# Tech Saksham

## Capstone Project Report

## "Detecting Spam Emails"

### "The Kavery Engineering College"

| NM ID | NAME |
|---|---|
| aut2261270036 | Manoj kumar P |

Trainer Name

Ramar Bose

Sr. AI Master Trainer

## ABSTRACT

The ubiquity of email communication has brought forth the persistent challenge of spam messages, posing threats to user privacy and security. This project endeavors to develop a proficient system for the detection of spam emails through the utilization of machine learning algorithms. The proposed system will leverage a diverse array of features such as sender analysis, content inspection, and linguistic patterns to discern between legitimate correspondence and spam. By harnessing a dataset comprising labeled emails, we will train and evaluate several machine learning models, including Support Vector Machines, Naive Bayes, and Random Forests. The project's primary objective is to construct a robust spam detection mechanism that can bolster email security, mitigate the risks associated with phishing attacks, and augment user experience. The outcomes of this endeavor are poised to make significant contributions towards combatting the proliferation of spam within the digital landscape.

# INDEX

# CHAPTER 1

# INTRODUCTION

## 1.1 Problem Statement

Spam emails continue to be a persistent issue, cluttering inboxes and potentially leading to security risks for individuals and organizations. The project aims to develop a robust system capable of automatically detecting and filtering out spam emails, thereby enhancing user experience and security in email communication..

## 1.2 Proposed Solution

Spam emails pose a significant challenge in modern communication, often inundating inboxes with unwanted content and potential security risks. To combat this issue, the project proposes a multi-faceted approach to accurately detect and filter out spam emails using machine learning techniques.

## 1.3 Feature

**Text Analysis Features:**

- **Word Frequency:** Count the frequency of each word in the email.
- **TF-IDF (Term Frequency-Inverse Document Frequency):** Weigh the importance of words in the email.

**Metadata Features:**

- **Sender Information:** Analyze the sender's email address and domain reputation.
- **Email Headers:** Extract information from email headers, such as date, subject, and source IP.
- **Attachments:** Flag emails with attachments, as they are common in spam messages.

**Attachment Analysis Features:**

- **File Type:** Identify the types of files attached (e.g., executable files often found in malware).
- **File Size:** Flag unusually large attachments, common in spam emails with malware.
- **Attachment Names:** Check for suspicious or random attachment names.

**Machine Learning Derived Features:**

- **Probability Scores:** Output probability scores from machine learning models indicating the likelihood of an email being spam.
- **Cluster Membership:** Assign emails to clusters based on similarity, identifying potential spam clusters.

## 1.4 Advantages

**Improved Email Filtering:**

- The system provides a more effective way to filter out unwanted spam emails, ensuring that users' inboxes remain clutter-free and focused on important messages

**Time Savings:**

- Users save time that would otherwise be spent manually sifting through spam emails, allowing them to focus on critical tasks and communication

**Real-time Detection:**

- Emails are analyzed in real-time, providing immediate feedback on whether an email is likely to be spam or legitimate

## 1.5 Scope

**Emotion monitoring:** Extend your spam detection model to analyze the emotional tone of emails. Sentiment analysis can help identify emotionally charged content, which may be relevant for user experience or targeted marketing.

**Therapeutic applications** Investigate how your spam detection model can be adapted for therapeutic purposes. For example:

**Mental Health Support**: Use email content to identify signs of distress or mental health issues in users. Provide appropriate resources or alerts.

**Wellness Reminders**: Send personalized wellness reminders or motivational messages based on email content.

**Enhanced Security**:Collaborate with cybersecurity experts to enhance email security beyond spam detection:

- ❑ **Phishing Detection**: Extend your model to identify phishing attempts or suspicious links.
- ❑ **Malware Detection**: Analyze attachments for potential malware.

# CHAPTER 2

# SERVICES AND TOOLS REQUIRED

## 2.1 Services Used

**Data Collection:**

- Various sources for collecting labeled email datasets, including public repositories, online resources, or proprietary datasets if available

**Git/GitHub:**

- Version control using Git for managing code changes and collaborating on the project, with GitHub as the remote repository

**Machine Learning Algorithms:**

- Naive Bayes, Support Vector Machines (SVM),(LR) and potentially neural network-based classifiers are used for training the spam detection model.

## 2.2 Tools and Software used

**Tools**:

**Python:**

- Primary programming language for implementing machine learning models, data preprocessing, and system development.

**Google Cloud Platform (GCP) (Optional):**

- Another cloud platform option for deploying the system, providing cloud computing services and storage

**Software Requirements**:

**Python 3.x:**

- The project requires Python 3.x to be installed on the system for implementing the machine learning models, data preprocessing, and system development.

**Pandas (1.3.3):**

- Version 1.3.3 or compatible version of Pandas library is necessary for data manipulation and analysis, used for handling labeled email datasets and feature extraction.

**Matplotlib (3.4.3) and Seaborn (0.11.2) (Optional):**

- Versions 3.4.3 and 0.11.2 or compatible versions of Matplotlib and Seaborn libraries are recommended for data visualization purposes.

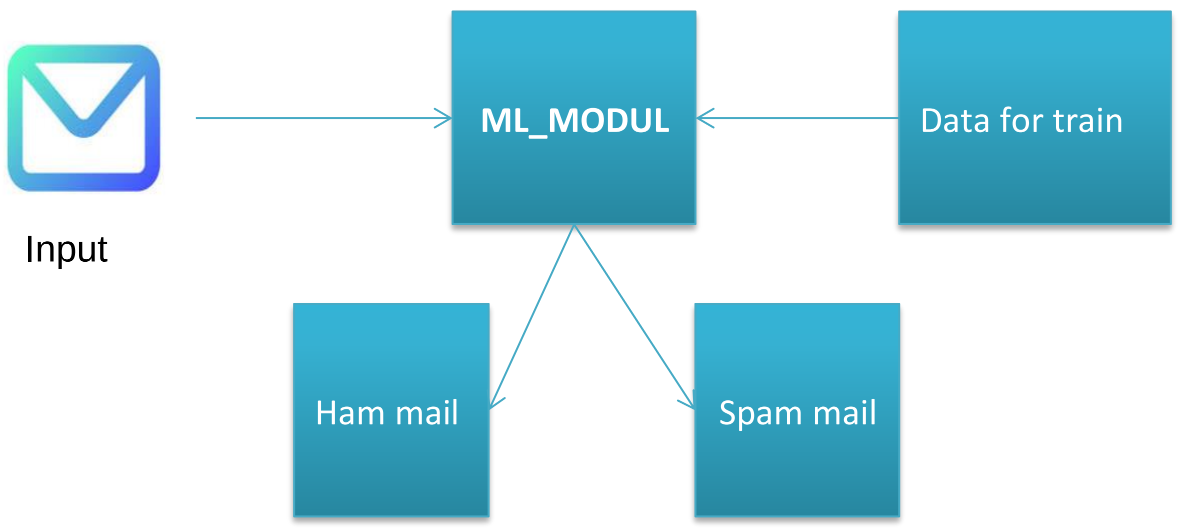
# CHAPTER 3

# PROJECT ARCHITECTURE

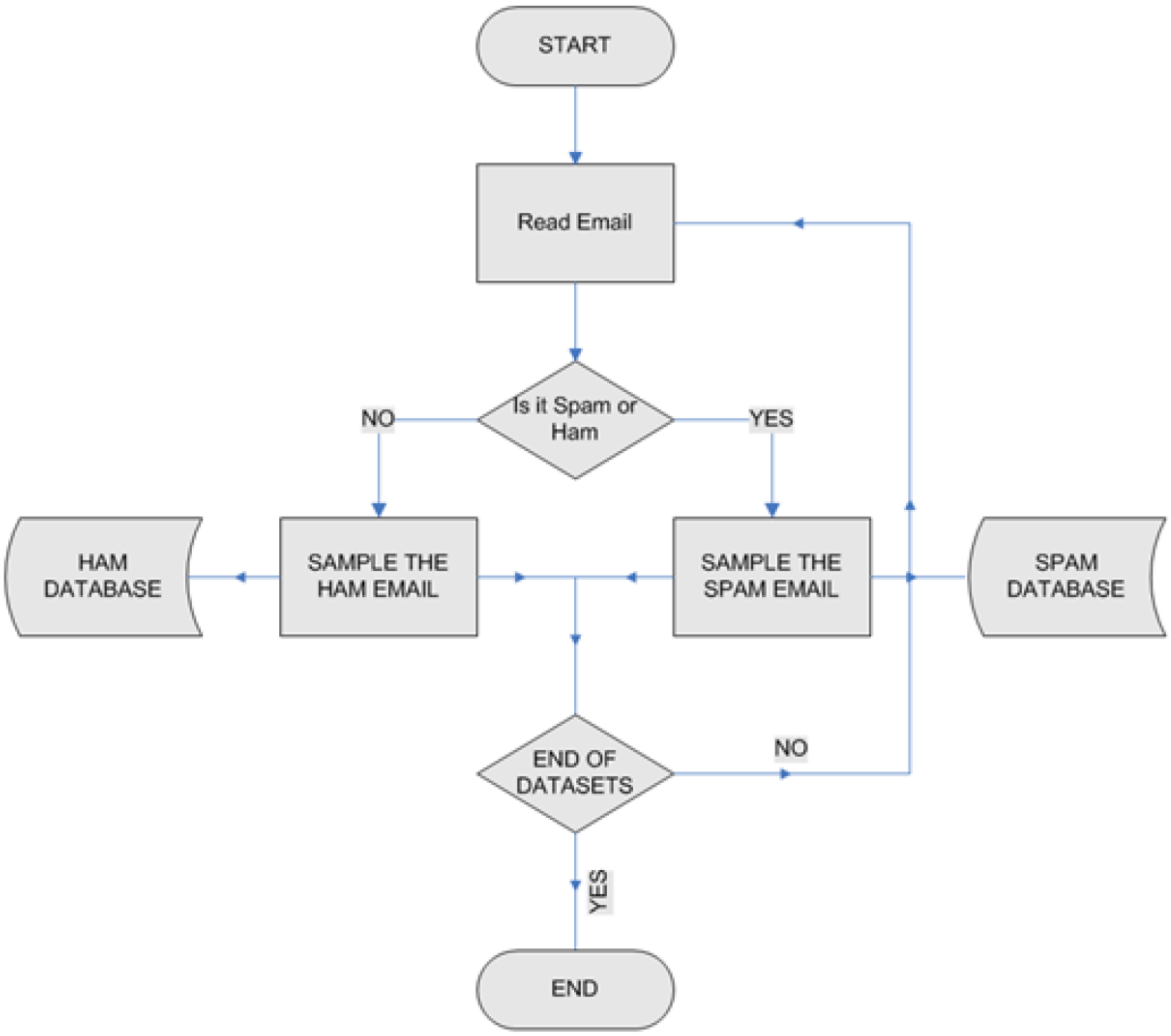
## 3.1 Architecture

### Email spam detection:

1. **System flow diagram**



## 2. Data Flow diagram



# CHAPTER 4 (code)

## MODELING AND PROJECT OUTCOME

### EDA – analysis report:

#### 1. Normalization

**Code:** `df['Category'] = encoder.fit_transform(df['Category'])`

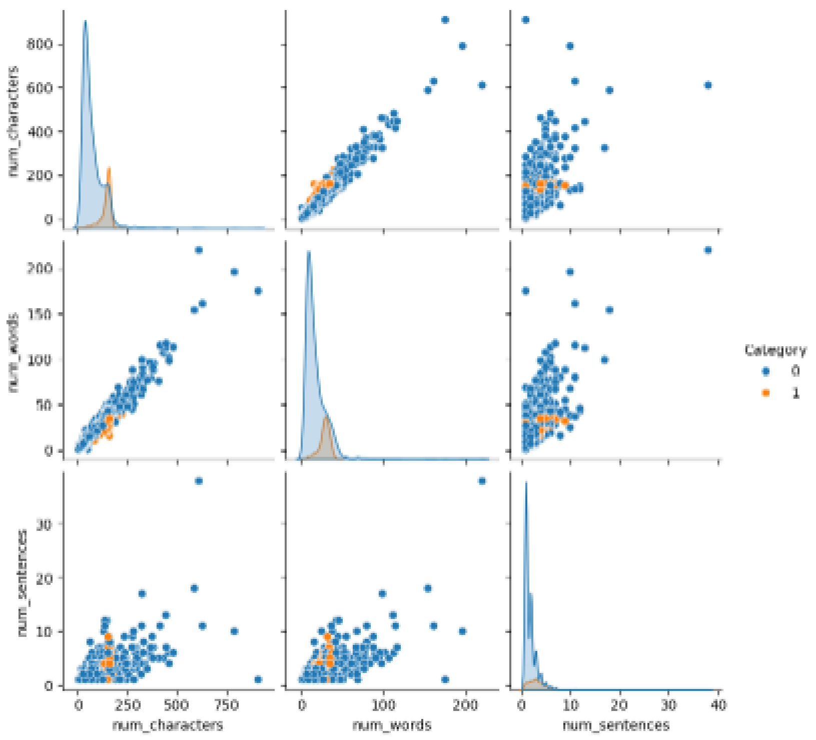
**Output:**

| | Category | 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... |

#### 2. Correlation

**Code:** `sns.pairplot(df,hue='Category')`

**Output:**



## 3. Outlier

**Code:**
```
#the mail was gave in the plase

input_your_mail = ["22 days to kick off! For Euro2004 U will be kept up
to date with the latest news and results daily. To be removed send GET
TXT STOP to 83222"]

input_data_feactures = feature_extraction.transform(input_your_mail)

prediction = model.predict(input_data_feactures)

print(prediction)

if(prediction[0]==1):
  print('Ham mail')
else:
  print('Spam mail')
```

**Output:**

```
Spam mail
```

## 4. Data Visualizations

**Code:** **1.**
```
import matplotlib.pyplot as plt

plt.pie(df['Category'].value_counts(),
labels=['ham','spam'],autopct="%0.2f")
plt.show()
```
2.
```
import seaborn as sns

plt.figure(figsize=(12,6))
sns.histplot(df[df['Category'] == 0]['num_characters'])
sns.histplot(df[df['Category'] == 1]['num_characters'],color='red')
```

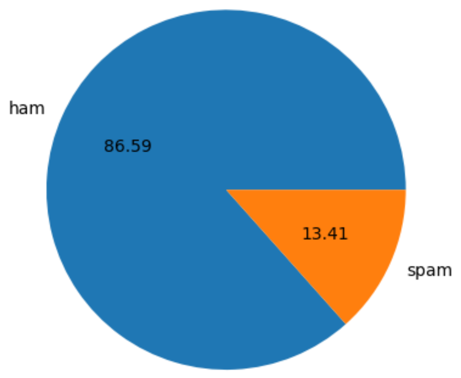
3.
```
plt.figure(figsize=(12,6))

sns.histplot(df[df['Category'] == 0]['num_words'])
sns.histplot(df[df['Category'] == 1]['num_words'],color='red')
```

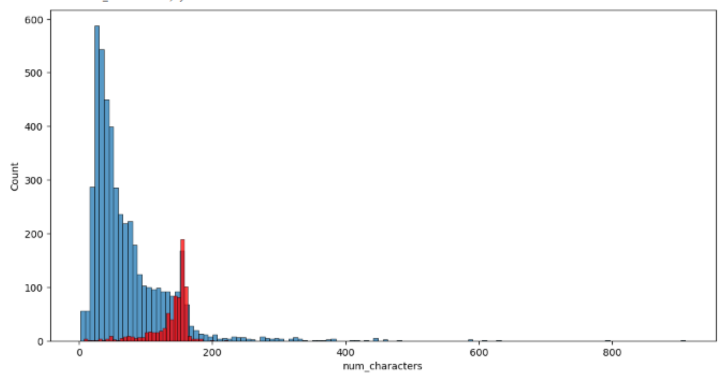
4.
```
sns.pairplot(df,hue='Category')
```

5.
```
 sns.catplot(x = 'Algorithm', y='value',
                   hue = 'variable',data=performance_df1,
kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```

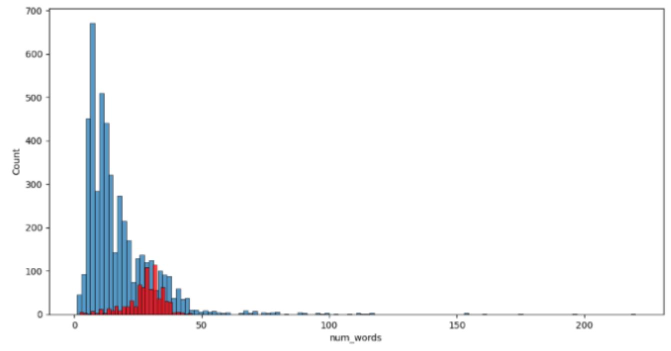
**Output:**

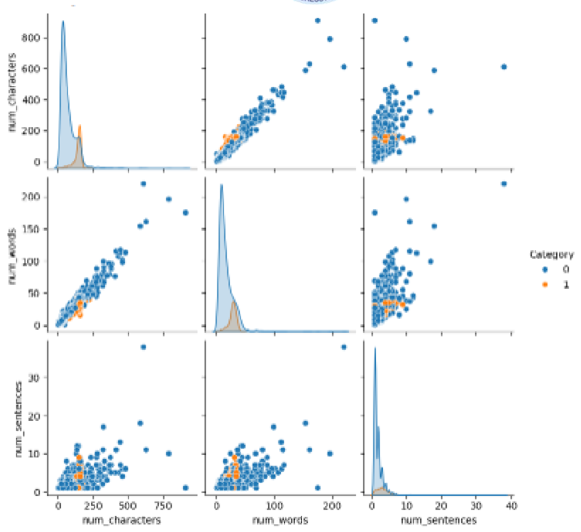


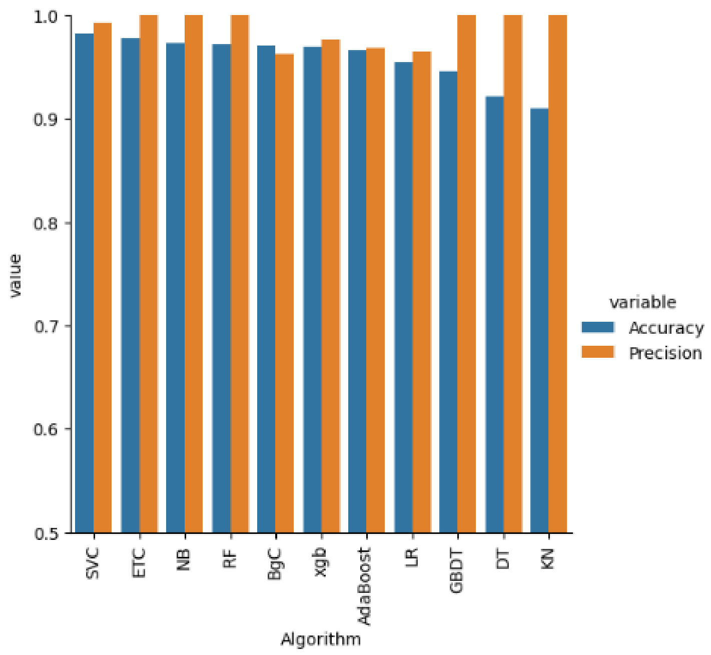
```
<Axes: xlabel='num_characters', ylabel='Count'>
```



```
<Axes: xlabel='num_words', ylabel='Count'>
```




num_characters
num_words
num_sentences
Category
0
1

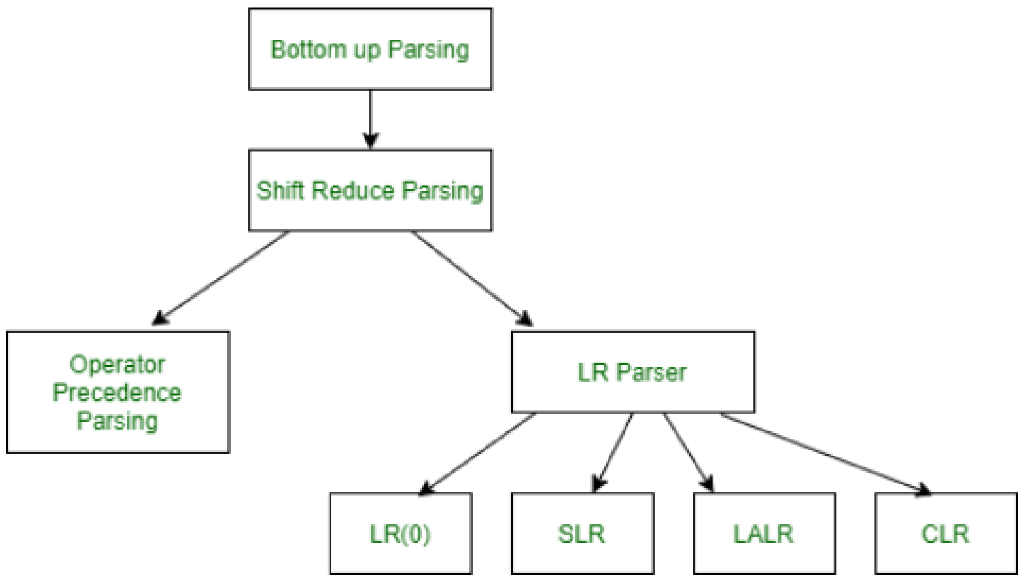

value
Algorithm
variable
Accuracy
Precision
SVC
ETC
NB
RF
BgC
xgb
AdaBoost
LR
GBDT
DT
KN

## Model output:

### 1. LR expalin (def, math f)

The LR(1) parser is a deterministic automaton and as such its operation is based on static state transition tables. These codify the grammar of the language it recognizes and are typically called "parsing tables".

### 2. How it working (diagram)



### 3. Training data – Model performance (code, with output)

```
     model = LogisticRegression()
     model.fit(X_train_features , Y_train)

   prediction_on_training_data = model.predict(X_train_features)

accuracy_on_training_data = accuracy_score(Y_train ,
prediction_on_training_data)
prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
```

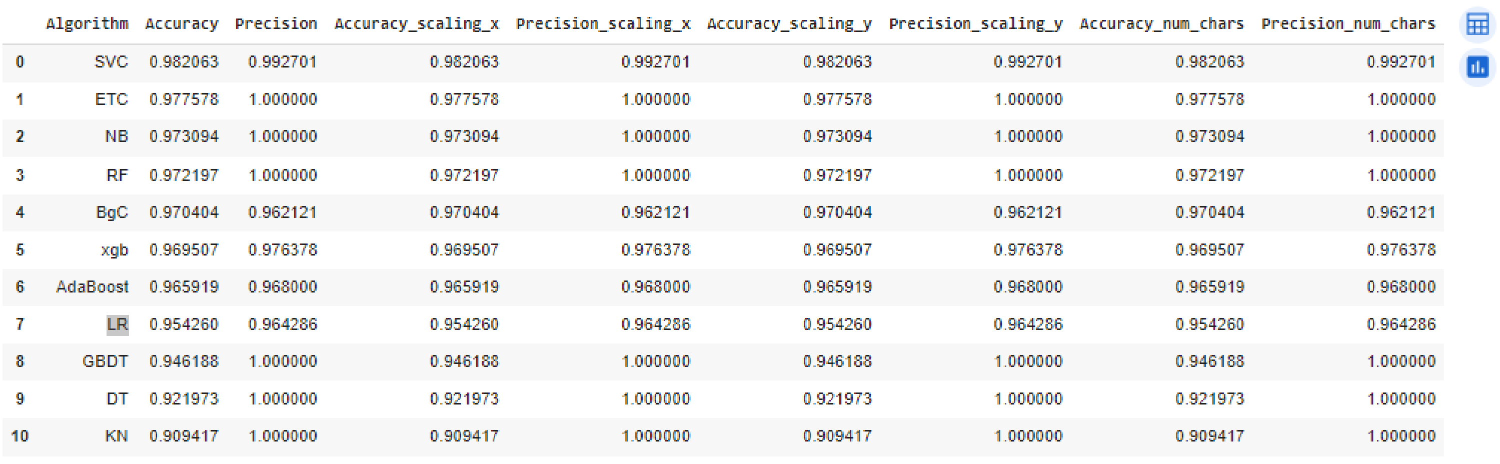
## 4. Output – Testing output (predicted output)

```
print(prediction)
```

**predicted output**

```
[0]
```

## 5. Compare the Model  (lr vs, KNN, NB,..etc)



| | Algorithm | Accuracy | Precision | Accuracy_scaling_x | Precision_scaling_x | Accuracy_scaling_y | Precision_scaling_y | Accuracy_num_chars | Precision_num_chars |
|---|---|---|---|---|---|---|---|---|---|
| 0 | SVC | 0.982063 | 0.992701 | 0.982063 | 0.992701 | 0.982063 | 0.992701 | 0.982063 | 0.992701 |
| 1 | ETC | 0.977578 | 1.000000 | 0.977578 | 1.000000 | 0.977578 | 1.000000 | 0.977578 | 1.000000 |
| 2 | NB | 0.973094 | 1.000000 | 0.973094 | 1.000000 | 0.973094 | 1.000000 | 0.973094 | 1.000000 |
| 3 | RF | 0.972197 | 1.000000 | 0.972197 | 1.000000 | 0.972197 | 1.000000 | 0.972197 | 1.000000 |
| 4 | BgC | 0.970404 | 0.962121 | 0.970404 | 0.962121 | 0.970404 | 0.962121 | 0.970404 | 0.962121 |
| 5 | xgb | 0.969507 | 0.976378 | 0.969507 | 0.976378 | 0.969507 | 0.976378 | 0.969507 | 0.976378 |
| 6 | AdaBoost | 0.965919 | 0.968000 | 0.965919 | 0.968000 | 0.965919 | 0.968000 | 0.965919 | 0.968000 |
| 7 | LR | 0.954260 | 0.964286 | 0.954260 | 0.964286 | 0.954260 | 0.964286 | 0.954260 | 0.964286 |
| 8 | GBDT | 0.946188 | 1.000000 | 0.946188 | 1.000000 | 0.946188 | 1.000000 | 0.946188 | 1.000000 |
| 9 | DT | 0.921973 | 1.000000 | 0.921973 | 1.000000 | 0.921973 | 1.000000 | 0.921973 | 1.000000 |
| 10 | KN | 0.909417 | 1.000000 | 0.909417 | 1.000000 | 0.909417 | 1.000000 | 0.909417 | 1.000000 |

## Manage relationship

**User-System Interaction:**

- **User Input:** Users interact with the system by uploading emails to be checked for spam.
- **System Response:** The system processes the uploaded emails, analyzes them using machine learning models, and provides feedback on whether the email is spam or legitimate.

**System-Database Interaction:**

- **Data Storage:** The system stores user preferences, email data, and model results in a database (e.g., SQLite).
- **Retrieval and Update:** The system retrieves user settings, email features, and model outputs from the database for processing and updates the database with new information.

**Model-Feature Interaction:**

- **Feature Extraction:** The system extracts features from email content, metadata, and attachments.
- **Feature Importance:** The model evaluates the importance of each feature in classifying emails as spam or legitimate.

- **Model Training and Tuning:** Based on feature analysis, the system trains and tunes machine learning models (e.g., Naive Bayes, SVM) to achieve optimal performance.

**Model-Data Interaction:**

- **Data Preprocessing:** The system preprocesses the email dataset, removing noise, and converting text into numerical features.
- **Model Training:** Machine learning models are trained using the preprocessed dataset to learn patterns and characteristics of spam emails.
- **Model Evaluation:** The system evaluates model performance using metrics such as accuracy, precision, recall, and F1-score.

**Feedback Loop:**

- **User Feedback:** Users can provide feedback on the system's classification results, marking emails as spam or legitimate.
- **Model Improvement:** The system uses user feedback to improve the model's accuracy and effectiveness in detecting spam emails.
- **Continuous Learning:** Through iterative updates and retraining, the system adapts to new spamming techniques and email patterns.

**Deployment and Scalability:**

- **Cloud Deployment (Optional):** The system can be deployed on cloud platforms like AWS, Azure, or Google Cloud for scalability and accessibility.
- **Containerization (Optional):** Docker can be used to containerize the application, making it easier to deploy and manage across different environments.
- **Monitoring and Maintenance:** The system's performance and health are monitored regularly, with updates and maintenance scheduled as needed.

**Security Measures:**

- **Data Encryption:** User data, email content, and system settings are encrypted to ensure privacy and security.
- **Authentication and Authorization:** Users are authenticated before accessing the system, and roles/permissions are assigned for secure access.
- **Email Attachment Handling:** The system scans email attachments for potential malware or malicious content before processing.

**Documentation and Support:**

- **User Guide:** A comprehensive user guide is provided to assist users in navigating the system, uploading emails, and managing spam settings.
- **Technical Documentation:** Detailed technical documentation is available for developers and administrators, outlining system architecture, components, and APIs.

## CONCLUSION

- Building an effective spam email detection model involves a series of crucial steps, from data cleaning and exploratory data analysis to model development and evaluation. Here’s a summary of our journey
- We loaded the **Email Spam Classification Dataset** containing information about 5,172 emails.
- Unnecessary columns were dropped, and the target variable was encoded.
- Missing values were handled, and duplicate rows were removed.
- We visualized the distribution of spam and non-spam emails using pie charts.
- EDA helped us understand the dataset’s characteristics and identify potential patterns.

## FUTURE SCOPE

By using the Naïve Bayes section we can determine if the email is spam or not by this we can save our time as we do not have to go through all the mail to check spam and non-spam emails. We can protect our knowledge in a variety of ways viruses as we do not have spam emails in our system. We can also be as safe from hackers as Internet criminals can get into our system with these spam emails. Many anti-spam filters provide spam email storage service for a few days. Allows you to make sure you don't useful emails are deleted along with junk mail. The Naïve Bayes Classifier is over accurate compared to other spam filtering techniques so use this process in the future very appropriate. Email has become a very important means of communication today, with internet connection any message can be delivered worldwide. More than 270 billion emails are changed daily, about 57% of which are spam emails. Spam emails, also known as private, unwanted commercial emails or malicious emails, affecting or hacking personal information like a bank, money related or anything that causes the destruction of one person or company or group of people. In addition to advertising, these may contain links to malicious theft of sensitive information or websites that contain malicious software designed to steal confidential information. Spam is a serious problem that can annoy not only end users but also financially harmful and security risks. This program is therefore designed to detect unwanted and unwanted emails and thus prevent them from reducing the spam message which can be of great benefit to individuals and the company

## REFERENCES

1. Github link : https://github.com/ramar92/TNSDC-NM-Engineering-Colleges.git

2. YouTube:

   https://www.youtube.com/watch?v=FkF2jhaRJIs&t=233s

## GIT Hub Link of Project Code:

https://github.com/Believer23/NM_Detecting_Spam_mail.git