CSCE 5290.003 (13149)

Natural Language Processing

Spam Email Detection with Text Classification and Sentiment Analysis

Using Machine Learning Random Forest

Project Proposal

SPAM Email Detection with Sentiment Analysis

Project Proposal

1. Project Title and Team Members

Project Group **- 5**

Project Title **- Spam Email Detection with Sentiment Analysis**

Team Members:

- **Danda Reethika Reddy - 11608030**
- **Surya Sai Raj Lakkoju - 11610081**
- **Ashesh Pinisetti - 11649062**
- **Mohammad Khaja Moinuddin - 11603687**

2. Goals and Objectives:

- **Motivation:**
  The reason behind getting motivated about this project is that in the current existing machine learning models we see there are models that can detect whether the emails are spam or not, but not about categorizing which type of email that content actually defines. Here we analyze how the spam emails were detected using the Text classification techniques and then we will be implementing the sentimental analysis techniques to find out what kind of email it is for example: Threaten, Marketing, Phishing or etc. Here we will be using a random forest Machine learning model which will be trained with the dataset to analyze the accuracy.

- **Significance:**
  The main significance of our project is to identify and classify the spam emails. Identifying the spam emails will help users in not getting involved in any type of fraud. The users will have a basic understanding of what the email is about if it is categorized as marketing, anti-virus, phishing, or money scam spam.

  Importance for spam mails to be filtered is, as it can contain malicious content that can spread viruses and cyber attacks.To overcome these types of attacks, spam mails filtering is necessary.

After identification of spam emails,Based on the content present in the mail we are categorizing them into groups by using text recognition and sentiment analysis.

- **Objective:**
  The primary objective of spam email detection with sentiment analysis is to identify and filter out unwanted emails from the inbox of an email account. This involves using machine learning algorithms to automatically detect and categorize emails as spam or not.

  Sentiment analysis, on the other hand, involves analyzing the language used in a piece of text to determine the underlying emotional tone or sentiment. In the context of email filtering, sentiment analysis can be used to determine whether an email is trying to manipulate or deceive the recipient, such as through the use of overly positive or negative language.

  By combining spam email detection with sentiment analysis, it is possible to create a more effective email filtering system that can identify and block unwanted emails that may be attempting to manipulate or deceive the recipient. This can help improve the user's email experience by reducing the amount of unwanted or malicious emails that they receive, and also enhance their security and privacy online

- **Features:**
  Here are some common features we are using in spam email detection:

  Content Analysis: Spam emails often contain certain patterns or phrases that can be identified and used as indicators of spam. These include things like suspicious keywords, unusual formatting, and overuse of capital letters or exclamation points.

  Blacklists and Whitelists: These are lists of known good and bad email addresses, domains, and IP addresses. Emails from whitelisted sources are allowed to pass through, while those from blacklisted sources are blocked or flagged as spam.

  Reputation Analysis: The reputation of an email sender can be evaluated based on factors such as how often their emails are marked as spam, the types of content they send, and the quality of their email infrastructure.

  URL Analysis: Links in spam emails often lead to fraudulent or malicious websites. URL analysis can be used to identify these links and block or flag emails that contain them.

3 Related work :

3.1 Sentiment analysis using unlabeled dataset.

Sentiment analysis is the process of classifying whether a block of text is positive, negative,or,neutral.Usually,sentiment analysis view it as text classification problem but it needs labeled data to train the model.But most of the times we will get the unlabeled data,which is to be done manually.So every time to label the data manually takes time and it is hard.In this the author found the best techniques to label the dataset automatically by avoiding manual labeling.In this lexicon labeling and k-mean labeling are compared and lexicon labeling gave the better results.They used TF-IDF for feature extraction, to train Naïve Bayes and Support vector machine (SVM) classifiers.

3.2 A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms

Email spam classification is a well-studied problem in the field of natural language processing (NLP). In the paper[2], the authors provide a comprehensive review of the existing work in email spam classification using machine learning algorithms. The authors compare the strengths and weaknesses of various machine learning algorithms and discuss the features that can be extracted from emails for classification. The article identifies some of the challenges in email spam classification, such as the imbalance between spam and non-spam emails, and the need for feature selection to avoid overfitting. To address these challenges, some researchers have proposed hybrid approaches that combine multiple machine learning algorithms or use feature selection techniques like Principal Component Analysis (PCA). They have primarily focused on analyzing the content and metadata of emails for classification, while sentiment analysis can help identify the sentiment of the email content and potentially improve classification accuracy.

3.3 An email classification model based on rough set theory

In this paper[3], the authors presented a novel email classification model based on rough set theory. They proposed a three-stage process for email classification, which involves data preprocessing, feature extraction, and classification. In the feature extraction, they used rough set theory to identify the most relevant features for email classification. This involves reducing the feature space by identifying the essential attributes that are most discriminatory for the classification task. Finally, in the classification stage, the authors used a decision tree algorithm to classify the email data into spam or non-spam

categories. The key contribution of the paper is the application of rough set theory to the problem of email classification, which allows for the identification of the most relevant features for classification, thus reducing the dimensionality of the feature space and improving classification accuracy. The paper demonstrates the effectiveness of their proposed model in email classification and highlights the potential of rough set theory as a useful tool in the field of email classification.

3.4 Sentiment Analysis for Fake News Detection by Means of Neural Networks

The authors proposed a two-stage approach for fake news detection, which involves sentiment analysis and classification using neural networks. In the first stage, they performed sentiment analysis on the news articles to extract sentiment-related features such as positive or negative sentiment, emotion, and opinion. In the second stage, they used a neural network model to classify the news articles as fake or real based on the sentiment-related features. By incorporating sentiment-related features into the classification process, the authors had aimed to improve the accuracy of fake news detection. The experimental results show that the proposed approach achieves high accuracy in detecting fake news, outperforming some of the existing methods. The article presents a promising approach to fake news detection that combines sentiment analysis and neural networks.

3.5 Spam email classification and sentiment analysis based on semantic similarity methods

Now-a-days electronic email is widely used for communication purposes.So classifying the spam and non-spam email is necessary for security purposes.There are many NLP techniques to do this but these cannot handle unbalanced classes and have lower efficiency due to irrelevant feature extraction. So in paper[4] the author used sentiment analysis- based semantic FE and hybrid FS techniques to increase the efficiency. In this technique, it will measure the polarity of the text and then the spam detection will be done.The TF-IDF, Information gain and gini-index is used.This analysis is showing that gini index and SVM classifier shows the higher performance of 95.17%.

3.6 Novel email spam detection method using sentiment analysis and personality recognization.

Different techniques were developed for detecting spam emails in recent days.In this paper[6] the author focused on specific machine learning algorithms-content based

filters.These filters will understand the content of the emails to classify them as spam or non-spam.Filters include such as heuristic filtering,learning-based filtering and filtering by compression.The author compared different techniques like rule-based system,IP blacklist,Heuristic-based filters , bayesian-network based filters white list and DNS black-holes.After comparison concluded that bayesian-network based filters are most effective and accurate.Author demonstrated that even there are many techniques implemented ,bayesian methods of text classification are still useful.

3.7 Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism.

The fast growth of Internet technology has drastically altered the experience of online users, while security concerns are becoming increasingly overpowering. In the current state, new threats have the potential to seriously harm consumers systems as well as steal their money and personal information (PII). Among these concerns, phishing stands out as a prominent criminal activity that employs social engineering and technology to steal a victim's identification data and account information. According to an Anti-Phishing Working Group (APWG) report, the number of phishing attacks have increased significantly by 46% during the first quarters of 2018 when compared to the final quarters of 2017. To identify these attacks many MNC companies came up with a solution of finding such phishing emails and providing one layer of security by eliminating them to steal the customers PII using Deep Learning models. In this there are different DL models used in finding such phishing or spam or cyber attacking emails. In these models some of the highly used models are RCNN, BI-LSTM etc. These models extract the features from the emails and in-turn learn themselves by understanding the context of what kind of emails these are based on tagging those emails as SPAM/JUNK.

3.8 Efficient spam and phishing emails filtering based on deep learning

The number of emails is increasing quickly because they are the primary, quick, and affordable form of communication in all industries. As a result, the demand for more precise spam filters has increased. To have a reliable and secure email filter, it is essential to be able to quickly identify spam emails. A specific type of social network attacking technique is phishing. Phishing aims to deceive victims by providing sensitive information, including identification and financial details. Emails with malicious attachments or redirected URLs are frequent forms of phishing attacks. To avoid such types of cyber-attacks, Google, Yahoo and Microsoft some of the famous internet email service providers have implemented some of the deep learning techniques where these

neural networks extract the features and train themselves using the training data and run the same simulations on the testing data. These types of models are called supervised models. There are different types of supervised and unsupervised models.

3.9 Sentiment Analysis for Fake News Detection by Means of Neural Networks

Public opinion research can offer us vital information. Many uses may be found for the analysis of sentiment on social media platforms like Twitter and Facebook, which has grown to be a potent tool for discovering what consumers think. Yet, the difficulties in natural language processing are impeding sentiment analysis's effectiveness and accuracy. Last, a few years, deep learning models have improved a lot in answering the problems related to NLP. In this publication we are discussing the most recent research using deep learning to address issues with sentiment analysis, like sentiment polarity. Models that use term frequency-inverse document frequency (TF-IDF) and word embedding techniques have been applied on a series of datasets. Here the sentiment analysis is mainly focused on the words which are tokenized. The tokenized words are finally segregated into two different categories: positive and negative. In positive categories the words which are not related to violence, threatening kind of words. Based on these categories, the sentiment analysis will be done on the content passed to the model. Automatic sentiment analysis (SA) is an issue that is becoming a more popular research area. Despite the importance of SA and the variety of applications that it currently has, there are several difficulties with natural language processing that must be overcome. Using deep learning models like CNN, RNN, and DNN to automatically classify the context.

4. Dataset

The dataset contains 48 features extracted from 10,000 web pages, consisting of 5,000 phishing webpages and 5,000 legitimate webpages. The webpages were downloaded from January to May 2015 and from May to June 2017.

The dataset's primary purpose is to be used for machine learning-based phishing detection, where the features can be used to train and evaluate machine learning models that can classify a given webpage as either a phishing webpage or a legitimate web page.

Each of the 10,000 web pages in the dataset is labeled as either a phishing webpage or a legitimate webpage. This means that the dataset is a supervised learning dataset, where the machine learning model can learn from the labeled examples to make predictions on new, unlabeled web pages.

Overall, this dataset is a valuable resource for anyone interested in developing and

evaluating machine learning models for phishing detection, as it provides a large set of features extracted from webpages and labels indicating whether each webpage is a phishing or legitimate webpage.

Here are some high-level description about the features in the dataset :

- NumDots: The number of dots (.) in the URL.
- SubdomainLevel: The number of levels in the subdomain.
- PathLevel: The number of levels in the URL path.
- UrlLength: The length of the URL.
- AtSymbol: Whether the URL contains an @ symbol.
- RightClickDisabled: Whether right-clicking is disabled on the webpage.
- PopUpWindow: Whether the webpage contains a pop-up window.
- SubmitInfoToEmail: Whether the webpage submits information to an email address.
- IframeOrFrame: Whether the webpage contains an iframe or frame.
- MissingTitle: Whether the webpage is missing a title.
- ImagesOnlyInForm: Whether the webpage contains only images in a form.
- SubdomainLevelRT: The number of levels in the subdomain in the redirected URL.
- UrlLengthRT: The length of the redirected URL.
- PctExtResourceUrlsRT: The percentage of external resources in the redirected URL.
- ExtMetaScriptLinkRT: Whether the redirected URL contains external meta, script or link tags.
- CLASS_LABEL: A binary label indicating whether the URL is a phishing email or not.

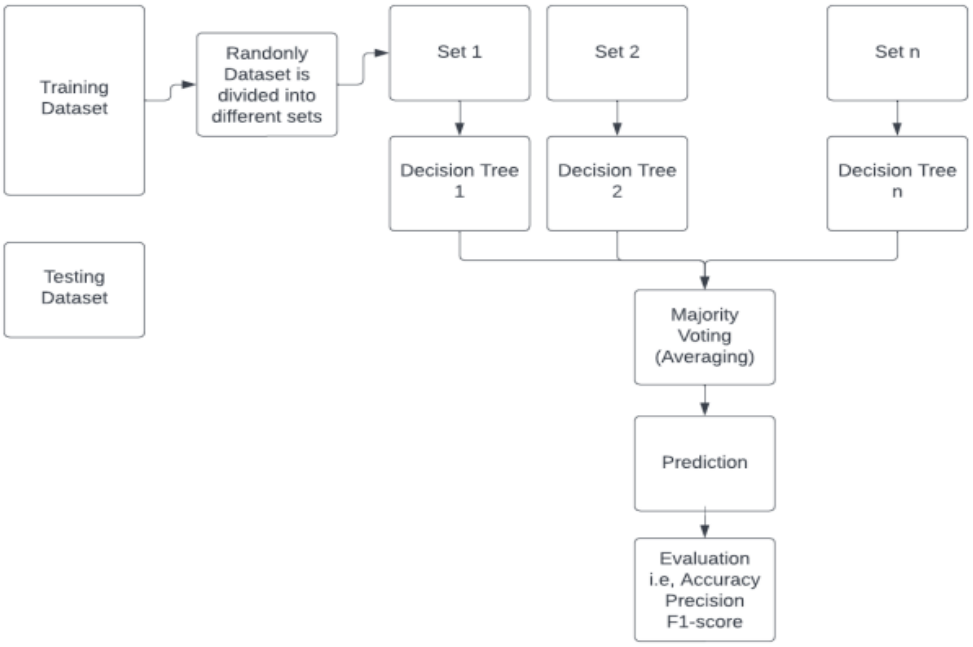
Sample view of our dataset is :

id	NumDots	Subdomai	PathLevel	UrlLength	NumDash	NumDashl	AtSymbol	TildeSymb	NumUnde	NumPerce	NumQuer	NumAmpe	NumHash	NumNume	NoHttps	RandomSt	IpAddress	DomainIn
1	3	1	5	72	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	3	1	3	144	0	0	0	0	2	0	2	1	0	41	1	0	0	0
3	3	1	2	58	0	0	0	0	0	0	0	0	0	0	1	0	0	0
4	3	1	6	79	1	0	0	0	0	0	0	0	0	0	1	0	0	0
5	3	0	4	46	0	0	0	0	0	0	0	0	0	2	1	1	0	0
6	3	1	1	42	1	0	0	0	0	0	0	0	0	0	1	0	0	0
7	2	0	5	60	0	0	0	0	0	0	0	0	0	1	1	0	0	0
8	1	0	3	30	0	0	0	0	0	0	0	0	0	3	1	1	0	0
9	8	7	2	76	1	1	0	0	0	0	0	0	0	2	1	1	0	1
10	2	0	2	46	0	0	0	0	0	0	0	0	0	0	1	0	0	0
11	5	4	2	64	1	1	0	0	0	0	0	0	0	3	1	1	0	1
12	2	0	2	47	0	0	0	0	0	0	0	0	0	1	1	0	0	0
13	2	1	2	61	1	1	0	0	0	0	0	0	0	0	0	1	0	0
14	2	1	3	35	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	2	1	2	60	1	1	0	0	0	0	0	0	0	0	1	1	0	0
16	3	0	4	73	0	0	0	0	0	0	0	0	0	0	1	0	0	0
17	3	0	5	50	0	0	0	1	0	0	0	0	0	10	1	1	1	0
18	3	1	2	59	1	1	0	0	0	0	0	0	0	7	1	1	0	0
19	2	0	3	28	0	0	0	0	0	0	0	0	0	1	1	0	0	0
20	1	0	4	59	0	0	0	0	0	0	0	0	0	22	1	1	0	0
21	1	0	4	32	0	0	0	0	0	0	0	0	0	4	1	1	0	0
22	5	1	2	52	0	0	0	0	0	0	0	0	0	1	1	1	0	0
23	2	1	6	62	1	0	0	0	0	0	0	0	0	0	1	1	0	0
24	1	0	10	105	2	0	0	0	0	0	0	0	0	24	1	1	0	0
25	4	1	2	55	0	0	0	0	0	0	0	0	0	0	1	1	0	0
26	5	0	3	134	3	0	0	0	0	0	0	0	0	1	1	1	0	0
27	2	0	3	43	0	0	0	0	0	0	0	0	0	0	1	0	0	0
28	6	0	3	210	2	0	0	0	7	0	5	4	0	24	1	1	0	0

5. Features and implementation :

In order to train our model we used the random forest algorithm.

Random forest Algorithm :



Random Forest is a supervised machine learning algorithm which can be used for both the Classification and Regression problems. This produces good results even without hyper-parameter tuning. In this Algorithm, the training dataset is divided randomly into n different datasets i.e, set 1, set 2,....., set n. For each set, a decision tree is constructed. Then majority voting i.e, average of the results of all decision trees is done. Then, the predictions are done. Then the output is evaluated using measures like Accuracy, Precision and F1-score.

Pseudocode:

rfc = RandomForestClassifier(criterion='entropy', max_features='sqrt', max_samples=0.5, min_samples_split=80) rfc.fit(X_train, y_train) y_pred = rfc.predict(X_val) print(classification_report(y_val, y_pred, zero_division=0))

The hyperparameters used here are:

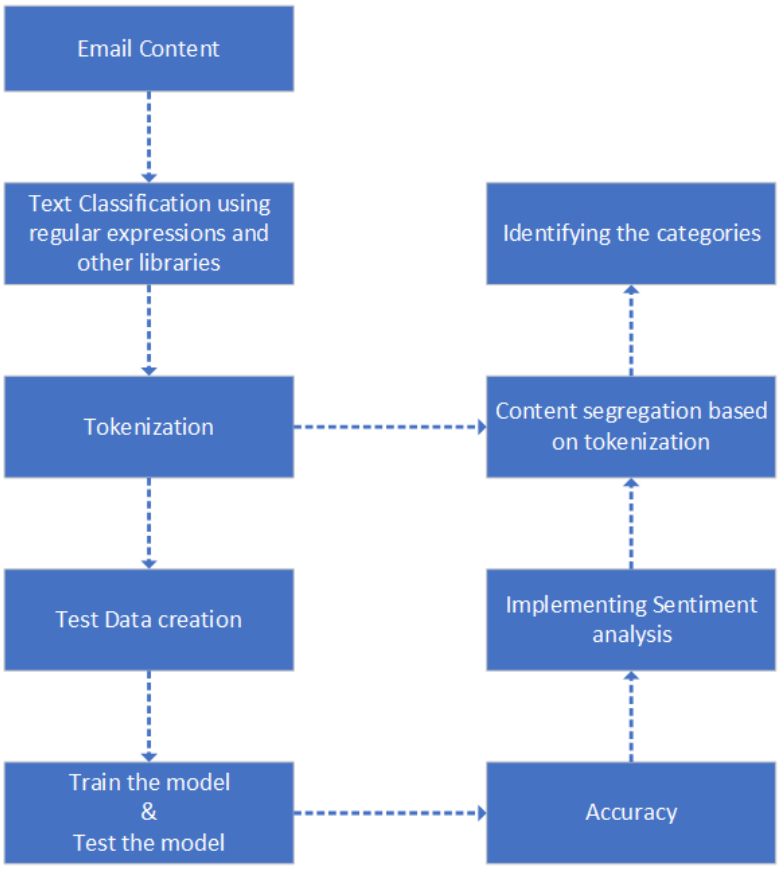
● criterion - is set to ‘entropy’ to calculate information gain, which determines the effectiveness of split.

● max_featues - this decides the maximum number of features that can be considered to split. We cannot assign all the features because it will not give an efficient decision tree. So, here we set it to sqrt i.e, square root of total number of features.

● max_samples - this defines what percentage of dataset to be included in an individual tree.

● min_samples_split - this determines the minimum number of observations to be considered for each split.

Detail design of features :



The above diagram shows the detailed workflow of the whole Project.

Step 1 : Email content i.e The message present in the email is extracted and then text classification is done based on the regular expressions and some other libraries.

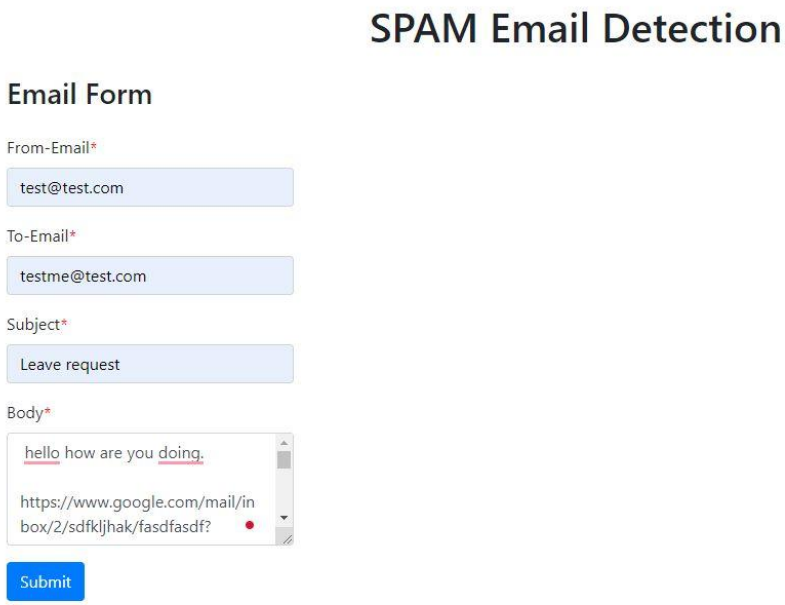
Step 2 : The classified data is sent into the tokenization process. This process is done by using the beautify method.

```
def beautify(text):
    import nltk
    nltk.download('stopwords')
    from nltk.corpus import stopwords
    token_list = {}

    tokens = [t for t in text.split()]
    clean_tokens = tokens[:]
    sr = stopwords.words('english')
    for token in tokens:
        if token in stopwords.words('english'):
            clean_tokens.remove(token)

    freq = nltk.FreqDist(clean_tokens)

    for key, val in freq.items():
        token_list[key] = val

    return token_list
```

Step 3 : After the tokenization,pattern validation is done by using below patterns.

```
emailpattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
nohttps = r'/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/'
urlpattern = r'^https?://[\w.-]+(?:\.[\w]+)+[\w\-._~:/?#[\]@!$&\'()*+,;=]*$'
nohttpspattern = r'^http://[\w.-]+(?:\.[\w]+)+[\w\-._~:/?#[\]@!$&\'()*+,;=]*$'
httpspattern = r'^https://[\w.-]+(?:\.[\w]+)+[\w\-._~:/?#[\]@!$&\'()*+,;=]*$'
dotpattern = r'\.'
ippattern = r'^(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.(?:25[0-5]|2[0-4
dashurlpattern = r'/\(\([a-z_-]*\.[a-z_-]*\)\|\([a-z_]*\.[a-z_-]*\.[a-z_-]*\)\)/'
numpattern = r'[0-9]'
stringpattern = r'[a-zA-z]'
sensitivepattern = r'(.)\1{9,}'
hostnamepattern = r'^https?://([\w.-]+)'
hostpathpattern = r'^https?://[\w.-]+(/.*)'
querypattern = r'(?<=\?)\S+'    # r'^https?://[\w.-]+(/.*)\?(\S+)'
subdomainpattern = r'(?<=://)([\w.-]+)\.'
```

Step 4 : For the frontend data that we are passing initially, a test dataset is created in the backend while validating the patterns.



Spam detected :

```
[7, 2, 0, 178, 2, 0, 2, 5, 0, 9, 0, 7, 6, 4, 1, 27, 0, 0, 0, 1, 28, 135, 100, 2, 0, 0, 0.909090909, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, -1, 1, 1, 0, 1]]
```

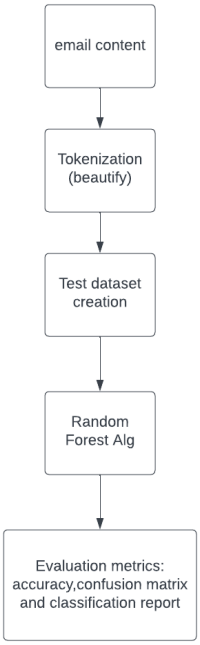
Not spam :

```
[[6, 2, 0, 178, 2, 0, 2, 0, 0, 0, 0, 7, 0, 4, 1, 13, 0, 0, 0, 1, 28, 135, 100, 2, 0, 0, 0.909090909, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 1, 1, 0, 1]]
```

Step 5 : This testing dataset is passed to our model (random forest). As our training model is trained it will predict whether the email is spam or not.

```
    dataset = pd.read_csv(filepath, names=header_names)
    dataset.head()
    x = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, 48].values

    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
    classifier = RandomForestClassifier(n_estimators=100)
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    # print(y_pred)
    result = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:")
    print(result)
    result1 = classification_report(y_test, y_pred)
    print("Classification Report:", )
    print(result1)
    result2 = accuracy_score(y_test, y_pred)
    print("Accuracy:", result2)

    x_test = data_values
    y_pred = classifier.predict(x_test)
    print(y_pred)
    return jsonify({'data': "Spam Detected" if y_pred[0] == 1 else "Not Spam"})
except Exception as err:
    return jsonify({'data': err})
```

Step 6 : After passing the dataset to the model, it will predict and give us the accuracy results.Here we used confusion matrix,classification report and accuracy_score as our evaluation metrics.



Evaluation Methods:

1. A confusion matrix is a table that summarizes the performance of a machine learning model on a classification task. It shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) predictions made by the model.

   Here is an example of a confusion matrix:

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

   True Positives (TP): The number of positive instances correctly predicted by the model.
   False Positives (FP): The number of negative instances incorrectly predicted as positive by the model.
   True Negatives (TN): The number of negative instances correctly predicted by the model.
   False Negatives (FN): The number of positive instances incorrectly predicted as negative by the model.

2. **Classification Report:** A classification report is a summary of the performance of a classification model on a set of test data. It typically includes metrics such as precision, recall, F1 score, and support for each class label. A classification report provides a detailed evaluation of the model's performance and can help identify areas where the model is doing well and areas where it needs improvement.

3. **Accuracy:** This is the most straightforward evaluation method that measures how many predictions made by the model are correct. It is calculated by dividing the number of correct predictions by the total number of predictions.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

4. **Precision and Recall:** These evaluation metrics are used to evaluate the performance of classification tasks. Precision measures how many of the positive predictions made by the model are correct, while recall measures how many of the actual positives in the dataset the model has identified correctly.

$$\text{Precision} = TP / (TP + FP)$$

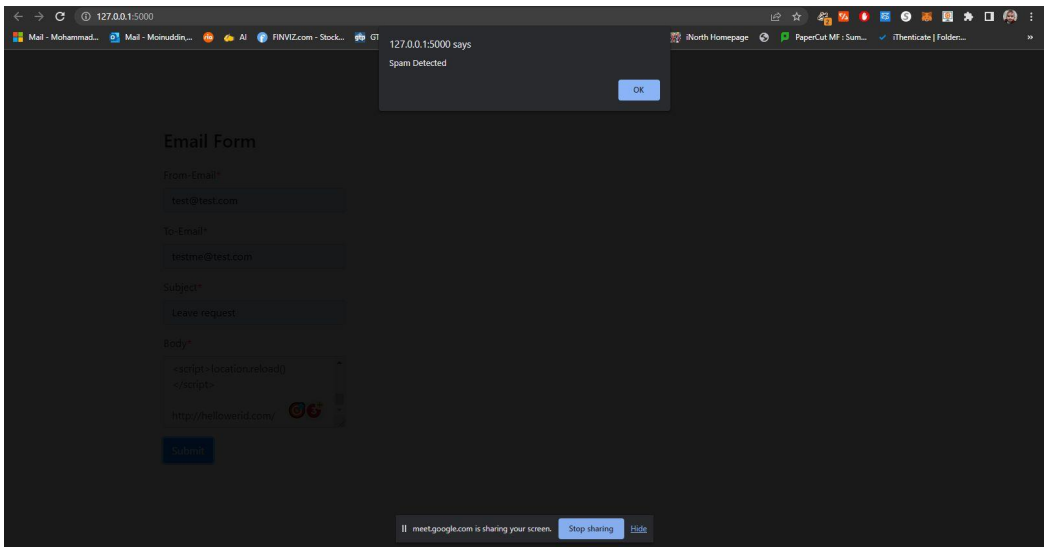
$$\text{Recall} = TP / (TP + FN)$$

5. **F1 Score:** The F1 score is the harmonic mean of precision and recall. It is a commonly used evaluation metric for classification tasks that balances precision and recall.

$$\text{F1 Score} = 2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}))$$

6. Analysis and preliminary results:

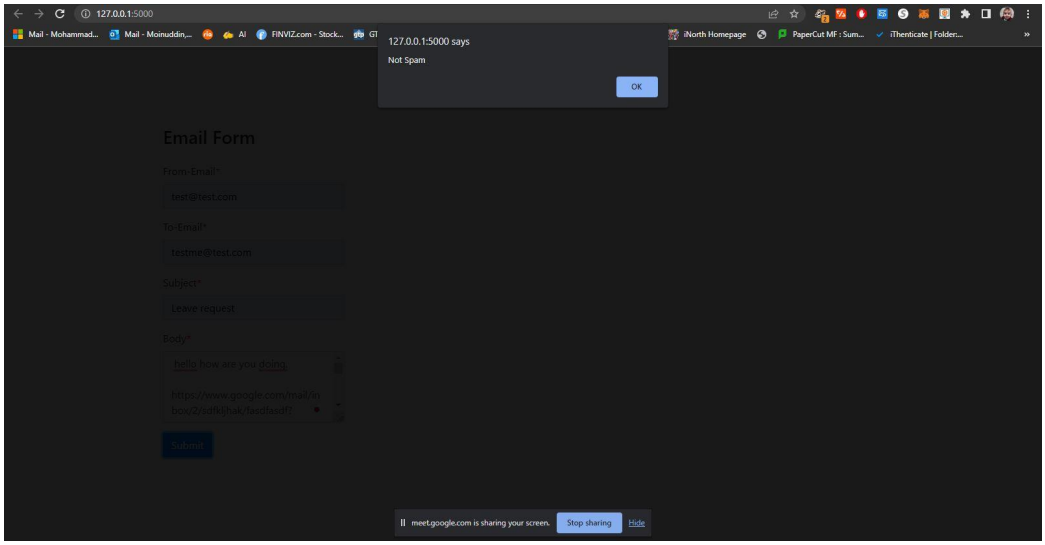
Spam detected :

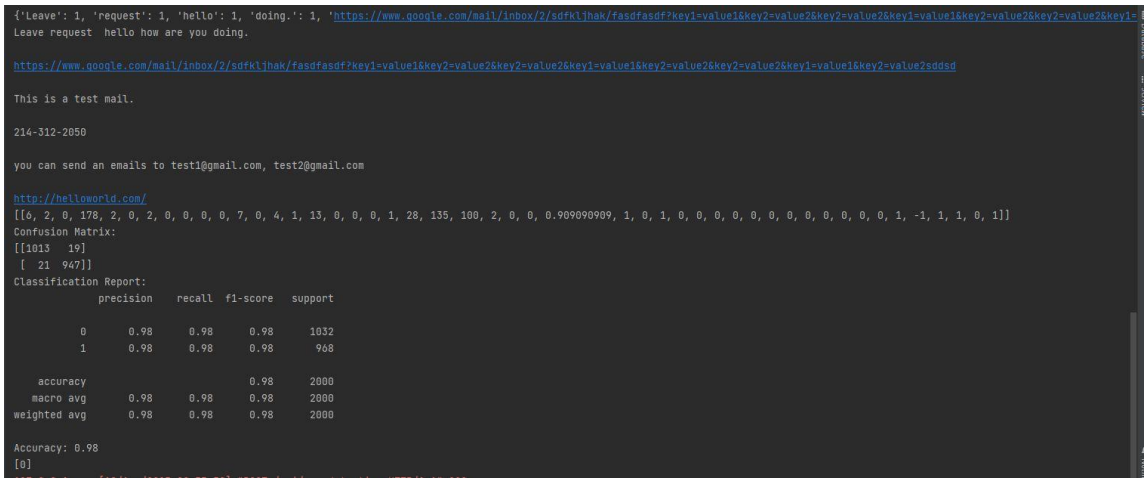


```
{'Leave': 1, 'request': 1, 'hello': 1, 'doing.': 1, 'https://www.google.com/mail/inbox/2/sdfkljhak/fasdfasdf?key1=value1&key2=value2&key2=value2&key1=value1&key2=value2&key2=value2&key1
Leave request  hello how are you doing.

https://www.google.com/mail/inbox/2/sdfkljhak/fasdfasdf?key1=value1&key2=value2&key2=value2&key1=value1&key2=value2&key2=value2&key1=value1&key2=value2sddsd

^^^^^^ ###### $$$$$$$ %%%%%%%%%

sdafsadfasdfsadf
This is a test mail.
sdf dfasdfsadfasdf sadfs sdafsadfsadf sadfsda fsad fsdaf asfsadf sda fsdaf sdfsdfasdfsafrwerwecbgshth Frtr456t
214-312-2050

you can send an emails to test@gmail.com, test1@gmail.com

<script>location.reload()</script>

http://hellowerld.com/
```

```
[[7, 2, 0, 178, 2, 0, 2, 5, 0, 9, 0, 7, 6, 4, 1, 27, 0, 0, 0, 1, 28, 135, 100, 2, 0, 0, 0.909090909, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, -1, 1, 1, 0, 1]]
Confusion Matrix:
[[985  26]
 [ 16 973]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1011
           1       0.97      0.98      0.98       989

    accuracy                           0.98      2000
   macro avg       0.98      0.98      0.98      2000
weighted avg       0.98      0.98      0.98      2000

Accuracy: 0.979
```

Not spam :





7. References:

Dataset : https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning?select=Phishing_Legitimate_full.csv

[3.1] https://ieeexplore.ieee.org/abstract/document/9004372/

[3.2] https://ieeexplore-ieee-org.libproxy.library.unt.edu/document/9334020

[3.3] https://ieeexplore.ieee.org/abstract/document/1505383

[3.4] https://link.springer.com/chapter/10.1007/978-3-030-50423-6_49

[3.5]https://www.inderscienceonline.com/doi/abs/10.1504/IJCSE.2023.129147?journalCode=ijcse

[3.6]https://academic.oup.com/jigpal/article-abstract/28/1/83/5680435

[3.7]https://ieeexplore.ieee.org/abstract/document/8701426

[3.8]https://www.sciencedirect.com/science/article/abs/pii/S1389128622000469

[3.9]https://link.springer.com/chapter/10.1007/978-3-030-50423-6_49

8. Git Repository URL: https://github.com/surya5a72/csce5290_NLP

Google drive link (vedio) :
https://drive.google.com/file/d/1R_dP8b0tzhHJLTS4ys-JWgNz3SvDD7oo/view?usp=sharing