

Optimizing Spam Filtering with Machine Learning

1.INTRODUCTION :

1.1 Overview :

- ✓ Spam filtering is the process of automatically identifying and removing unsolicited and unwanted messages, such as spam emails, text messages, or social media posts, from users' inboxes and other communication channels.
- ✓ Spam messages are a common problem that can clutter users' inboxes, waste their time, and potentially expose them to security threats, such as phishing scams, malware, or viruses. Spam filtering uses various techniques to detect and block spam messages before they reach users, improving the overall security and productivity of communication channels.

1.2 Purpose :

- ✓ **. Improving user productivity:** By reducing the amount of spam that users receive, spam filtering can help users to focus on the messages that are important to them, improving their productivity and reducing distractions

- ✓ **Reducing network traffic:** Spam messages can put a strain on network resources, as they consume bandwidth and processing power. By blocking spam messages, spam filtering can reduce network traffic and improve performance.
- ✓ **Protecting brand reputation:** Spam messages that appear to come from a legitimate brand can damage that brand's reputation. By blocking spam messages that use a brand's name or logo, spam filtering can help to protect the brand's reputation.
- ✓ **Complying with regulations:** Some industries, such as healthcare and finance, are subject to regulations that require them to protect sensitive information. Spam filtering can help these industries to comply with these regulations by preventing sensitive information from being transmitted through spam messages.

2.PROBLEM DEFINITION & DESIGN THINKING:

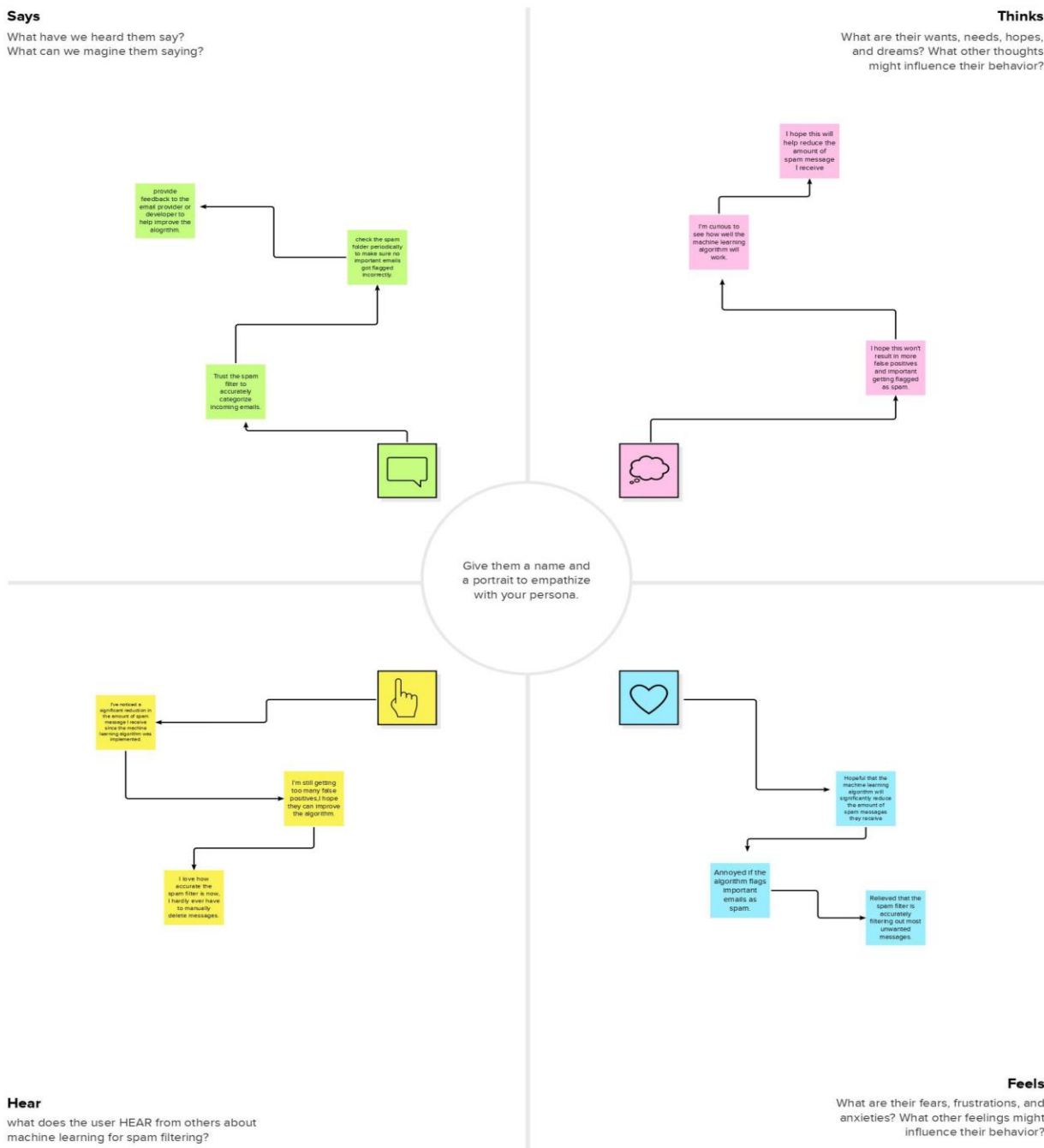
2.1 Empathy Map :

Says

What have we heard them say?
What can we imagine them saying?

Thinks

What are their wants, needs, hopes,
and dreams? What other thoughts
might influence their behavior?



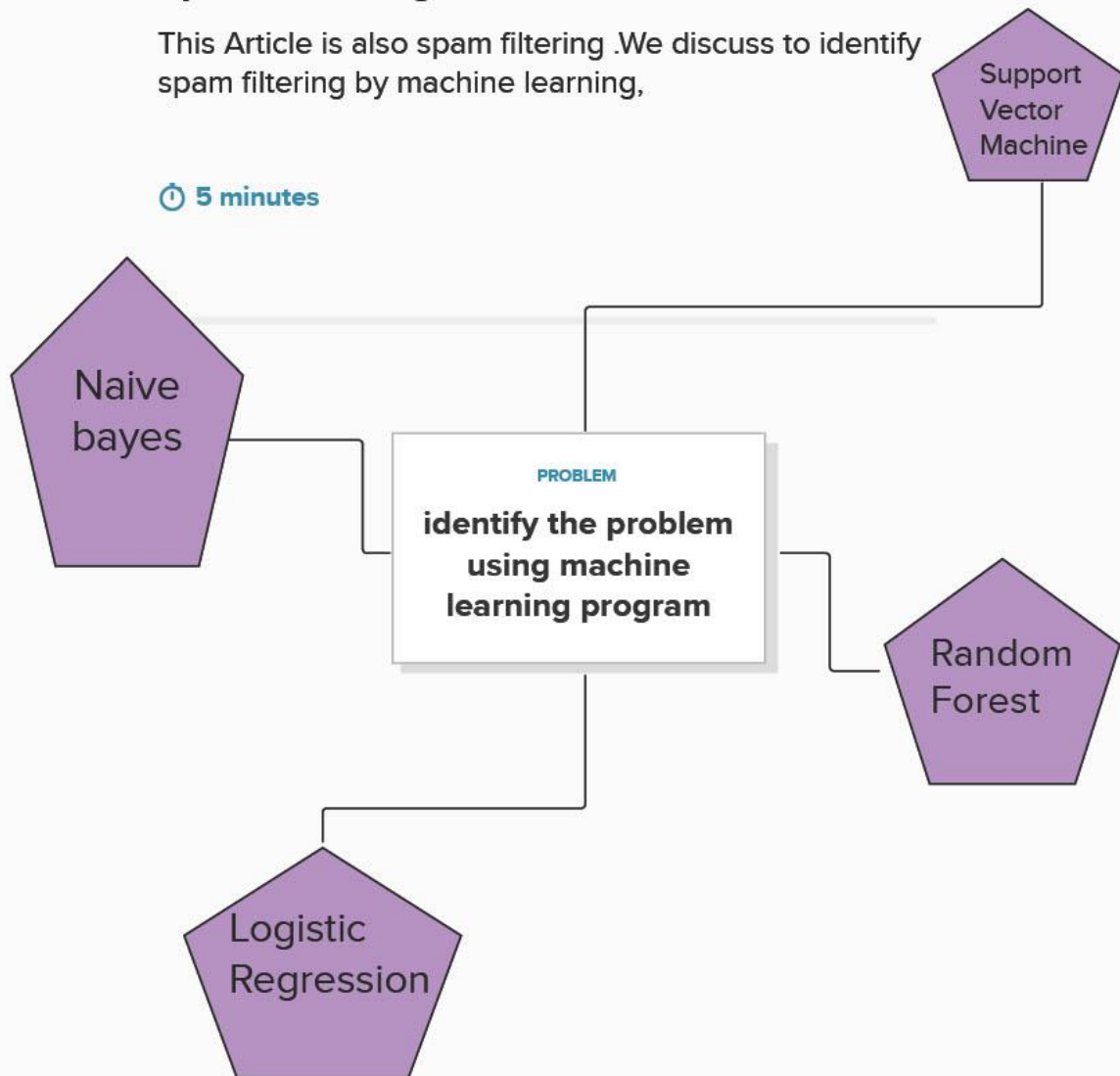
2.2 Ideation & Brainstorming Map:

1

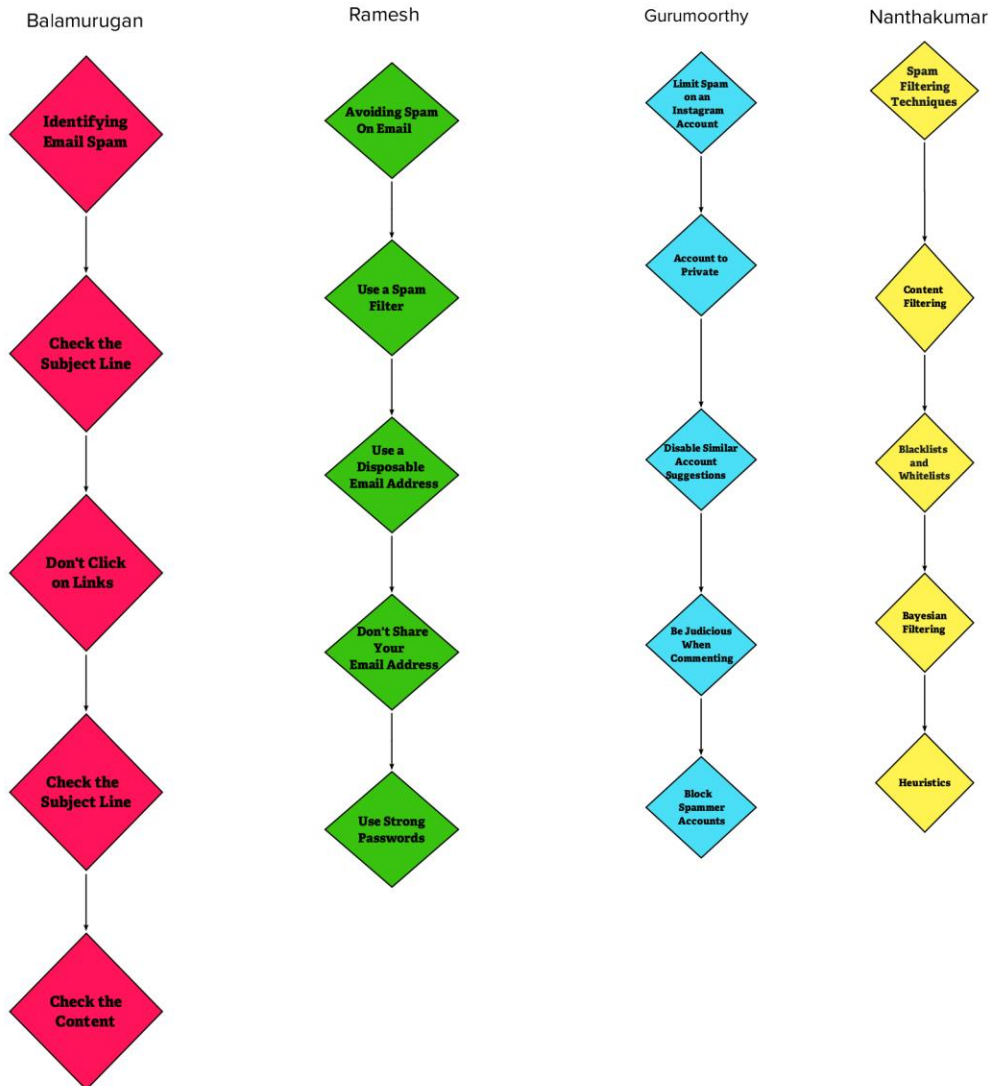
Spam Filtering▲

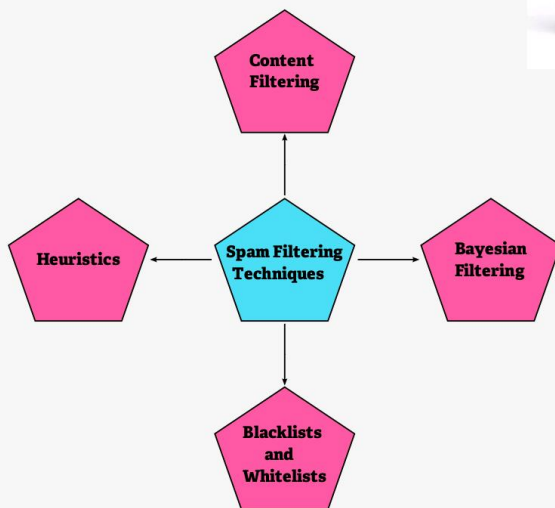
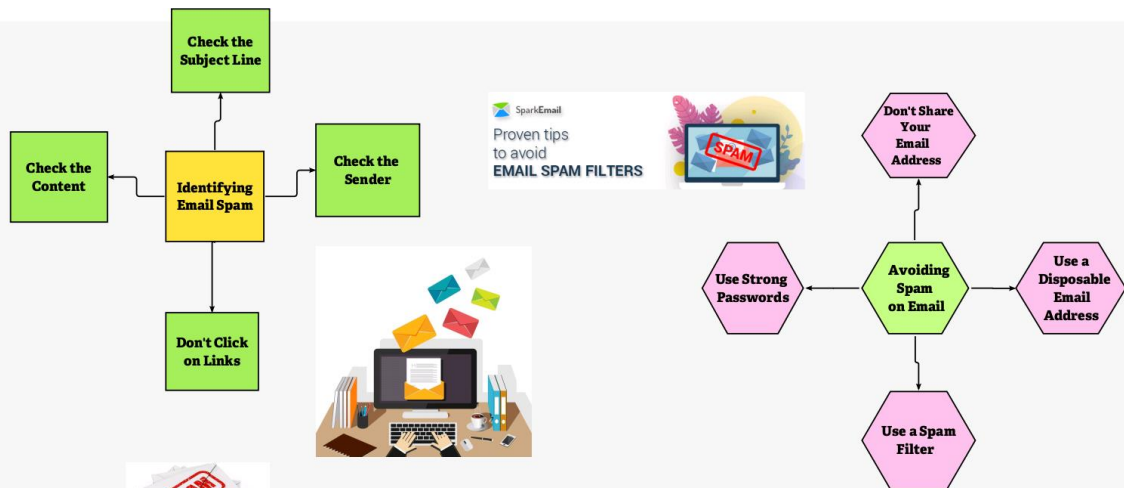
This Article is also spam filtering .We discuss to identify spam filtering by machine learning,

🕒 5 minutes



A magnifying glass with a wooden handle is positioned over a grid of random alphanumeric characters. The word "SPAM" is highlighted in red within the grid, and the magnifying glass's lens is centered on it, making it the focal point of the image.





3. RESULT:

3.1 Data Model:

Object Name And Field Name	Fields in the Object	
SMS_Message		
	Field Lable	Data type
	Id	Text
	Message_Text	Text(255)
	Message_Date_Time	DateTime
	Is_Spam	Boolean
	Is_Read	Boolean
SMS_Sender		
	Field Lable	Data type
	Id	Text
	Sender_Phone_Number	Phone
	Sender_Name	Text(255)
	Is_Blacklisted_Sender	Boolean
	Is_Known_Sender	Boolean
	Total_Message_Spam	Number
	Last_Seen_Date_Time	DateTime
	Total_Message_Sent	Number
	Total_Message_Ham	Number

3.2 Activity & Screenshot :

Collect the Dataset :

- ✓ In this project we have spam .csv data. This data is downloaded from kaggle.com

Link: <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

Importing the libraries:

```
In [1]: import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

Read the dataset :

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

Out[2]:

	v1	v2	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

Data preparation :

- ✓ Handling missing values
- ✓ Handling categorical data
- ✓ Handling Imbalance Data

Handling missing values :

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5572 entries, 0 to 5571  
Data columns (total 5 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   v1          5572 non-null   object  
1   v2          5572 non-null   object  
2   Unnamed: 2   50 non-null     object  
3   Unnamed: 3   12 non-null     object  
4   Unnamed: 4    6 non-null     object  
dtypes: object(5)  
memory usage: 217.8+ KB
```

```
In [4]: df.isna().sum()
```

```
Out[4]: v1          0  
v2          0  
Unnamed: 2   5522  
Unnamed: 3   5560  
Unnamed: 4   5566  
dtype: int64
```

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

```
In [6]: df.rename(columns = {"v1": "label", "v2": "message"}, inplace = True)
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
```

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

Out[7]:

	label	message
5567	1	This is the 2nd time we have tried 2 contact u...
5568	0	Will i_b going to esplanade fr home?
5569	0	Pity, * was in mood for that. So...any other s...
5570	0	The guy did some bitching but I acted like i'd...
5571	0	Rofl. Its true to its name

Cleaning the text data :

```
In [8]: import nltk
import re
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\Raman\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
In [9]: corpus = []
ps = PorterStemmer()
```

```
In [10]: for sms_string in list(df.message):
    message = re.sub(pattern='[^a-zA-Z]', repl=' ', string=sms_string)
    message = message.lower()
    words = message.split()
    words = [word for word in words if word not in set(stopwords.words('english'))]
    words = [ps.stem(word) for word in words]
    message = ' '.join(words)
    corpus.append(message)
```

```
In [11]: from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2500)
X = cv.fit_transform(corpus).toarray()
```

```
In [12]: y = pd.get_dummies(df['label'])
y = y.iloc[:, 1].values
```

```
In [13]: pickle.dump(cv, open('cv-transform.pkl', 'wb'))
```

Exploratory Data Analysis :

Descriptive statistical :

```
In [14]: df.describe()
```

Out[14]:

	label
count	5572.000000
mean	0.134063
std	0.340751
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

```
In [15]: df.shape
```

Out[15]: (5572, 2)

Visual analysis :

Splitting data into train and test:

```
In [19]: from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

Model Building:

Naïve Bayes model:

```
In [17]: from sklearn.naive_bayes import MultinomialNB  
classifier = MultinomialNB(alpha=0.3)  
classifier.fit(X_train, y_train)
```

```
Out[17]: MultinomialNB(alpha=0.3)
```

Performance Testing & Hyperparameter Tuning :

```
In [18]: filename = 'spam-sms-mnb-model.pkl'  
pickle.dump(classifier, open(filename, 'wb'))
```

Model Deployment :

This section has the following tasks

- ✓ Building HTML Pages
- ✓ Building server side script
- ✓ Run the web application

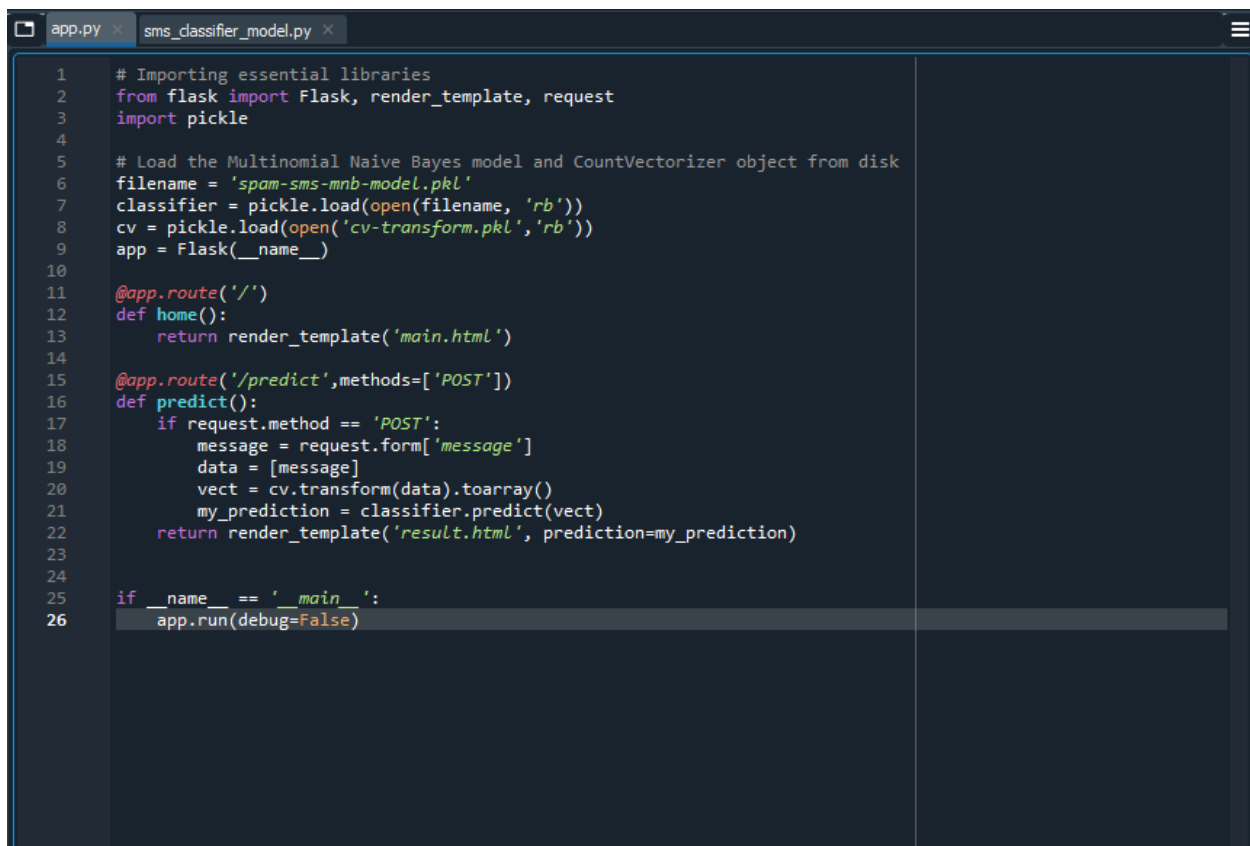
Building Html Pages:

For this project create two HTML files namely

- ✓ index.html
- ✓ spam.html
- ✓ result.html

and save them in the templates folder.

Build Python code:



```
1 # Importing essential libraries
2 from flask import Flask, render_template, request
3 import pickle
4
5 # Load the Multinomial Naive Bayes model and CountVectorizer object from disk
6 filename = 'spam-sms-mnb-model.pkl'
7 classifier = pickle.load(open(filename, 'rb'))
8 cv = pickle.load(open('cv-transform.pkl', 'rb'))
9 app = Flask(__name__)
10
11 @app.route('/')
12 def home():
13     return render_template('main.html')
14
15 @app.route('/predict', methods=['POST'])
16 def predict():
17     if request.method == 'POST':
18         message = request.form['message']
19         data = [message]
20         vect = cv.transform(data).toarray()
21         my_prediction = classifier.predict(vect)
22         return render_template('result.html', prediction=my_prediction)
23
24
25 if __name__ == '__main__':
26     app.run(debug=False)
```

```
app.py x sms_classifier_model.py x
1 # Importing essential libraries
2 import pandas as pd
3 import numpy as np
4 import pickle
5
6 # Loading the dataset
7 df = pd.read_csv("spam.csv", encoding="Latin-1")
8 # Dropping the redundant looking columns
9 df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
10 #renaming columns
11 df.rename(columns = {"v1": "label", "v2": "message"}, inplace = True)
12 df['label'] = df['label'].map({'ham': 0, 'spam': 1})
13
14 # Importing essential libraries for performing Natural Language Processing on 'SMS Spam Collection' dataset
15 import nltk
16 import re
17 nltk.download('stopwords')
18 from nltk.corpus import stopwords
19 from nltk.stem.porter import PorterStemmer
20
21 # Cleaning the messages
22 corpus = []
23 ps = PorterStemmer()
24
25 for sms_string in list(df.message):
26
27     # Cleaning special character from the message
28     message = re.sub(pattern='[^a-zA-Z]', repl=' ', string=sms_string)
29
30     # Converting the entire message into lower case
31     message = message.lower()
32
33     # Tokenizing the review by words
34     words = message.split()
35
```

```
app.py x sms_classifier_model.py* x
35 # Removing the stop words
36 words = [word for word in words if word not in set(stopwords.words('english'))]
37
38 # Stemming the words
39 words = [ps.stem(word) for word in words]
40
41 # Joining the stemmed words
42 message = ' '.join(words)
43
44 # Building a corpus of messages
45 corpus.append(message)
46
47 # Creating the Bag of Words model
48 from sklearn.feature_extraction.text import CountVectorizer
49 cv = CountVectorizer(max_features=2500)
50 X = cv.fit_transform(corpus).toarray()
51
52 # Extracting dependent variable from the dataset
53 y = pd.get_dummies(df['Label'])
54 y = y.iloc[:, 1].values
55
56 # Creating a pickle file for the CountVectorizer
57 pickle.dump(cv, open('cv-transform.pkl', 'wb'))
58
59 # Model Building
60 from sklearn.model_selection import train_test_split
61 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
62
63 # Fitting Naive Bayes to the Training set
64 from sklearn.naive_bayes import MultinomialNB
65 classifier = MultinomialNB(alpha=0.3)
66 classifier.fit(X_train, y_train)
67
68 # Creating a pickle file for the Multinomial Naive Bayes model
69 filename = 'spam-sms-mnb-model.pkl'
70 pickle.dump(classifier, open(filename, 'wb'))
```

Run the web application :

- ✓ Open anaconda prompt from the start menu
- ✓ Navigate to the folder where your python script is.
- ✓ Now type “python app.py” command
- ✓ Navigate to the localhost where you can view your web page.
- ✓ Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

```
In [1]: runfile('C:/Users/Raman/SMS-Spam-Classifler-Deployment-
master/app.py', wdir='C:/Users/Raman/SMS-Spam-Classifler-
Deployment-master')
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a
  production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Result:

Spam Detector for Short Message Service (SMS)

A Machine Learning and NLP based Web Application, Built with Flask

Enter Your Message Here...

Predict

Activate Windows
Go to Settings to activate Windows.

Spam Detector for Short Message Service (SMS)

A Machine Learning and NLP based Web Application, Built with Flask

hello world

Predict

Activate Windows
Go to Settings to activate Windows.

Spam Detector for Short Message Service (SMS)

A Machine Learning and NLP based Web Application, Built with Flask, Deployed using Heroku.

Prediction: **Great! This is NOT a Spam Message.**



Activate Windows
Go to Settings to activate Windows.

Spam Detector for Short Message Service (SMS)

A Machine Learning and NLP based Web Application, Built with Flask

congratulations you won 1000 call on this
number to get your prize

Predict

Activate Windows
Go to Settings to activate Windows.

Spam Detector for Short Message Service (SMS)

A Machine Learning and NLP based Web Application, Built with Flask, Deployed using Heroku.

Prediction: **Be Alert! This is a SPAM Message.**



Activate Windows
Go to Settings to activate Windows.

4. TRAILHEAD PROFILE PUBLIC URL:

Team Lead – <https://trailblazer.me/id/balan55>

Team Member 1 – <https://trailblazer.me/id/ramesh212>

Team Member 2 – <https://trailblazer.me/id/guru556>

Team Member 3 – <https://trailblazer.me/id/nanthu>

5. ADVANTAGES & DISADVANTAGES:

Advantages :

- ✓ **Accuracy:** Machine learning algorithms can accurately classify emails as spam or non-spam. They can analyze the content and the sender's behavior to determine if an email is likely to be spam.
- ✓ **Adaptability:** Machine learning algorithms can adapt to new spamming techniques quickly. As spammers change their tactics, the algorithm can learn to recognize these new tactics and adjust its spam filtering accordingly.
- ✓ **Efficiency:** Machine learning algorithms can analyze a large number of emails quickly, reducing the need for manual review and intervention.
- ✓ **Customization:** Machine learning algorithms can be customized to fit the specific needs of an organization. They can be trained on a specific dataset or set of rules, allowing for more precise spam filtering.

Disadvantages :

- ✓ **False positives:** Machine learning algorithms can sometimes classify legitimate emails as spam, leading to important messages being filtered out.
- ✓ **Bias:** Machine learning algorithms can be biased towards certain types of emails, leading to unequal treatment of certain senders or messages.
- ✓ **Data requirements:** Machine learning algorithms require large amounts of data to be trained effectively. Organizations may need to collect and label data for the algorithm to work optimally.
- ✓ **Complexity:** Machine learning algorithms can be complex and difficult to understand, making it challenging for non-experts to configure or troubleshoot.

6.APPLICATION:

Machine learning can be a powerful tool for optimizing spam filtering. Spam filters typically work by analyzing messages for certain characteristics that are common to spam, such as certain words, phrases, or patterns of behavior. However, spammers are constantly evolving their tactics, so traditional rule-based approaches can quickly become outdated.

By contrast, machine learning allows spam filters to adapt and learn from new data over time. Here are some steps you can take to optimize spam filtering using machine learning:

- ✓ **Collect training data:** The first step is to collect a large dataset of messages, both spam and non-spam. You can use public datasets or collect your own data.

- ✓ **Preprocess the data:** Next, you'll need to preprocess the data to extract relevant features. This might involve analyzing the text of messages to identify common spam keywords or looking at message metadata to detect patterns of behavior.
- ✓ **Train a machine learning model:** Once you have preprocessed the data, you can train a machine learning model to classify messages as spam or non-spam. There are many different types of machine learning algorithms you can use, including decision trees, logistic regression, and neural networks.
- ✓ **Evaluate the model:** After training the model, you'll need to evaluate its performance using a validation dataset. This will help you determine if the model is accurate and effective at identifying spam.
- ✓ **Optimize the model:** If the model is not performing well, you can make adjustments to the algorithm or the training data to improve its accuracy. This might involve adjusting the threshold for classifying messages as spam or non-spam or collecting additional training data to address areas of weakness.
- ✓ **Deploy the model:** Once you are satisfied with the performance of the model, you can deploy it in your spam filtering system. The model will then be able to automatically classify new messages as spam or non-spam based on its training.

7.CONCLUSION:

- ✓ In conclusion, optimizing spam filtering with machine learning is a powerful approach to identifying and filtering unwanted messages. By leveraging machine learning algorithms to analyze large datasets of messages, you can build accurate models that adapt over time to new patterns of spam behavior. To optimize spam filtering with machine learning, you need to collect and preprocess training data, train a machine learning model, evaluate and optimize its performance, and deploy it in your spam filtering system. By taking these steps, you can improve the effectiveness of your spam filtering system and ensure that your users receive only the messages they want and need.

8.FUTURE SCOPE :

The future scope of optimizing spam filtering with machine learning is vast and promising. Here are some potential areas of growth:

- ✓ **Improved accuracy:** As machine learning algorithms become more sophisticated and datasets grow larger, spam filtering systems will become more accurate at identifying and filtering unwanted messages.
- ✓ **Real-time filtering:** Real-time spam filtering will become more prevalent, enabling users to receive immediate protection from new and evolving spam threats.
- ✓ **Personalization:** Machine learning can be used to personalize spam filtering based on user preferences, enabling users to customize their filtering settings to better suit their needs.

- ✓ **Multilingual support:** As global connectivity increases, spam filtering systems will need to support multiple languages to effectively filter unwanted messages from all corners of the world.
- ✓ **Integration with other systems:** Machine learning can be integrated with other systems, such as email clients or messaging apps, to provide seamless spam filtering capabilities.