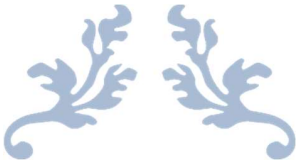

SPAM HAM CLASSIFIER REPORT

MOUNIKA BHARGAVI GIRIDI

ABSTRACT

Nowadays, we frequently use e-mails or messages, one of the communication channels, in an electronic environment. It plays an important role in our lives because of many reasons such as personal communications, marketing, advertising etc. E-mails and messages make our life easier because of meeting many different types of communication needs. On the other hand, they can make life difficult when they are used outside of their purposes. Hackers can steal information, leading to data theft, time wasted, and other security concerns. This brings us to the need to introduce a filter that can segregate emails into spam and ham. ‘Ham’ refers to genuine mail that is important to the user and is informative for his means. In contrast, ‘Spam’ is a spurious mail sent from unreliable sources with harmful intentions. Spam emails can be not only annoying to receivers, but also dangerous for receiver's information security. Detecting and preventing spam e-mails has been a separate issue. Thus, Machine learning algorithms can be used to identify spam or ham by fitting the system with the respective labels and using the trained model to make predictions or classification as the project objective is to classify the text in spam or ham which is Binary Classification problems so the project will test the performance of several classifiers like Naive Bayes, SVM, KNN. In this project we have used the Naive Bayes classification which is validated with simplicity and high accuracy.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 OVERVIEW

Apart from the numerous benefits and conveniences people around the world can enjoy due to the Internet, there are also multiple drawbacks. Not all of them are obvious to an average user, and perhaps only professional IT workers face them from time to time. However, there is a problem almost any Internet user has encountered at least once in a lifetime. Unlike many people might think, spam is not just an annoying but harmless email message or a social media message; in fact, spam can be a dangerous tool capable of harming its recipients, and should be outlawed.

Spam can cause real damage. If you wonder how a mere electronic letter can be harmful, first recall the usual contents of spam letters. Along with intrusive commercials and newsletters from electronic shops you have used only once, every email and social media user is also at risk of receiving spam letters advertising pornography, weapons, and other questionable content.

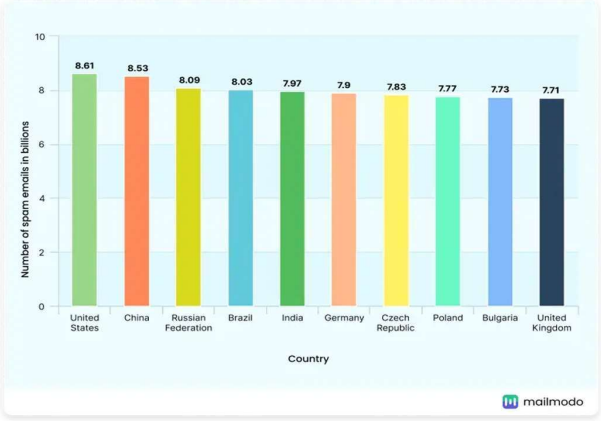

Fig 1.1 Spam Email Rates(Country Wise)

1.2 APPLICATION

Email/text messages have become a crucial part of our daily life as it is handy and easy to use. Since the user batch is enormous and contains a lot of sensitive information, it is susceptible to being compromised. Hackers can steal your information, leading to data theft, time wasted, and other security concerns. This brings us to the need to introduce a filter that can segregate emails into spam and ham.

Ham refers to genuine mail that is important to the user and is informative for his means. In contrast, spam is spurious mail sent from unreliable sources with harmful intentions. In this, we will be segregating such messages into spam/ham using Naive Bayes in machine learning.

Fig 1.2 Steps to prevent Spam

1.3 PROBLEM DEFINITION

The increased number of unsolicited emails known as spam has necessitated the development of increasingly reliable and robust anti-spam filters. Recent machine learning approaches have been successful in detecting and filtering spam emails.

1.4 AIM OF THE PROJECT

The goal is to build a web application to classify spam or ham from the dataset which 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 5,574 messages, tagged according to ham (legitimate) or spam.

2. SYSTEM REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS

2.1.1 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

2.1.2 VSCode

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

2.1.3 HTML

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

2.1.4 Dataset

Dataset Link: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection

The above dataset consists of a set of SMS tagged messages that have been collected for SMS Spam research. It contains 5572 rows and 2 columns. Each row represents the message in the text as spam or ham(not spam).

2.4 HARDWARE REQUIREMENTS

Windows 11

Processor: Minimum 1 GHz; Recommended 2GHz or more

Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

Hard Drive: Minimum 32 GB; Recommended 64 GB or more

Memory (RAM): Minimum 1 GB; Recommended 4 GB or above

3. PROPOSED SYSTEM / METHODOLOGY

This project takes a set of parameters from the user and based on which the algorithm classifies the arising possibilities of Spam/Ham.

The set of parameters are considered to be:

1. Email or

2.SMS

3.1 SYSTEM ARCHITECTURE / SYSTEM DESIGN / ALGORITHM

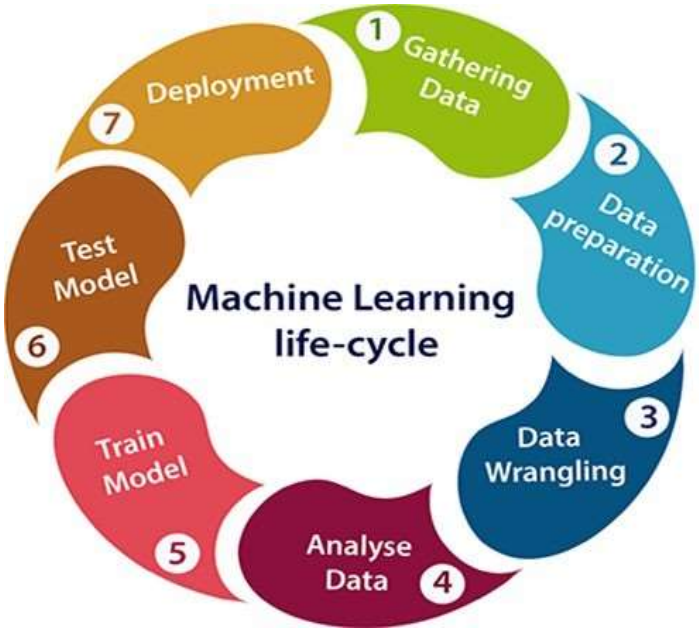

Fig 3.1 Machine Learning Process

Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

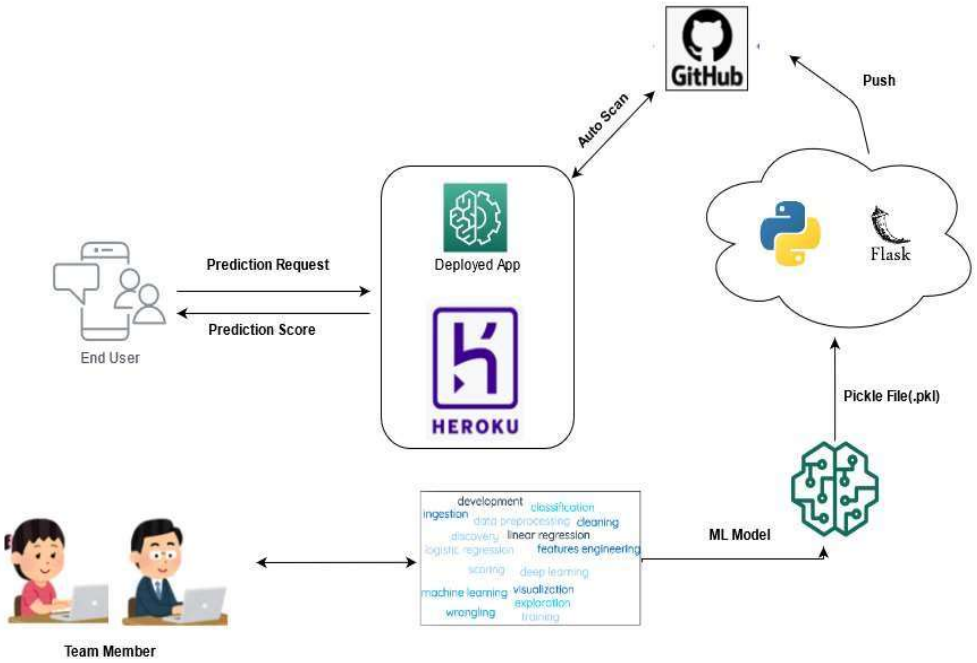

Fig 3.2 Structure of ML Model with Flask API and Heroku Deployment

- Create ML Model and save (pickle) it
- Create Flask files for UI and python main file (app.py) that can unpickle the machine learning model from step 1 and do predictions
- Create requirements.txt to setup Flask web app with all python dependencies
- Create Procfile to initiate Flask app command
- Commit files from Step 1, 2, 3 & 4 in the Github repo
- Create account/Login on Heroku, create an app, connect with Github repo, and select branch
- Select manual deploy (or enable Automatic deploys) on Heroku

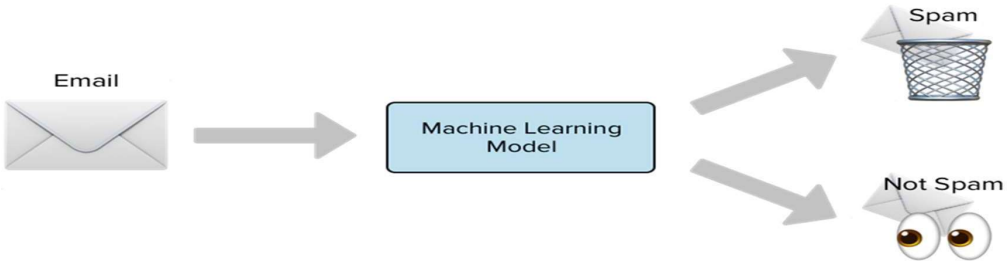

Fig 3.3 Flow chart

Here, First Email is sent through the Machine Learning model which further tells whether a particular predicted text/message is spam or not spam.

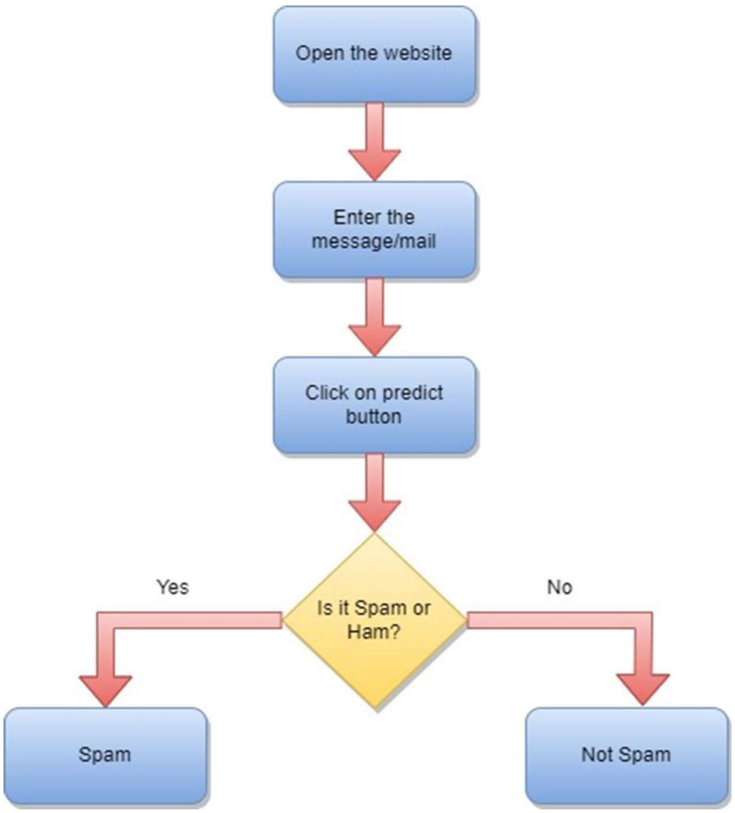

Fig 3.4 Project Flow

First open the website then enter the message/mail in textarea and click on predict button then it gives whether the message is spam or not spam.

3.2 IMPLEMENTATION AND RESULTS

i. Importing the required modules

```
import pandas as pd
import numpy as np
import pickle
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.metrics import accuracy_score,fbeta_score,classification_report
from wordcloud import WordCloud
from nltk.tokenize import word_tokenize
```

Fig 3.5 Importing Modules

ii. Importing and Exploring Dataset

```
msg_df = pd.read_csv('spam.csv', sep='\t', names=["label", "message"])
msg_df.shape
msg_df.head(5)
```

	label	message
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...

Fig 3.6 Exploring Dataset

iii. Data Visualization:

Before you apply machine learning algorithms to a dataset, it is always a good practice to visualize data to identify important data trends.

```
msg_df["label"].value_counts().plot(kind = 'pie', explode = [0, 0.1], figsize = (6, 6), autopct = '%1.2f%%')
plt.ylabel("Spam vs Ham")
plt.legend(["Ham", "Spam"])
plt.show()
```

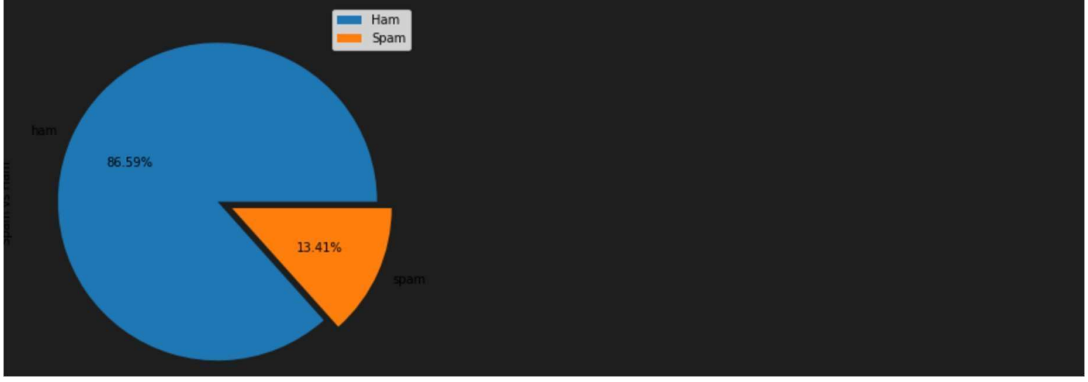

Fig 3.7 distribution of ham and spam messages in dataset

iv. The histogram for Spam and Ham

```
msg_df.hist(column='length', by='label', bins=50,figsize=(11,5))
```

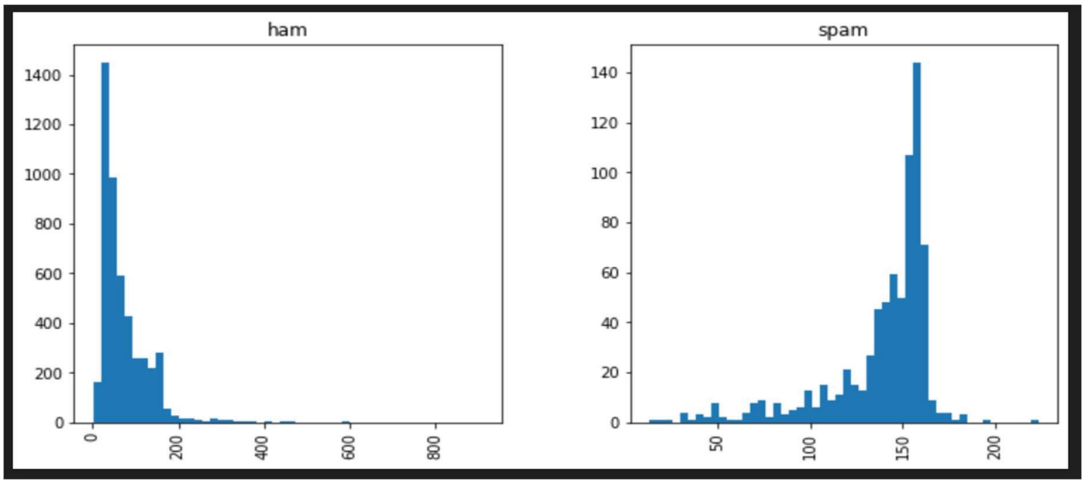

Fig 3.8 ham vs spam

v. Data Preprocessing:

Text data may contain special characters and digits. Most of the time these characters do not really play any role in classification. So, it is good to clean the text by removing special characters and digits.

```
import string
def cleanText(message):
    #message = message.translate(str.maketrans('ranjan', 'ranjan', string.punctuation))
    message = re.sub('[^a-zA-Z]', ' ', message)
    message = message.lower()
    message = message.split()
    words = [ss.stem(word) for word in message if word not in stop]
    return " ".join(words)

msg_df["message"] = msg_df["message"].apply(cleanText)
msg_df.head(n = 10)
```

	label	message	length
0	ham	go jurong point crazi avail bugi n great world...	111
1	ham	ok lar joke wif u oni	29
2	spam	free entri wkli comp win fa cup final tkts st ...	155
3	ham	u dun say earli hor u c alreadi say	49
4	ham	nah think goe usf live around though	61
5	spam	freemsg hey darl week word back like fun still...	147
6	ham	even brother like speak treat like aid patent	77
7	ham	per request mell mell oru minnaminungint nurun...	160
8	spam	winner valu network custom select receivea pri...	157

Fig 3.9 Data Cleaning

vi. Converting Text to Numbers:

Machine learning algorithms are statistical algorithms that work with numbers. Messages are in the form of text. So, we need to convert messages to text form. There are various ways to convert text to numbers.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vec = TfidfVectorizer(encoding = "latin-1", strip_accents = "unicode")
features = vec.fit_transform(msg_df["message"])
print(features.shape)
```

```
(5572, 6292)
```

Fig 3.10 Converting Text to Numbers(1)

vii. Counting the number of columns

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X=cv.fit_transform(msg_df["message"])
print (X.shape)
```

```
(5572, 6292)
```

Fig 3.11 Converting Text to Numbers(2)

viii. Training model using Naive bayes classifier by importing MultinomialNB

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df, y, test_size = 0.20, random_state = 0)

# Training model using Naive bayes classifier

from sklearn.naive_bayes import MultinomialNB
spam_detect_model = MultinomialNB().fit(X_train, y_train)

y_pred=spam_detect_model.predict(X_test)
```

Fig 3.12 Training model using Naive bayes classifier

ix. printing accuracy_score and fbeta_score.

```
print(accuracy_score(y_test,y_pred))
print(fbeta_score(y_test,y_pred,beta =0.5))
```

```
0.9811659192825112
0.9390862944162438
```

Fig 3.13 Accuracy score

x. App.py

```
from flask import Flask,render_template,url_for,request
import pickle
import joblib

filename = 'pickle.pkl'
clf = pickle.load(open(filename, 'rb'))
cv=pickle.load(open('tranform.pkl','rb'))
app = Flask(__name__)

@app.route('/')
def home():
        return render_template('home.html')

@app.route('/predict',methods=['POST'])
def predict():
        if request.method == 'POST':
                message = request.form['message']
                data = [message]
                vect = cv.transform(data).toarray()
                my_prediction = clf.predict(vect)
        return render_template('result.html',prediction = my_prediction)

if __name__ == '__main__':
        app.run(debug=True)
```

Fig 3.14 :app.py

xi. Frontend:
Home.html has the code for the starting page.

```
<!DOCTYPE html>
<html>
<head>
    <title>Home</title>
    <link rel="stylesheet" type="text/css"  href="static/styles.css">
</head>
<body>

    <header>
        <div class="container">
        <h1>SPAM HAM CLASSIFIER</h1>

    </div>
    </header>

    <div class="ml-container">

        <form action="/predict" method="POST">
        <h3>Enter Your Message/Mail Here which you want to predict Spam or Ham</h3>
        <textarea name="message" rows="6" cols="50" placeholder="Enter Your Message..." ></textarea>
        <br/>

        <input type="Submit" class="btn-info" value="Predict">

    </form>

    </div>
</body>
</html>
```

Fig 3.15 Home.html

result.html consists the HTML for result page

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <link rel="stylesheet" type="text/css"  href="static/styles.css">
</head>
<body>

    <header>
        <div class="container">
        <h1>SPAM HAM CLASSIFIER</h1>

    </div>
    </header>

    <h3 style="color: rgb(113, 5, 5);font-size:20;text-align: center;"><b>Results for Message</b></h3>
    <div class="results">

    {% if prediction == 1%}
    <h1 style="color: red ;text-align: center;">Spam &#128721; </h1>
    <img src="https://i.pinimg.com/736x/72/00/53/720053d34b1608ef685c978290ccada4.jpg" height="200" />
    {% elif prediction == 0%}
    <h1 style="color: green;text-align: center;" >Not a Spam  &#128140; </h1>
    <img src="https://img.freepik.com/premium-vector/monkey-head-smiling-broadly-with-hair-crest-listening-music-u
    {% endif %}

    </div>
```

Fig 3.16 result.html

xii. In style.css file we are defining the style for home.html and result.html

```
body {
    background-image: linear-gradient(rgba(255,255,255,0.6), rgba(255,255,255,0.6)),url("paper.jpg");
    background-size: 1300px;
    text-align: center;
    padding: 70px;
   }
   .background{
    opacity: 0.3;
   }
  .App {
    text-align: center;
    }
    .information {
    display: flex;
    flex-direction: column;
    width: 100%;
    justify-content: center;
    align-items: center;
    }

    input {  /* predict button*/
    width: 300px;
    height: 50px;
    font-size: 20px;
    padding-left: 10px;
    margin: 5px;
    background-color: rgb(128, 176, 201);
    }
```

Fig 3.17 styles.css

3.3 RESULTS

i. This is the homepage of the website where we have to enter the message in textarea and click on the predict button.

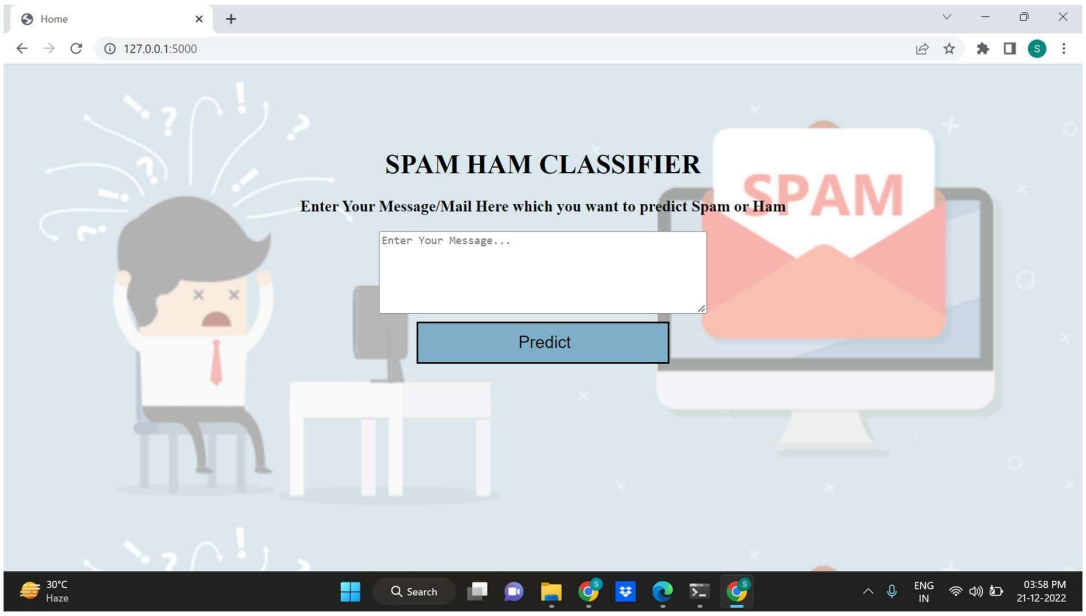

Fig 3.18 Home page

ii. In textarea we entered a message which is not spam

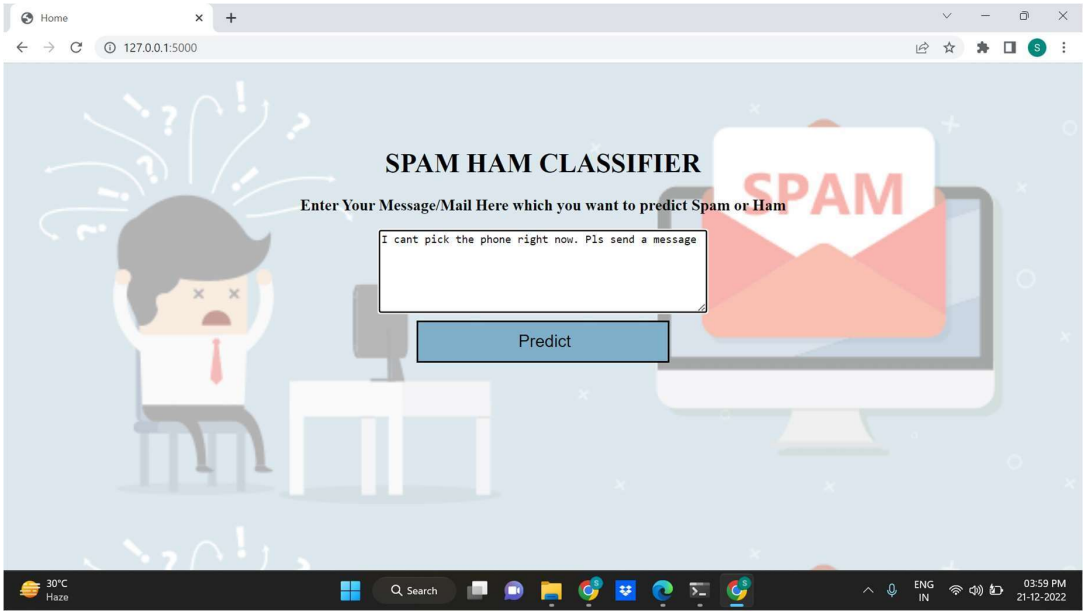

Fig 3.19 Home page(1)

iii. In the result page it gives the result as not spam for the message we entered in textarea.

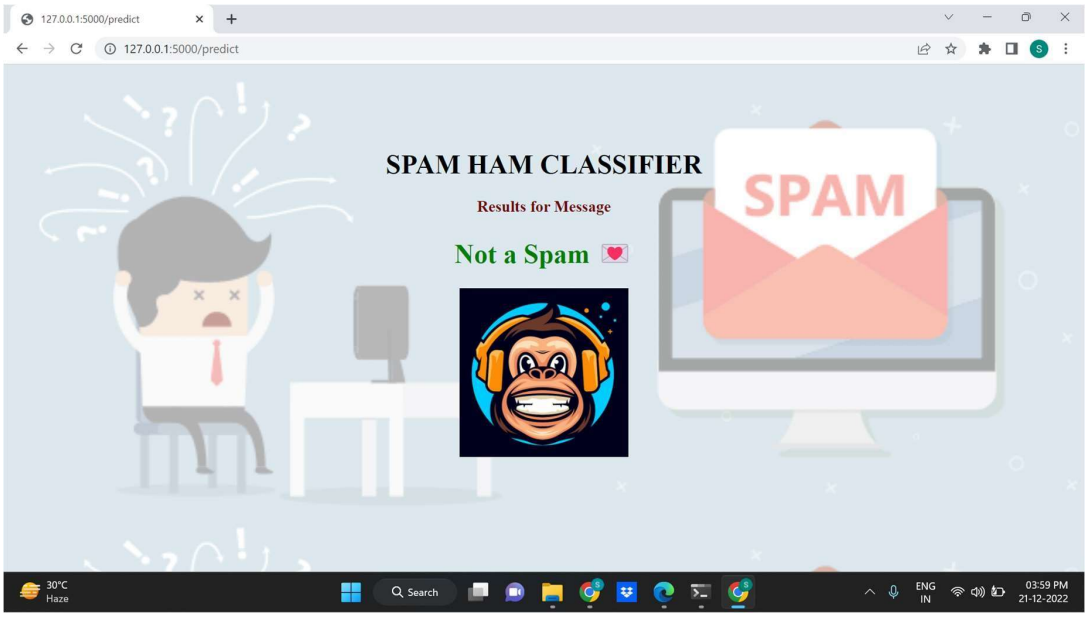

Fig 3.20 Result page(1)

iv. In textarea we entered a message which is spam

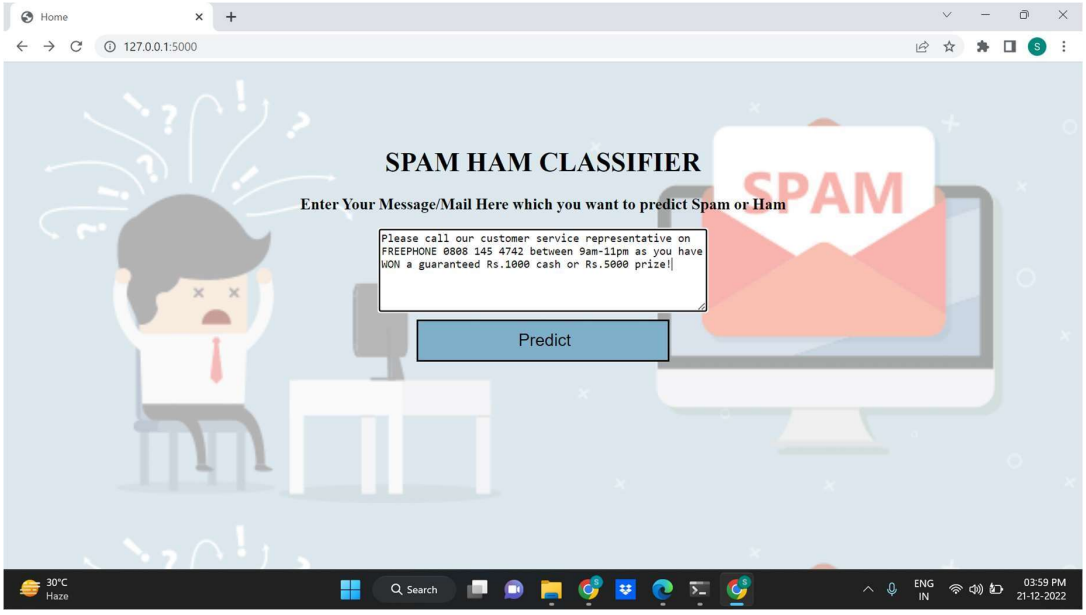

Fig 3.21 Home page(2)

v. In the result page it gives the result as spam for the message we entered in textarea.

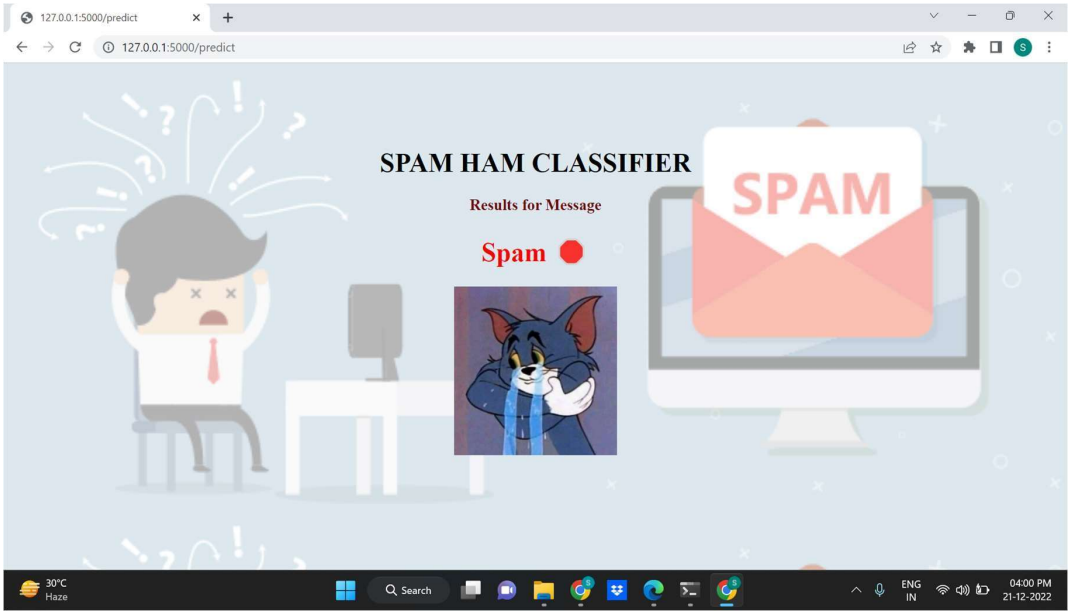

Fig 3.22 Result page(2)

4. CONCLUSION

Through this project we provide a user-friendly interface to users which helps users to classify Spam or ham from the dataset which is a set of SMS tagged messages that have been collected for SMS Spam research. This is a Spam Classifier web application built using Flask. This project takes a message/email as an input and predicts the message/email as spam or not spam (ham). The prediction is done on the basis of a dataset which contains 5572 rows and 2 columns. Each row represents the message in the text as spam or ham(not spam). For this dataset, linear models with high bias and low variance like the Naïve Bayes classifier have performed better than non-linear models. The 'home.html' page takes the required details from the user such as mail/messages. When you click on the predict we will be directed towards the 'result.html' page which shows where the given details are spam/ham.

BIBLIOGRAPHY

[1] https://www.tutorialspoint.com/flask/index.htm

[2] https://www.youtube.com/watch?v=inp6h0HDN74&t=2s

[3] https://dashboard.heroku.com/apps

[4] https://archive.ics.uci.edu/ml/datasets/sms+spam+collection