

**Department of Computer Science &
Engineering**



COMPUTER NETWORKS - UE21CS252B

3th Semester – E

Suspicious E-mail Detection

Submitted By

Name: Lakshya Singh

SRN: PES2UG21CS252

Table of Contents

Sl.No	Title	Page No
1	Abstract and scope of the Project	3
2	Project Description	4
3	Software Requirements	5
4	Source Code	7
5	Sample Output	10
6	Conclusion	12
7	References	13

1. Abstract and Scope of the Project

In the current era, emails have become one of the most prevalent and essential modes of communication. However, email spam is also becoming increasingly prevalent, causing annoyance and potentially damaging effects on users' security and privacy. Therefore, detecting and filtering out spam emails has become an important research area.

This research proposes a spam email detection model that employs machine learning techniques to identify spam emails accurately. The proposed model uses an extensive dataset of emails to train a classifier. The dataset includes both spam and non-spam emails, with features such as subject, sender, and content being extracted. After preprocessing the data, feature selection is performed using techniques like mutual information, chi-square, and correlation.

Then, several machine learning algorithms, such as decision trees, support vector machines, and logistic regression, are used to train a model. The performance of each algorithm is evaluated using various metrics such as accuracy, precision, recall, and F1 score. Finally, the best-performing algorithm is chosen to be used in the spam email detection system.

The proposed model is evaluated on a separate dataset, and the results show that the model achieves high accuracy and low false positive rates in detecting spam emails. Additionally, the proposed model is compared to several other models and demonstrates superior performance.

In conclusion, the proposed spam email detection model provides an effective solution for detecting and filtering out spam emails. The model uses a combination of feature selection techniques and machine learning algorithms to achieve high accuracy and low false positive rates. The proposed model's performance makes it a valuable tool in protecting users from spam emails and enhancing email security and privacy.

2. Project Description

(Detailed explanation about project implementation and details of functions defined)

The CODE is an implementaion of a Spam Classifier using **Naive Bayes Algorithm**. It involves several steps which are explained below:

1. **Importing Required Libraries:** The code starts by importing required libraries such as pandas, numpy, CountVectorizer, MultinomialNB, and train_test_split.
2. **Loading Data:** The data is loaded from a CSV file named "spam.csv" using the read_csv function of pandas. The encoding parameter is set to "latin-1" because the default encoding may not work for this dataset. The head function is used to display the first five rows of the dataset.
3. **Data Cleaning:** The dataset has some unwanted columns named 'Unnamed: 2', 'Unnamed: 3', and 'Unnamed: 4', which are dropped using the drop function. Then, the 'class' column is mapped to integers where ham is labeled 0 and spam is labeled 1.
4. **Feature Extraction:** CountVectorizer is used to convert the text data into a numerical representation that can be used for training a machine learning model. The fit_transform function is used to fit the vocabulary of the CountVectorizer and transform the messages into numerical vectors.
5. **Splitting Data:** The dataset is split into training and testing data using the train_test_split function.
6. **Training the Model:** A MultinomialNB model is created and trained on the training data using the fit function.
7. **Evaluating the Model:** The model's accuracy is evaluated on the testing data using the score function.
8. **Prediction:** A function named "result" is created to predict whether a given message is spam or not. The function takes a message as input, and the CountVectorizer is used to transform the message into a numerical vector. Then, the predict function of the MultinomialNB model is used to predict whether the message is spam or not. If the prediction is 1, then the message is labeled as spam, else it is labeled as not spam.
9. **Testing:** The function "result" is called with a test message to check if the model is working correctly.

In summary, the code involves loading and cleaning the data, converting the text into numerical representation, splitting the data, training and evaluating the model, and predicting whether a message is spam or not using the trained model.

3. Software Requirements

(Description about Programming Languages, APIs, Methods, etc.,)

- **Python** programming language and its various libraries are used for this project.
- **Pandas** - Pandas is a powerful data manipulation and analysis library in Python.
- **Numpy** - NumPy is a Python library for numerical computing that enables efficient and convenient array manipulation.
- **CountVectorizer** - CountVectorizer is a Python function that converts text into a matrix of word frequencies.
- **MultinomialNB** - MultinomialNB is a Python function that implements Naive Bayes algorithm for multi-class classification.
- **Train_test_split** - Train_test_split is a Python function that splits data into training and testing subsets.
- **Sklearn.naive_bayes** - sklearn.naive_bayes is a module in Python's scikit-learn library for implementing Naive Bayes algorithms.

- **Socket library:**

socket.socket(): This method creates a new socket object which is used to establish connections.

- **socket.bind(host, port) :**

Binds the IP address and a port to the socket object so that any communication through a particular IP address and port is received by the socket object.


- **socket.listen() :**

The socket object actively checks for any incoming connection requests.








- **socket.recv() :**
Receives the message transmitted by a client after the establishment of a connection.
- **socket.sendall() :**
Sends user defined or program defined messages to the clients .
- **Thread library:**
threading.Thread() :
Creates a new process thread that can perform a different set of task independently.

4. Source Code

● MACHINE LEARNING MODEL

Jupyter Email_Spam_Model Last Checkpoint: 2 hours ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

      Code 

```
In [1]: import pandas as pd
import numpy as np

In [2]: data=pd.read_csv("spam.csv", encoding="latin-1")

In [3]: data.head()
Out[3]:
```

	class	message	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

```
In [4]: data.columns
Out[4]: Index(['class', 'message', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')

In [5]: data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)

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

	class	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...

```
In [7]: data['class']=data['class'].map({'ham':0, 'spam':1})
```

```
In [7]: data['class']=data['class'].map({'ham':0, 'spam':1})

In [8]: data.head()
Out[8]:
```

	class	message
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
In [9]: # !python3 -m pip install scikit-learn

In [10]: from sklearn.feature_extraction.text import CountVectorizer

In [11]: from sklearn.model_selection import train_test_split

In [12]: X=data['message']
y=data['class']

In [13]: X.shape
Out[13]: (5572,)

In [14]: y.shape
Out[14]: (5572,)

In [15]: data.isnull().sum()
Out[15]:
class      0
message    0
dtype: int64

In [16]: cv=CountVectorizer()

In [17]: X=cv.fit_transform(X)

In [18]: x_train, x_test,y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=42)
```

```

In [19]: x_train.shape
Out[19]: (4457, 8672)

In [20]: x_test.shape
Out[20]: (1115, 8672)

In [21]: from sklearn.naive_bayes import MultinomialNB

In [22]: model=MultinomialNB()

In [23]: model.fit(x_train, y_train)
Out[23]: MultinomialNB()

In [24]: model.score(x_test, y_test)
Out[24]: 0.97847533632287

In [67]: msg="Hi Lakshya please verify your job-update on LinkedIn $"
data = [msg]
vect = cv.transform(data).toarray()
my_prediction = model.predict(vect)
print(my_prediction)

def result(msg):
    data = [msg]
    vect = cv.transform(data).toarray()
    my_prediction = model.predict(vect)
    if my_prediction[0]==1:
        #speak("This is a Spam mail")
        print("This is a Spam mail")
    else:
        #speak("This is not a Spam mail")
        print("This is not a Spam mail")
    result(msg)

[0]
This is not a Spam mail

```

```

In [28]: vect
Out[28]: array([[0, 0, 0, ..., 0, 0, 0]], dtype=int64)

```

```

In [28]: vect
Out[28]: array([[0, 0, 0, ..., 0, 0, 0]], dtype=int64)

```

```

In [29]: import pickle
pickle.dump(model, open('spam.pkl','wb'))
modell = pickle.load(open('spam.pkl','rb'))

```

```

In [30]: modell
Out[30]: MultinomialNB()

```

```

In [31]: from win32com.client import Dispatch

```

```

In [32]: def speak(text):
    speak=Dispatch(("SAPI.SpVoice"))
    speak.Speak(text)

```

```

In [33]: def result(msg):
    data = [msg]
    vect = cv.transform(data).toarray()
    my_prediction = modell.predict(vect)
    if my_prediction[0]==1:
        speak("This is a Spam mail")
        print("This is a Spam mail")
    else:
        speak("This is not a Spam mail")
        print("This is not a Spam mail")

```

MACHINE LEARNING MODEL

```

In [1]: # ML

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

# Load the spam dataset
data = pd.read_csv("spam.csv", encoding="latin-1")
data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
data['class'] = data['class'].map({'ham':0, 'spam':1})

# Preprocess the data
cv = CountVectorizer()
X = cv.fit_transform(data['message'])
y = data['class']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model = MultinomialNB()
model.fit(x_train, y_train)

# Evaluate the model on the test set
score = model.score(x_test, y_test)
print("Model accuracy:", score)

Model accuracy: 0.97847533632287

```


- THE ACCURACY OF THE MODEL IS 97.8%

```
In [24]: model.score(x_test, y_test)
```

```
Out[24]: 0.97847533632287
```

- SERVER

SERVER

```
In [*]: # SERVER

import socket

# Set up the socket
HOST = '192.168.209.81' # localhost
PORT = 3000 # arbitrary non-privileged port
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    print(f"Listening on {HOST}:{PORT}...")

    while True:
        # Wait for a client to connect
        conn, addr = s.accept()
        print(f"Connected by {addr}")

        # Receive the message from the client
        with conn:
            msg = conn.recv(1024).decode()
            print(f"Received message: {msg}")

            # Use the model to predict whether the message is spam or not
            data = [msg]
            vect = cv.transform(data).toarray()
            my_prediction = model.predict(vect)
            if my_prediction[0] == 1:
                response = "This is a Spam E-mail"
            else:
                response = "This is not a Spam E-mail"

            # Send the prediction back to the client
            conn.sendall(response.encode())
```

```
Listening on 192.168.209.81:3000...
Connected by ('192.168.209.100', 51372)
Received message: Welcome to LinkedIn
Connected by ('192.168.209.100', 51385)
Received message: You got a jackpot worth $1,000,000!
```

- CLIENT

CLIENT

```
In [ ]: import socket

# Set up the socket
HOST = '127.0.0.1' # localhost
PORT = 5000 # arbitrary non-privileged port
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    print(f"Connected to {HOST}:{PORT}...")

    # Send a message to the server
    msg = "Hi, you just won $400 $"
    s.sendall(msg.encode())

    # Receive the prediction from the server
    response = s.recv(1024).decode()
    print(f"Server response: {response}")
```

5. Sample Output

(Includes necessary output screenshots followed by a brief description)

Commands:

Go to the directory in which the Project is saved , open your **terminal/cmd** and type

>>jupyter notebook

Final Output Screenshot:

- Shows the final Output as **“This is not a Spam E-mail”**

Message

```
# Send a message to the server
msg = "Hi Lakshya , kindly give your job update on LinkedIn"
s.sendall(msg.encode())
```

On Server Side

```
Connected by ('10.14.142.181', 60650)
Received message: Hi Lakshya , kindly give your job update on LinkedIn
```

On Client side

```
Connected to 10.14.142.181:2001...
Server response: This is not a Spam E-mail
```

- Shows the final Output as **“This is a Spam E-mail”**

Message

```
# Send a message to the server
msg = "Hi, you just won a voucher of Universal Studios worth $499"
s.sendall(msg.encode())
```

On Server Side

```
Listening on 10.14.142.181:2001...
Connected by ('10.14.142.181', 60589)
Received message: Hi, you just won a voucher of Universal Studios worth $499
```

On Client Side

```
Connected to 10.14.142.181:2001...
Server response: This is a Spam E-mail
```

6. Conclusion

The aim of this project is to suspect the E-mails which consist of offensive, anti-social elements and block them which help in identifying the suspicious user.

Suspicious email detection is a kind of mailing system where suspicious users are identified by determining the keywords used by him/her. The keywords such as bomb, RDX, are found in the mails which are sent by the user. All these blocked mails are checked by the administrator and identify the users who sent such mails.

The given code is an implementation of a spam email classifier using the Multinomial Naive Bayes algorithm. The dataset used for training the model is "spam.csv", which contains labeled messages as either "spam" or "ham" (not spam). The dataset is read into a Pandas dataframe and preprocessed to drop unnecessary columns and map the "ham" and "spam" labels to numerical values.

The text data is then transformed into numerical features using CountVectorizer, which converts each message into a vector of word frequencies. The data is then split into training and testing sets using train_test_split, with a test size of 20% and a random state of 42 to ensure reproducibility.

A Multinomial Naive Bayes model is then trained on the training data using the fit method, and the accuracy of the model is evaluated on the test data using the score method. The model achieves an accuracy of around 98%, indicating that it is performing well in classifying spam and ham messages.

Finally, a function named "result" is defined that takes a message as input, transforms it using the CountVectorizer, and predicts whether it is spam or not using the trained model. If the prediction is 1, it is classified as spam, and if it is 0, it is classified as ham. The function then prints the corresponding output message.

Overall, the code provides a simple and effective implementation of a spam email classifier using the Multinomial Naive Bayes algorithm. It can be easily adapted to other datasets and classification tasks by modifying the preprocessing steps and the choice of machine learning algorithm.

7. References

- Python Documentation - <https://docs.python.org/3/howto/sockets.html>
- Monkey Learn - [https://monkeylearn.com/machine-learning/#:~:text=Machine%20learning%20\(ML\)%20is%20a,to%20make%20their%20own%20predictions.](https://monkeylearn.com/machine-learning/#:~:text=Machine%20learning%20(ML)%20is%20a,to%20make%20their%20own%20predictions.)