Regression classifier (LogisticRegression()) on the training data and makes predictions on the testing data.

```
In [49]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score
         lr=LogisticRegression(C=10.0)
         lr.fit(X_train,Y_train)
         Y_pred=lr.predict(X_test)
         confusion_matrix(Y_test,Y_pred)
         print(classification_report(Y_test,Y_pred))
         print("The Accuracy is ",accuracy_score(Y_test,Y_pred))
         sns.heatmap(confusion_matrix(Y_test,Y_pred),annot=True)
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       734
           1       0.97      0.97      0.97       301

    accuracy                           0.98      1035
   macro avg       0.98      0.98      0.98      1035
weighted avg       0.98      0.98      0.98      1035

The Accuracy is  0.9826086956521739
```

Fig.3.4.1 Logistic Regression Result

```
Out[49]: <Axes: >
```

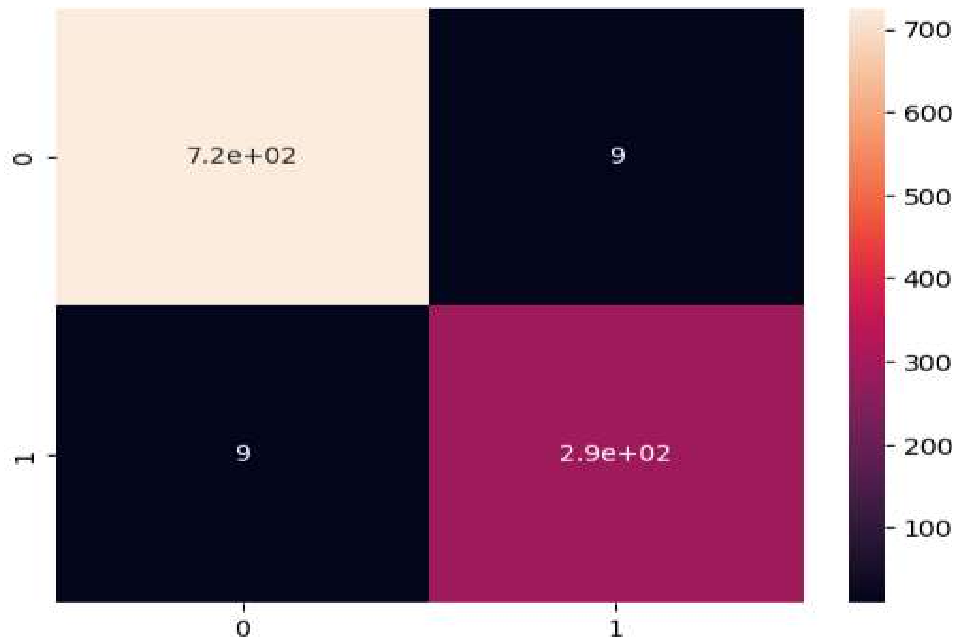

Fig.3.4.2 Logistic Regression Confusion Matrix

3.5 Decision Tree Classifier:

The code trains a Decision Tree classifier (DecisionTreeClassifier()) on the training data and makes predictions on the testing data.

```
In [23]: from sklearn.tree import DecisionTreeClassifier
         from sklearn import tree
         dt=DecisionTreeClassifier(max_depth=5)
         dt.fit(X_train,Y_train)
         Y_pred=dt.predict(X_test)
         print(classification_report(Y_test,Y_pred))
         dtc=DecisionTreeClassifier(max_depth=3)
         dtc.fit(X_train,Y_train)
         plt.figure(figsize=(20,20))
         tree.plot_tree(dt)
         plt.show()
         print(accuracy_score(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.78      0.87       734
           1       0.64      0.98      0.78       301

    accuracy                           0.84      1035
   macro avg       0.82      0.88      0.82      1035
weighted avg       0.89      0.84      0.84      1035
```

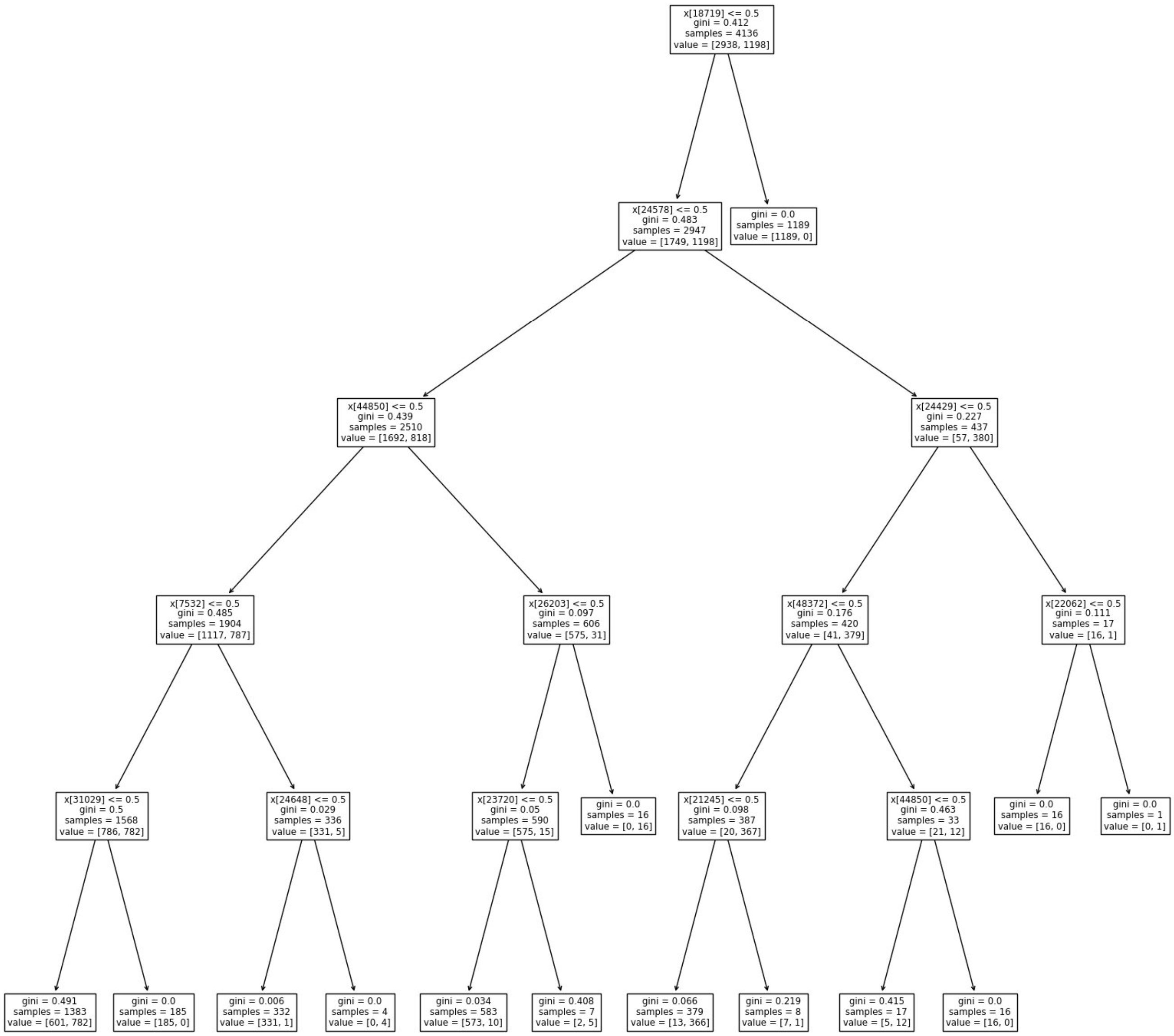

Fig.3.5.2 Decision Tree visualization

3.6 K-Nearest Neighbors Classifier:

The code trains a K-Nearest Neighbors classifier (KNeighborsClassifier()) on the training data and makes predictions on the testing data.

```
In [25]: from sklearn.neighbors import KNeighborsClassifier
         neigh = KNeighborsClassifier(n_neighbors=3)
         neigh.fit(X_train, Y_train)
         Y_pred=neigh.predict(X_test)
         print(classification_report(Y_test,Y_pred))
         print(accuracy_score(Y_test,Y_pred)
         sns.heatmap(confusion_matrix(Y_test,Y_pred),annot = True)
```

```
              precision    recall  f1-score   support

           0       0.95      0.88      0.91       734
           1       0.75      0.88      0.81       301

    accuracy                           0.88      1035
   macro avg       0.85      0.88      0.86      1035
weighted avg       0.89      0.88      0.88      1035

Out[25]: <Axes: >
```

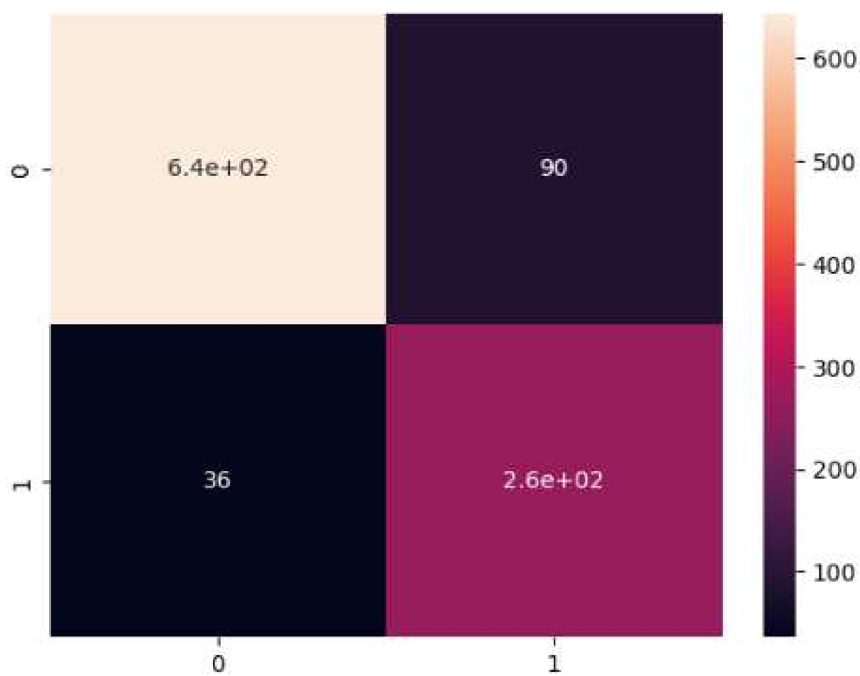

Fig.3.6 Knn Result

3.7 Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees to create a more robust and accurate predictive model. It is a supervised learning algorithm used for both classification and regression tasks.

```
In [57]: from sklearn.ensemble import RandomForestClassifier
         rf=RandomForestClassifier(n_estimators=500,min_samples_split=2,max_depth=10,max_features=7)
         rf=rf.fit(X_train,Y_train)
         cross_val_score(rf,X_train,Y_train,cv=10).mean()
         print("The Accuracy Score is",accuracy_score(Y_test,Y_pred))
         sns.heatmap(confusion_matrix(Y_test,Y_pred),annot = True)

         The Accuracy Score is 0.5855072463768116

Out[57]: <Axes: >
```

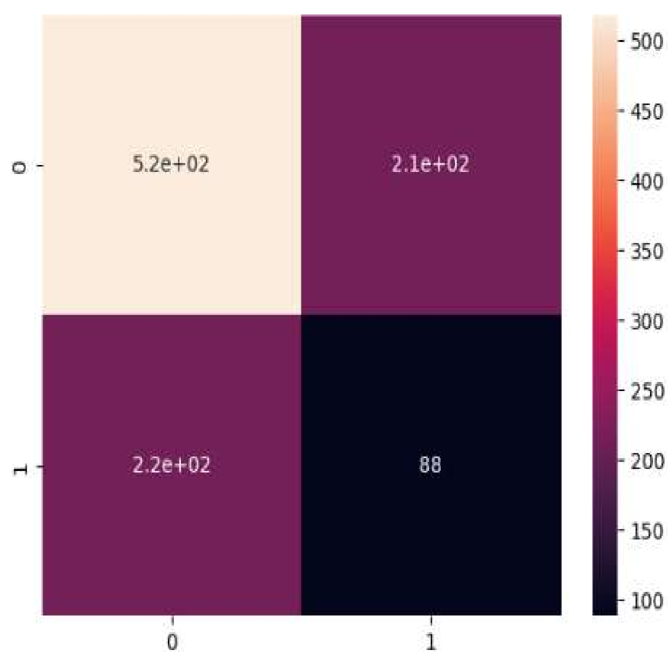

Fig.3.7 Random Forest report

3.8 SVM

SVM (Support Vector Machines) is a supervised machine learning algorithm used for classification and regression tasks. It is a powerful algorithm that can handle both linear and non-linear relationships between variables.

```
In [53]: from sklearn import svm
         model=svm.SVC(kernel='linear')
         model.fit(X_train,Y_train)
         Y_pred=model.predict(X_test)
         print(classification_report(Y_test,Y_pred))
         print("The Accuracy is",accuracy_score(Y_test,Y_pred))
         sns.heatmap(confusion_matrix(Y_test,Y_pred),annot = True)

              precision    recall  f1-score   support

           0       0.98      0.98      0.98       734
           1       0.95      0.96      0.96       301

    accuracy                           0.97      1035
   macro avg       0.97      0.97      0.97      1035
weighted avg       0.97      0.97      0.97      1035

The Accuracy is 0.9739130434782609
```

Fig.3.8.1 SVM Result

```
Out[53]: <Axes: >
```

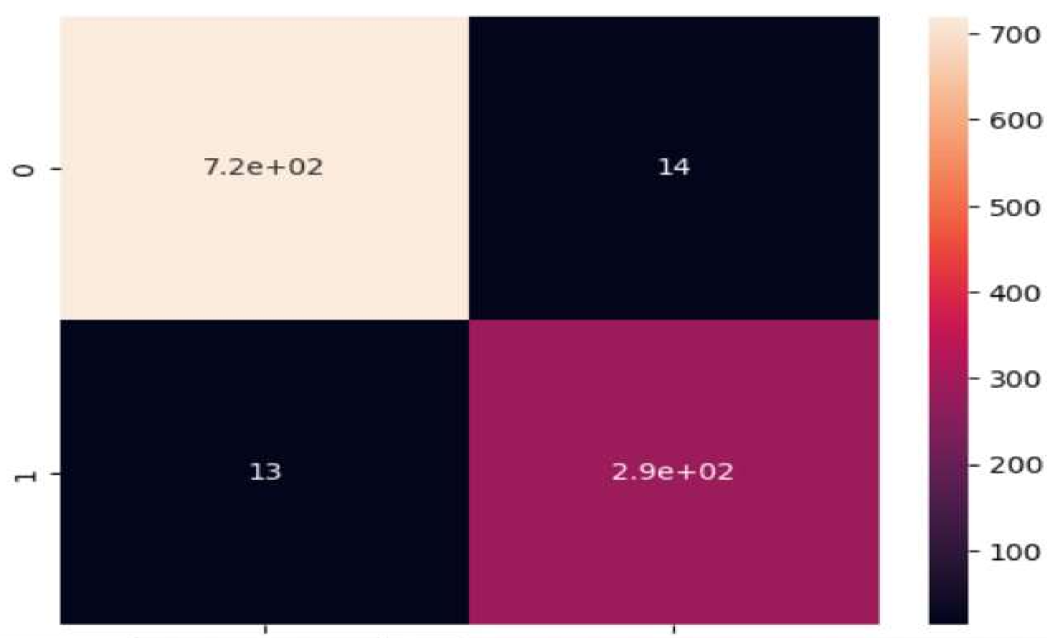

Fig.3.8.2 SVM Confusion Matrix

3.9 Classification Reports:

The code generates classification reports using Scikit-learn'sclassification_report() function to evaluate the performance of the classifiers.

```
print(classification_report(y_test,y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       731
           1       0.98      0.97      0.97       304

    accuracy                           0.98      1035
   macro avg       0.98      0.98      0.98      1035
weighted avg       0.98      0.98      0.98      1035
```

3.9 classification Report

3.10 Accuracy Scores:

The code calculates accuracy scores using Scikit-learn'saccuracy_score() function to measure the accuracy of the classifiers.

```
fromsklearn.metricsimportclassification_report,confusion_matrix,accuracy_score
```

3.11 Confusion Matrix Visualization:

The code uses Seaborn'sheatmap() function to visualize the confusion matrices generated by the classifiers.

```
fromsklearn.metricsimportclassification_report,confusion_matrix,accuracy_score
```

CHAPTER 4

IMPLEMENTATION

4.1 SOURCE CODE

```
import pandas aspd
fromsklearn.feature_extraction.textimportCountVectorizer
fromsklearn.naive_bayesimportMultinomialNB
importmatplotlib.pyplotasplt
importseabornassns
importgradioas gr
```

```
ds = pd.read_csv('spam_ham_dataset.csv')
ds
```

```
defdata_set():
    spam_df  =pd.read_csv('spam_ham_dataset.csv')
    returnspam_df.drop('Unnamed: 0',axis=1)
```

```
data = data_set()
data
data.shape

data.tail()
data.info()
data.describe()
```

```
plt.hist(data['label'])
plt.show()
```

```
fromsklearn.feature_extraction.textimportCountVectorizer
vectorizer=CountVectorizer()
spamham_countVectorizer=vectorizer.fit_transform(data['text'])
```

```
label=data['label_num']
X=spamham_countVectorizer
y=label
```

```
fromsklearn.model_selectionimporttrain_test_split,cross_val_score
fromsklearn.metricsimportclassification_report,confusion_matrix,accuracy_score
X_train,X_test,Y_train,Y_test=train_test_split(X,y,test_size=0.2)
```

```
#Naive Bayes

NB_classifier=MultinomialNB()
NB_classifier.fit(X,y)
Y_pred=NB_classifier.predict(X_test)
cm=confusion_matrix(Y_test,Y_pred)
print(classification_report(Y_test,Y_pred))
```

```
print("The Accuracy is",accuracy_score(Y_test,Y_pred))
sns.heatmap(cm,annot=True)
```

```
fromsklearn.linear_modelimportLogisticRegression
fromsklearn.metricsimportaccuracy_score
lr=LogisticRegression(C=10.0)
lr.fit(X_train,Y_train)
Y_pred=lr.predict(X_test)
confusion_matrix(Y_test,Y_pred)
print(classification_report(Y_test,Y_pred))
print("The Accuracy is ",accuracy_score(Y_test,Y_pred))
sns.heatmap(confusion_matrix(Y_test,Y_pred),annot=True)
```

```
fromsklearn.treeimportDecisionTreeClassifier
fromsklearnimport tree
dt=DecisionTreeClassifier(max_depth=5)
dt.fit(X_train,Y_train)
Y_pred=dt.predict(X_test)
print(classification_report(Y_test,Y_pred))
dtc=DecisionTreeClassifier(max_depth=3)
dtc.fit(X_train,Y_train)
plt.figure(figsize=(20,20))
tree.plot_tree(dt)
plt.show()
print("The Accuracy is",accuracy_score(Y_test,Y_pred))
```

```
fromsklearn.neighborsimportKNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, Y_train)
Y_pred=neigh.predict(X_test)
print(classification_report(Y_test,Y_pred))
print("The Accuracy is",accuracy_score(Y_test,Y_pred))
sns.heatmap(confusion_matrix(Y_test,Y_pred),annot = True)
```

```
fromsklearnimportsvm
model=svm.SVC(kernel='linear')
model.fit(X_train,Y_train)
Y_pred=model.predict(X_test)
print(classification_report(Y_test,Y_pred))
print("The Accuracy is",accuracy_score(Y_test,Y_pred))
sns.heatmap(confusion_matrix(Y_test,Y_pred),annot = True)
```

```
fromsklearn.ensembleimportRandomForestClassifier
rf=RandomForestClassifier(n_estimators=500,min_samples_split=2,max_depth=10,max_feat
ures=7)
rf=rf.fit(X_train,Y_train)
cross_val_score(rf,X_train,Y_train,cv=10).mean()
```

```python
print("The Accuracy Score is",accuracy_score(Y_test,Y_pred))
sns.heatmap(confusion_matrix(Y_test,Y_pred),annot = True)
```

```python
defcheck(text):
    a= []
    df =data_set()
    X = df['text']
    y = df['label_num']
    vectorizer=CountVectorizer()
    X = vectorizer.fit_transform(X)
    new_email_features = vectorizer.transform([text])

    model = MultinomialNB()
    model.fit(X, y)
    prediction = model.predict(new_email_features)
    a.append(prediction[0])

    dt=DecisionTreeClassifier(max_depth=5)
    dt.fit(X,y)
    prediction = dt.predict(new_email_features)
    a.append(prediction[0])

    lr=LogisticRegression(C=10.0)
    lr.fit(X,y)
    prediction = lr.predict(new_email_features)
    a.append(prediction[0])

    neigh = KNeighborsClassifier(n_neighbors=3)
    neigh.fit(X,y)
    prediction = neigh.predict(new_email_features)
    a.append(prediction[0])

    model=svm.SVC(kernel='linear')
    model.fit(X,y)
    prediction=model.predict(new_email_features)
    a.append(prediction[0])


    ifa.count(0) <a.count(1):
        return"The email is spam!"
    else:
        return"The email is not spam."
```

```python
app1 = gr.Interface(fn=data_set,inputs=None,
outputs=gr.Dataframe(),description="Reading csv")
app3 = gr.Interface(fn=check, inputs='text', outputs='text',description=" result")
demo = gr.TabbedInterface([app1, app3], ["Reading_csv", "result"])
demo.launch()
```

4.2 RESULT

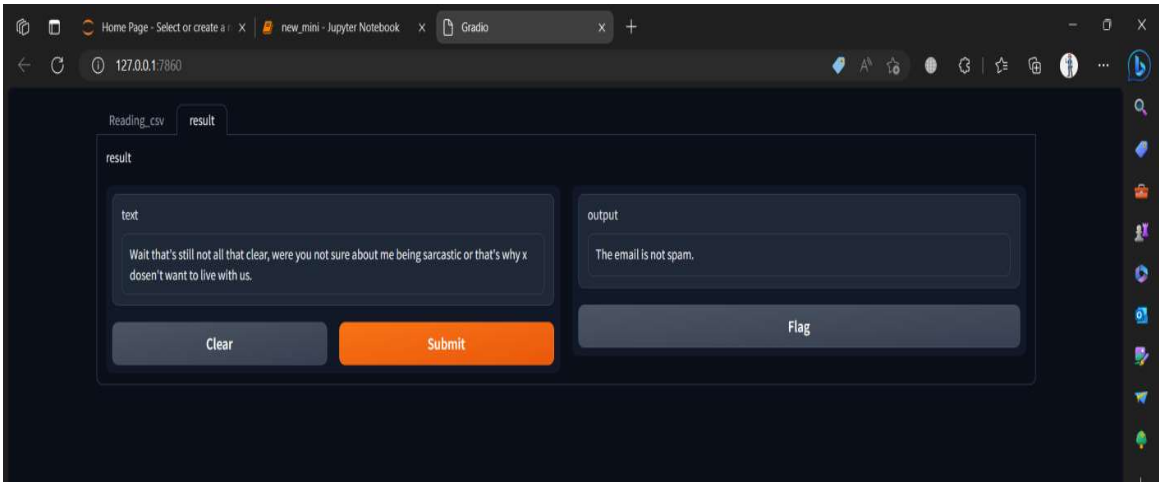

Fig.4.2.1 Result 1

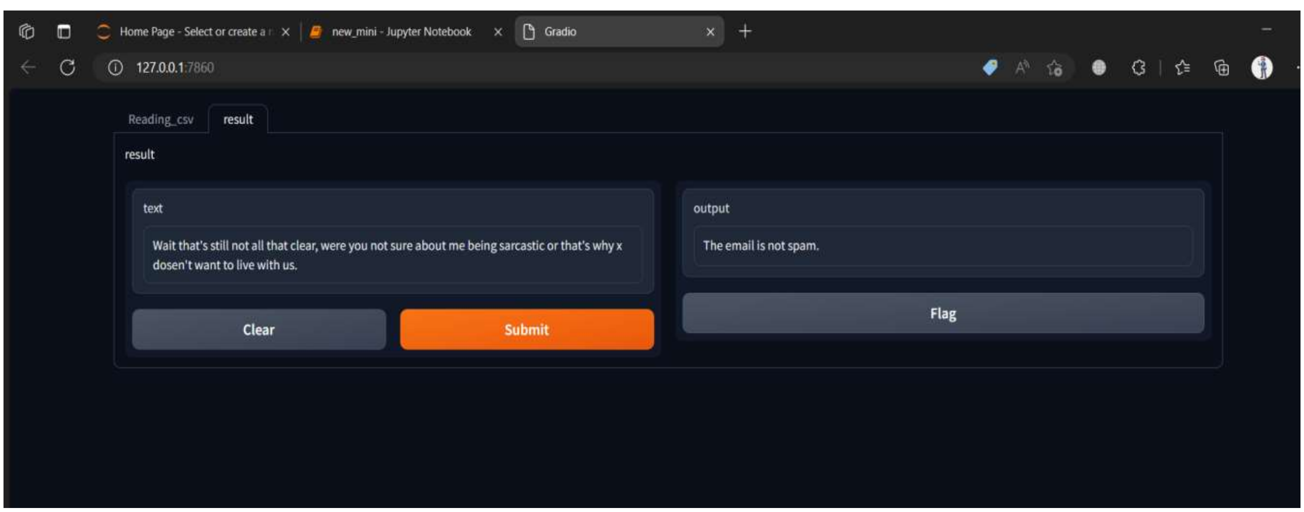

Fig.4.2.2 Result 2

CHAPTER 5
CONCLUSION

In conclusion, this project provides a solid foundation for spam mail detection using the Naive Bayes classifier. The developed system showcases promising results and serves as a starting point for future advancements in the field. By incorporating more sophisticated techniques and exploring additional features, we can continue to improve the accuracy and efficiency of spam mail detection systems, thereby contributing to a safer and more reliable email communication environment.

CHAPTER 6
BIBLIOGRAPHY

1. https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/
2. https://www.javatpoint.com/fit-transform-and-fit_transform-methods-in-python
3. https://gradio.app/docs/
4. https://www.javatpoint.com/machine-learning-naive-bayes-classifier
5. http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html