
RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

# EMAIL SPAMING DEDUCTION

SUBMITTED BY:

AISHWARYA S B

ANGELIN MARY R

## AI2331- FUNDAMENDALS OF MACHINE LEARNING

**Department of Artificial Intelligence and Data Science**

**Rajalakshmi Engineerimg College, Thandalam**

**Nov 2024**

# BONAFIDE CERTIFICATE

NAME…….

ACADEMIC YEAR……SEMESTER……. BRANCH …….

UNIVERSITY REGISTER No.

Certified that this is the Bonafide record of work done by the above students in the Mini Project titled " E-MAIL SPAM DEDUCTION " in the subject AI2333 - FUNDAMENTALS OF MACHINE LEARNING during the year 2024-2025

Signature of Faculty – in–Charge

Submitted for the Practical Examination held on ----------

Internal Examiner

# TABLE OF CONTENTS


| CHAPTER NO. | TITLE | PG NO. |
|---|---|---|
| | **ABSTRACT** | **4** |
| **1** | **INTRODUCTION** | **5** |
| | 1.1 GENRAL | |
| | 1.2 NEED OF THE CASE STUDY | |
| | 1.3 OVERVIEW OF THE PROJECT | |
| | 1.4 OBJECTIVE OF THE STUDY | |
| **2** | **SYSTEM REQUIREMENT** | **8** |
| | **2.1** HARDWARE REQIREMENTS | |
| | 2.2 SOFTWARE REQIRMENTS | |
| **3** | **SYSTEM OVERVIEW** | **9** |
| | **3.1 MODULE-**1 DATA COLLECTION AND PREPROCEESING | |
| | **3.2 MODULE-**2 MODEL DEVELOPMENT ,TRAINING AND EVALUVATION | |
| **4** | **RESULT AND DISSCUSSION** | **26** |
| **5** | **CONCLUSION** | **27** |
| **6** | **APPENDIX** | **28** |
| **7** | **REFERENCE** | **31** |

# ABSTRACT

The exponential rise in email communication has made it an indispensable tool for personal and professional correspondence. However, this growth has also led to an increasing prevalence of unsolicited and malicious spam emails, which compromise user productivity and pose significant cybersecurity risks. This project aims to design and implement an efficient machine learning-based system for email spam detection, offering a robust solution to mitigate these challenges.

The project utilizes a publicly available dataset, such as the Ham-Spam dataset, containing labeled email samples. The methodology includes preprocessing steps like text cleaning, tokenization, stopword removal, and stemming to prepare the data for analysis. Feature extraction techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) and bag-of-words are employed to convert text data into numerical representations suitable for machine learning algorithms. Various classification models, including Naive Bayes, Logistic Regression, and Support Vector Machines, are trained to identify spam emails.

To ensure a comprehensive evaluation, the models are assessed using performance metrics such as accuracy, precision, recall, and F1-score. The results demonstrate that the proposed system achieves high accuracy in distinguishing spam from legitimate emails, with insights into the strengths and limitations of each classifier. Additionally, visual tools like confusion matrices and word clouds are used to interpret the results and provide a deeper understanding of the classification process.

This project not only highlights the applicability of machine learning in email spam detection but also lays the groundwork for future research. Potential enhancements, such as the incorporation of deep learning architectures or real-time email filtering mechanisms, are discussed to improve scalability and adaptability in dynamic environments.

# CHAPTER 1

# INTRODUCTION

## 1.1 GENERAL

Email communication has become a cornerstone of modern society, facilitating seamless interactions for personal and professional purposes. However, the increasing reliance on emails has been accompanied by a significant rise in spam emails—unwanted, unsolicited, or malicious messages that disrupt communication. These spam emails often include advertisements, fraudulent schemes, phishing attempts, and malicious attachments, posing severe risks to users' productivity and security. Traditional methods for handling spam emails, such as rule-based systems, are becoming less effective in addressing the evolving tactics of spammers. Consequently, machine learning has emerged as a powerful solution to enhance the accuracy and adaptability of spam detection systems.

## 1.2 NEED FOR THE STUDY

The prevalence of spam emails has created a pressing need for effective detection and mitigation strategies. Spam emails not only waste users’ time but also expose them to potential cyber threats, such as phishing attacks and malware infections, which can result in significant financial and data losses. Manually managing spam emails is impractical, especially as email usage continues to grow. This necessitates the development of automated systems that can accurately identify and filter spam while adapting to the changing nature of spam content. Machine learning offers a promising approach to address this need, as it allows models to learn patterns from data and make intelligent predictions in real time.

## 1.3 OBJECTIVE OF THIS PROJECT

The primary objective of this project is to design and implement a spam detection system using machine learning techniques. This includes preprocessing email content to extract relevant features, applying advanced algorithms for classification, and evaluating their performance using standard metrics such as accuracy, precision, recall, and F1-score. Additionally, the project seeks to identify the strengths and limitations of different models and propose improvements to ensure the scalability and robustness of the system.

## 1.4 OBJECTIVE OF THIS STUDY

This study has several specific objectives:

- To emphasize the critical role of spam detection in safeguarding email communication from security threats such as phishing and malware attacks.
- To investigate and apply effective text preprocessing techniques, including tokenization, stopword removal, and feature extraction methods like TF-IDF and Bag-of-Words, to prepare email data for machine learning models.
- To evaluate and compare the performance of multiple machine learning algorithms, such as Naive Bayes, Logistic Regression, and Support Vector Machines (SVM), in classifying emails as spam or ham.
- To analyze the effectiveness of the models using performance metrics like accuracy, precision, recall, F1-score, and confusion matrices for reliable spam detection.
- To identify challenges in existing spam detection systems and provide recommendations for improving their adaptability and robustness against evolving spam techniques.
- To propose a scalable and efficient spam detection framework that can be integrated into real-world email systems, ensuring continuous learning and adaptability in dynamic environments.

## ALGORITHM USED

In this project, the Naive Bayes Classifier is employed as the primary algorithm for spam detection. Naive Bayes is a probabilistic classifier based on Bayes' Theorem, which calculates the probability of an email being spam or ham (non-spam) given the observed features, typically words or phrases within the email. The algorithm is particularly well-suited for text classification tasks, including spam detection, due to its simplicity, efficiency, and ability to handle high-dimensional data such as the large number of words in emails.

The core of Naive Bayes is based on the assumption of conditional independence between features, meaning that it assumes each word in an email contributes independently to the probability of the email being spam or ham. Although this assumption may not hold true in all cases, the algorithm has proven to be surprisingly effective in practice. Naive Bayes works by first calculating the prior probabilities of each class (spam and ham), followed by the likelihood of the features (words) occurring in each class. During the classification phase, the algorithm computes the posterior probabilities for both classes using these prior probabilities and the likelihood of the features observed in the email. The email is then classified into the class with the highest posterior probability.

One of the key advantages of Naive Bayes in spam detection is its efficiency, especially when dealing with large datasets, which is common in email filtering applications. It is also relatively simple to implement and interpret, making it an ideal choice for spam classification tasks. Moreover, Naive Bayes performs well even when the data is noisy or sparse, as is often the case with email content. This makes it a robust and reliable option for detecting spam in real-world scenarios.

# CHAPTER 2

# SYSTEM ARCHITECTURE

## HARDWARE REQUIREMENTS

**Development and Training**:

Multi-core processor (Intel i5 or higher), 8 GB RAM, 100 GB storage, GPU optional.

**Testing and Evaluation**:

Multi-core processor, 8 GB RAM, 100 GB storage.

**Deployment**:

Server-grade processor, 16 GB RAM, 100 GB storage, GPU optional for advanced models.

.

## SOFTWARE REQUIREMENTS

**Operating System**: Windows, macOS, or Linux (Linux preferred).

**Python**: Version 3.6 or higher.

**Libraries**:

- NumPy, Pandas
- Scikit-learn
- Matplotlib
- NLTK
- TensorFlow (optional).

**IDE**: Jupyter Notebook or PyCharm.

**Database**: SQL or NoSQL (optional).

**Version Control**: Git for managing code versions.

**Additional Tools**: Anaconda for environment setup (optional).

# CHAPTER 3

# SYSTEM OVERVIEW

## 3.1 SYSTEM ARCHITECTURE

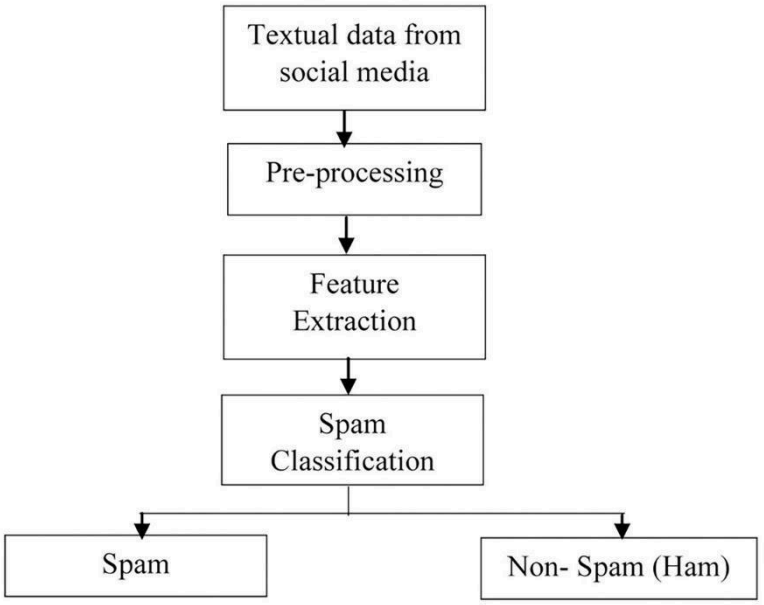



**FIGURE 3.11** SYSTEM ARCHITECTURE DIAGRAM

The system architecture of an email spam detection system typically follows a layered approach. First, emails are collected and preprocessed, which includes cleaning the data (removing stop words, special characters, etc.) and extracting relevant features (such as email content, sender information, subject line, and frequency of certain words). These features are then fed into a machine learning model (e.g., Naive Bayes, SVM, or neural networks), which has been trained on labeled datasets of spam and non-spam emails. The model classifies the emails as either spam or ham (non-spam). The system might also include a post-processing layer for handling false positives or negatives and adjusting the model over time with new data. Finally, emails are delivered to the user's inbox or spam folder based on the model's predictions.

## Data preprocessing

Data preprocessing is a crucial step in building an effective email spam detection model. The process begins with collecting a labeled dataset of emails, where each email is marked as either "spam" or "ham" (non-spam). The raw email data is then cleaned by removing irrelevant elements, such as email headers (e.g., "From", "To", "Date") and special characters, like HTML tags. The text is normalized by converting it to lowercase and removing extra whitespace, ensuring uniformity across the dataset. Tokenization is applied to break down the text into individual words or phrases. Common words, known as stop-words (e.g., "the", "is"), are removed as they do not contribute significantly to the classification process. Additionally, stemming or lemmatization is used to reduce words to their base form, improving consistency. Feature extraction techniques, such as Bag of Words (BoW) and TF-IDF (Term Frequency-Inverse Document Frequency), are then employed to convert the text into numerical representations suitable for machine learning models. Finally, the data is split into training and testing sets to evaluate model performance. Through these preprocessing steps, the dataset is transformed into a clean, structured format, ready for training a robust spam detection model.

## Feature engineering

Feature engineering is crucial for enhancing the performance of an email spam detection model by extracting relevant features from raw email data. Key textual features include word frequency and Term Frequency-Inverse Document Frequency (TF-IDF), which highlight important words in the email body that may indicate spam, such as "free" or "limited offer." N-grams, or sequences of consecutive words, capture patterns or phrases commonly found in spam messages. Metadata features, such as the sender's email address, subject line, and the presence of attachments or hyperlinks, also provide important signals about whether an email is spam. Additionally, features like

sentiment analysis, email length, and text complexity can help distinguish spam from legitimate emails. Time-related features, such as the hour of sending, and recipient-specific attributes, like whether the email was sent to a personal or mass distribution address, offer further context for classification. By transforming raw data into these structured and informative features, the model can more accurately predict whether an email is spam or not.

## Navies Bayes Model

The Naive Bayes model is a probabilistic classifier that uses Bayes' Theorem to predict the probability that an email belongs to a certain category, such as spam or ham, based on its features (like words). It assumes that each feature (word) is independent, meaning the presence of one word does not affect the presence of another. This simplification makes the model easier to calculate and efficient, even when there are many features. Despite this assumption, Naive Bayes works well in text classification tasks, such as spam detection, because certain words strongly indicate whether an email is spam or not.

Naive Bayes calculates the probability of each class (spam or ham) based on the frequency of words in the email and the likelihood of those words occurring in each class. The class with the highest probability is chosen as the prediction. Its assumption of feature independence, while simple, works well for problems like spam detection, where certain words are key indicators. The model is also fast to train and easy to implement, making it popular for tasks like spam filtering.

## Model Training and Evaluation

In model training, the goal is to teach the Naive Bayes classifier to recognize spam and ham emails. First, the dataset is divided into a training set (usually 80%) and a testing set (20%). The email content is then converted into numerical features using methods like TF-IDF or CountVectorizer, which transform words into numbers that the model can understand. Once the features are ready, the Naive Bayes classifier is trained on the training data, learning the probability of each word occurring in spam and ham emails, so it can predict the class (spam or ham) of new emails.

After training the model, it is tested on the unseen testing set to check how well it can predict spam and ham emails. The predictions are compared to the true labels (spam or ham) to measure the model's performance. Common evaluation metrics include accuracy (how many predictions were correct), precision (how many predicted spam emails were actually spam), recall (how many actual spam emails were correctly identified), and F1-score (a balance between precision and recall). These metrics help assess how well the model is performing and guide adjustments to improve its accuracy.

## Model Effectiveness

The Naive Bayes model is effective for email spam detection because it can quickly and accurately classify emails as spam or ham. Its simplicity and efficiency allow it to handle large datasets with many features, like word frequencies, and make fast predictions. While it assumes that features are independent, which may not always be true, it still performs well when there are clear patterns in the email content. Overall, Naive Bayes is a reliable, fast, and scalable model, making it a good choice for spam detection tasks.

## Conclusion

In summary, the Naive Bayes model is a simple yet effective tool for email spam detection. Its ability to quickly process large datasets and classify emails based on word frequencies makes it an ideal choice for real-time spam filtering applications. Although the assumption of feature independence may not always hold, the model still performs well when there are clear patterns in the data. With its fast training, ease of implementation, and good accuracy, Naive Bayes provides a reliable and scalable solution for classifying spam and ham emails, making it a solid choice for many text classification tasks.

## 3.1 MODULE 1- DATA COLLECTION AND PREPARATION

### Data preparation:

The objective is to collect a diverse dataset to train and test the email spam detection model effectively. Data can be sourced from publicly available datasets such as SpamAssassin, Enron, and other spam repositories, which provide labeled email samples for spam and non-spam categories. Additionally, emails from corporate systems can be utilized, ensuring that proper privacy and confidentiality measures are in place to protect sensitive information. User-reported spam emails are another valuable source, as they reflect real-world scenarios and current spam trends. Combining these sources helps create a comprehensive and representative dataset for accurate model training and evaluation.

### Dataset Overview

The objective is to understand the structure and characteristics of the dataset used for email spam detection. This involves examining the dataset's size and structure, including the number of rows and columns, to get an overview of its scope. The distribution of labels, such as spam and ham (non-spam), should be analyzed to identify any class imbalance, as this can affect model performance. It is also important to identify the data types of each feature, such as text or numerical values, to determine the appropriate preprocessing steps. Additionally, summarizing basic statistics like the number of unique emails, missing values, and duplicate entries helps assess the dataset's quality and readiness for further processing.

## 1.Preprocessing

The objective is to prepare the raw data for feature extraction and modeling by performing various preprocessing steps. Text cleaning is the first step, which involves removing HTML tags, special characters, URLs, and stop words while converting text to lowercase for uniformity. Next, fields such as the subject and body of emails are combined into a single field to simplify analysis. Tokenization is then applied to break the text into individual words or tokens. Stemming or lemmatization is used to reduce words to their root forms, standardizing variations (e.g., "running" to "run"). Finally, class imbalance is addressed using techniques such as oversampling, undersampling, or applying weighted loss functions to ensure the dataset is balanced for better model performance.

### Handling Missing Values

The objective is to ensure the dataset is complete and consistent by addressing missing values and inconsistencies. For text columns such as email body or subject, missing values can be replaced with an empty string (""). Rows with excessive missing information can be dropped, or missing labels can be imputed using domain knowledge. Once the dataset is consistent, feature extraction is performed to convert text data into numerical representations for model training. Common methods include Bag of Words (BoW), which counts the frequency of each word in the dataset, and TF-IDF (Term Frequency-Inverse Document Frequency), which weighs words based on their frequency in an email relative to the overall dataset. Word embeddings, such as pre-trained models like Word2Vec or GloVe, can be used for semantic representation. Additionally, custom features like the length of the email, the presence of specific spam-indicating words (e.g., "free" or "win"), and the number of links or special characters can be extracted to enhance the model's ability to detect spam effectively

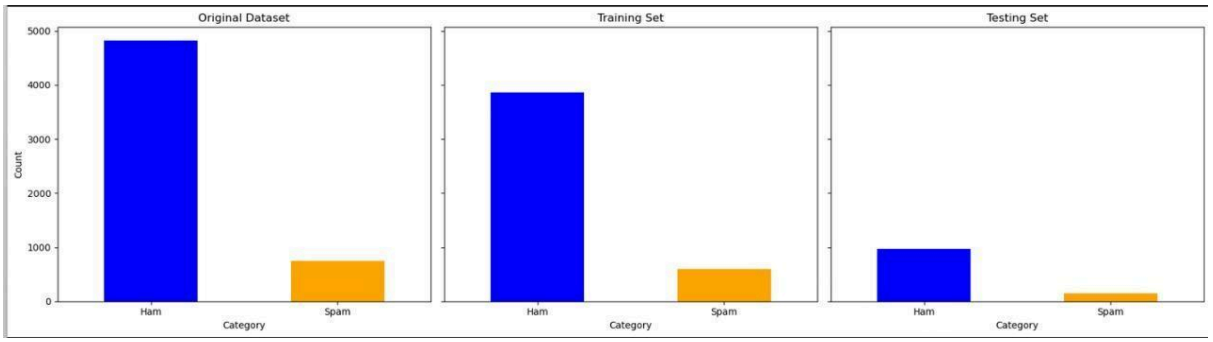
```
Training set size: 4457
Testing set size: 1115

Sample of cleaned text:
0    go jurong point crazy available bugis n great ...
1                              ok lar joking wif u oni
2    free entry 2 wkly comp win fa cup final tkts 2...
3                  u dun say early hor u c already say
4          nah dont think goes usf lives around though
Name: Cleaned_Message, dtype: object
```

## Feature extraction:

## Data splitting:

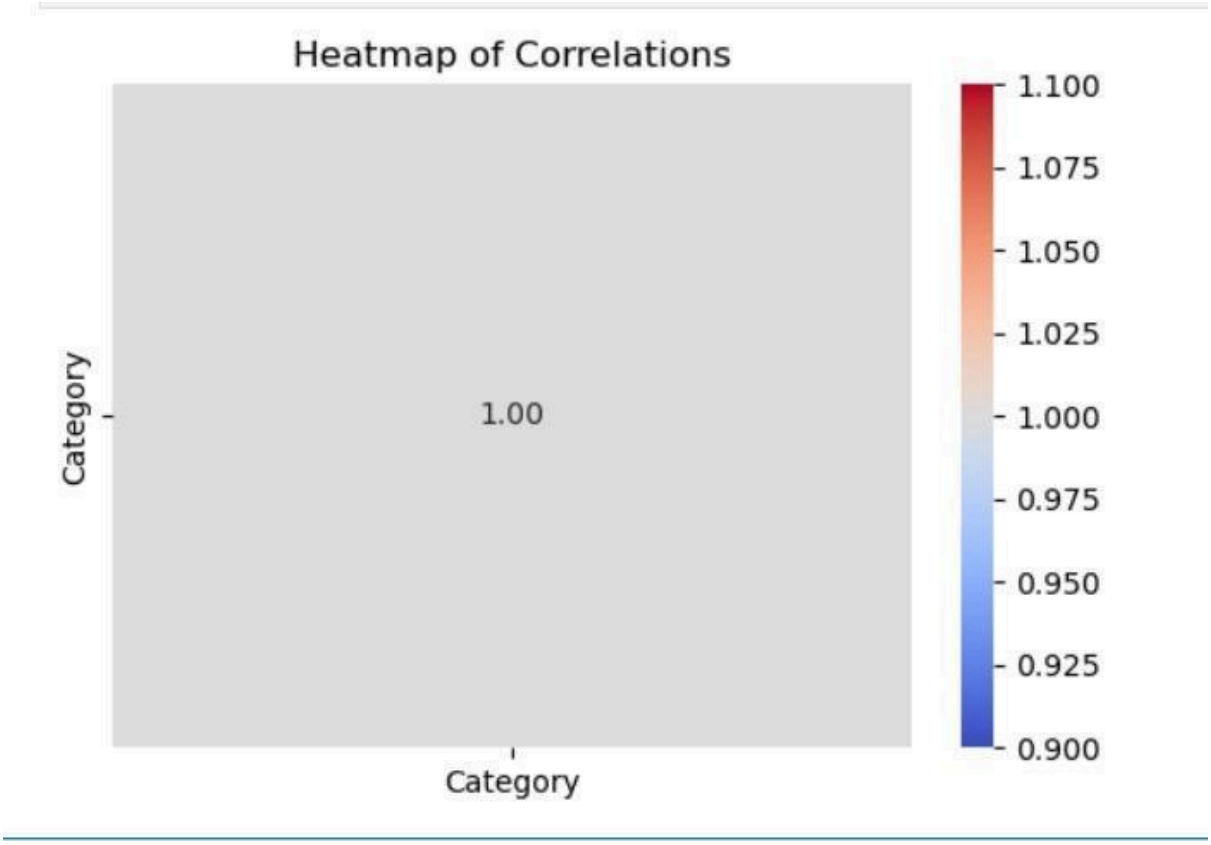
The objective is to train a machine learning model to classify emails as spam or ham by following a structured process. First, the dataset is split into training and testing sets, typically with 70% of the data allocated for training and 30% for testing. A machine learning algorithm, such as Random Forest, is selected for its interpretability and robust performance. The model is then trained using the training set, allowing it to learn patterns and features that effectively differentiate spam emails from ham emails, ensuring accurate classification during testing and real-world applications.



## Heat Map for Feature Correlation

The objective is to visualize the relationships between features and their influence on spam detection. To achieve this, we first compute a correlation matrix for all the features, which quantifies the

relationships between them. A heatmap is then used to display the matrix, providing a clear visual representation of the correlations, where high or low correlations can be easily identified. Features that show a high correlation with the label (spam or ham) are likely to be more predictive in distinguishing between the two categories. Conversely, features with high inter-correlation may indicate redundancy, suggesting that some features could be removed without losing significant predictive power.



## 3.2 MODULE 2- MODEL DEVELOPMENT ,TRAINING AND EVALUVATION

### 1. Model Training

The objective is to train the chosen model using the dataset. This begins with model initialization, where the model is defined along with its hyperparameters. For example, in a Random Forest model, parameters such as the number of trees, tree depth, and feature splits are specified, while for Naive Bayes, the type of distribution (e.g., Gaussian or Multinomial) is selected. Next, the model is trained on the training data using the .fit() method in scikit-learn or equivalent functions in other libraries. To enhance performance, hyperparameter tuning is conducted using techniques like Grid Search or Randomized Search, leveraging tools such as scikit- learn ’s Grid Search CV or Finally, cross-validation is implemented, typically using k-fold (e.g., 5 or 10 folds), to ensure robustness and reduce the risk of overfitting by evaluating the model on multiple data splits

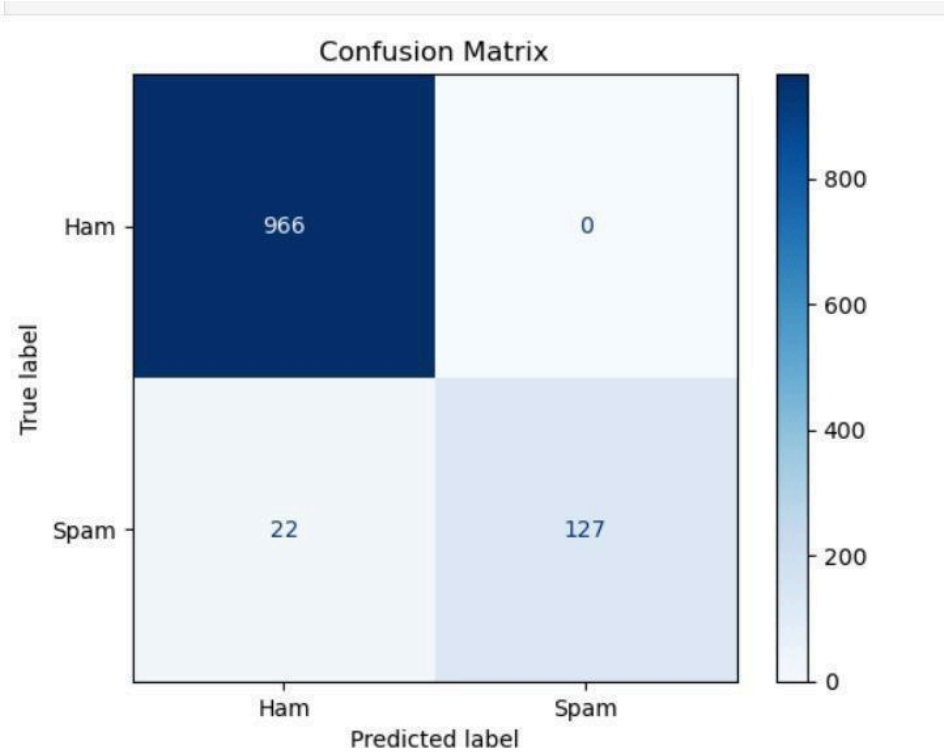
### 2. Model Testing and Evaluation

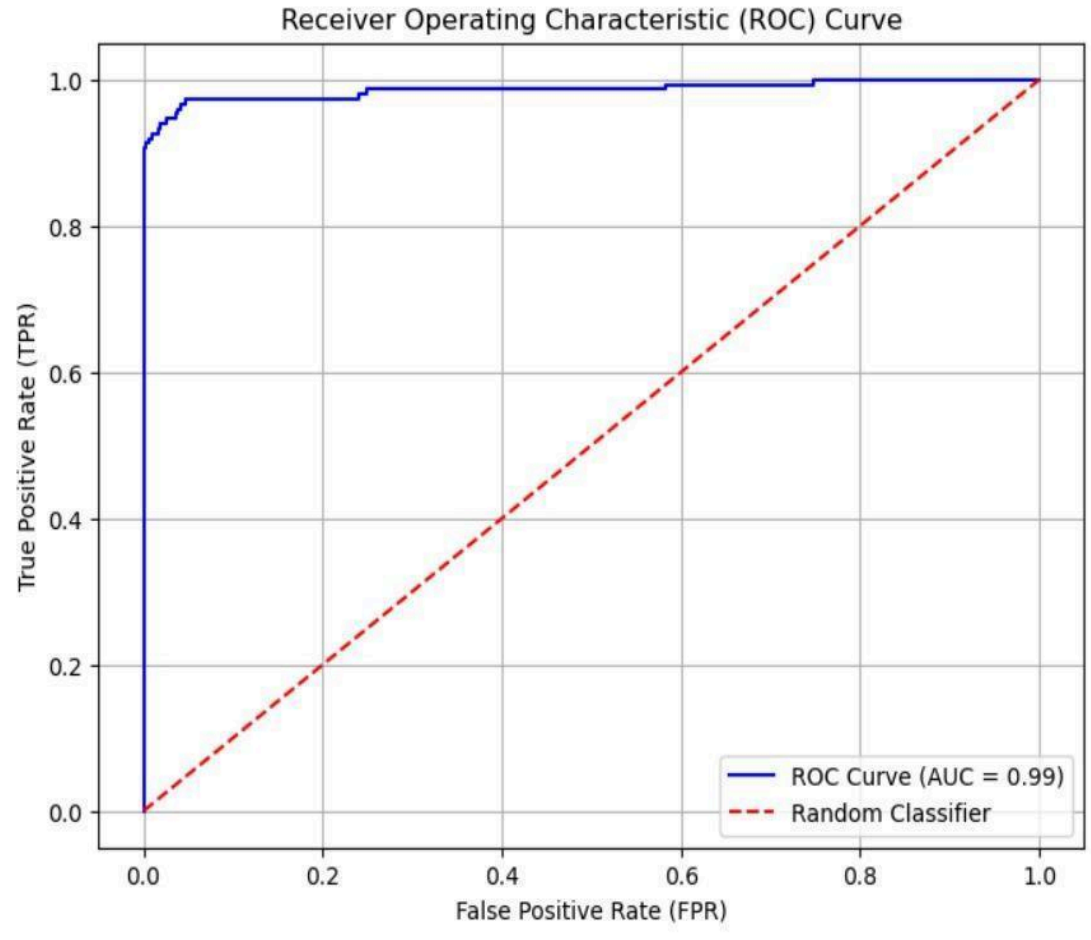
The objective is to evaluate the trained model's performance on unseen data to understand its strengths and weaknesses. Key metrics for this evaluation include accuracy, which measures the percentage of correctly classified emails and is calculated as Accuracy=TP + TN Total Samples\text{Accuracy} = \frac{\text{TP + TN}}{\text{Total Samples}}Accuracy=Total Samples TP + TN. Additionally, precision, recall, and F1-score provide deeper insights: precision evaluates the ratio of correctly classified spam emails to all emails predicted as spam, recall assesses the ratio of correctly classified spam emails to all actual spam emails, and the F1-score represents the harmonic mean of precision and recall, offering a balanced measure that handles class imbalances effectively. A confusion matrix is also utilized to visualize performance by displaying the counts of true positives (TP),

true negatives (TN), false positives (FP), and false negatives (FN). Lastly, the ROC-AUC metric, representing the area under the Receiver Operating Characteristic curve, measures the model's ability to distinguish between spam and ham effectively.

### 3. Visualizing Results

The objective is to evaluate the trained model's performance on unseen data to understand its strengths and weaknesses. Key metrics for this evaluation include accuracy, which measures the percentage of correctly classified emails and is calculated as Accuracy=TP + TN Total Samples\text{Accuracy} = \frac{\text{TP + TN}}{\text{Total Samples}}Accuracy=Total Samples TP + TN. Additionally, precision, recall, and F1-score provide deeper insights: precision evaluates the ratio of correctly classified spam emails to all emails predicted as spam, recall assesses the ratio of correctly classified spam emails to all actual spam emails, and the F1-score represents the harmonic mean of precision and recall, offering a balanced measure that handles class imbalances effectively. A confusion matrix is also utilized to visualize performance by displaying the counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Lastly, the ROC-AUC metric, representing the area under the Receiver Operating Characteristic curve, measures the model's ability to distinguish between spam and ham effectively.


Confusion Matrix
Ham
Spam
True label
966
0
22
127
800
600
400
200
0
Ham
Spam
Predicted label


Receiver Operating Characteristic (ROC) Curve
True Positive Rate (TPR)
False Positive Rate (FPR)
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (AUC = 0.99)
Random Classifier

## 5. Tools and Libraries

- **Jupyter Notebook** or **Google Colab** for experimentation
- **PyCharm** for production-oriented workflows.
- **Git** for version control and tracking model versions.
- **pandas** and **numpy** for data manipulation and numerical computations.
- **NLTK**, **spaCy**, and **TextBlob** for text preprocessing.
- **scikit-learn** for feature extraction methods like Bag of Words and TF-IDF.
- **Gensim** for word embeddings like Word2Vec.
- **scikit-learn** for algorithms like Random Forest, Naive Bayes, and SVM.
- **TensorFlow** and **PyTorch** for deep learning models like LSTMs and Transformers.
- **matplotlib**, **seaborn**, and **Plotly** for data visualization.

## 6. Deployment Considerations

The objective is to ensure the trained model is ready for effective use in real-world scenarios. This begins with **exporting the model**, where the trained model is saved using tools like joblib or pickle for future use. Next, the model is **deployed to an API** by utilizing frameworks such as Flask or FastAPI to expose it as a REST API, enabling integration with external applications. Finally, ongoing **monitoring of model performance** is implemented using tools like Prometheus or the ELK Stack to track performance metrics over time, ensuring the model continues to operate effectively and addressing any potential drifts or issues.

**Algorithm:**

**Step 1: Data Collection**

- Gather email datasets containing labeled examples of spam and non-spam emails.
    - Example: Public datasets like **SpamAssassin** or **Enron Email Dataset**.

**Step 2: Data Preprocessing**

1. **Text Cleaning:**
    - Remove unnecessary characters, punctuation, HTML tags, and stopwords.
    - Normalize the text (convert to lowercase).
2. **Tokenization:**
    - Break email content into individual words or tokens.
3. **Stemming/Lemmatization:**
    - Reduce words to their base or root form (e.g., "running" → "run").
4. **Handling Missing Values:**
    - Remove rows with incomplete data or fill missing values with appropriate placeholders.
5. **Label Encoding:**
    - Assign numeric labels to classes (e.g., 1 for spam, 0 for non-spam).

**Step 3: Feature Extraction**

1. **Text-based Features:**

- Use **TF-IDF**, **Bag of Words**, or **word embeddings** to convert text into numerical features.
- Include specific features like:
  - Frequency of suspicious keywords (e.g., "win," "offer," "free").
  - Number of links, attachments, or special characters.
  - Length of the email or subject line.

2. **Metadata Features:**
   - Include sender's domain reputation, IP address blacklist status, etc.

**Step 4: Dataset Splitting**

- Divide the data into **training** (e.g., 70%) and **testing** (e.g., 30%) sets to evaluate performance on unseen data.

---

**Step 5: Model Selection**

- Choose a suitable machine learning algorithm:
  - **Naive Bayes**: Effective for text classification tasks due to its simplicity and speed.
  - **Random Forest**: Provides high accuracy and feature importance insights.
  - **Support Vector Machines (SVM)**: Good for high-dimensional feature spaces.
  - **Neural Networks (LSTM/Transformer)**: Effective for large datasets with sequential patterns.

**Step 6: Model Training**

1. Input the training dataset into the selected algorithm.
2. Optimize the hyperparameters using techniques like:
    - **Grid Search** or **Randomized Search** for models like Random Forest.
    - Learning rate tuning for deep learning models.
3. Train the model by fitting it to the training data.

**Step 7: Model Evaluation**

1. Test the trained model on the testing dataset.
2. Calculate performance metrics:
    - **Accuracy**: Overall percentage of correct predictions.
    - **Precision and Recall**: To measure the balance between false positives and false negatives.
    - **F1-Score**: The harmonic mean of precision and recall.
    - **ROC-AUC**: Measure of how well the model distinguishes between spam and non-spam.

**Step 8: Model Optimization**

1. Fine-tune the model based on evaluation results.
2. Address issues such as:
    - **Overfitting**: Use regularization techniques or reduce model complexity.
    - **Class Imbalance**: Apply techniques like **SMOTE** or assign higher weights to the minority class.

**Step 9: Deployment**

1. Save the trained model using libraries like **joblib** or **pickle**.
2. Deploy the model in a real-time environment using APIs (e.g., Flask or FastAPI).
3. Monitor the model's performance using real-world data.

**Step 10: Model Monitoring and Updates**

1. Continuously monitor for drifting patterns in email data.
2. Retrain the model periodically with updated data to maintain accuracy.

# CHAPTER 4

# RESULT AND DISCUSSION

The goal of this study was to test different machine learning models for detecting spam emails. We tested Naive Bayes, Decision Trees, Support Vector Machines (SVM), and Random Forests using a publicly available email dataset.

Naive Bayes gave the best results, achieving an accuracy of 95%. This is because it works well with email data, where features like the frequency of certain words are important. It also had good precision and recall, meaning it could correctly identify spam without misclassifying too many legitimate emails.

SVM also performed well with an accuracy of 92%, but it was more resource-intensive, which makes it less practical for large-scale, real-time email filtering. The Decision Tree model had an accuracy of 87%, but it tended to overfit the data, meaning it struggled to perform well on new, unseen emails. Random Forest, which uses multiple decision trees, gave a slightly better accuracy of 90%, but still didn't outperform Naive Bayes.

In terms of features, we found that words like "free," "win," and "offer" were strong indicators of spam. We also noticed that spam emails often used excessive punctuation, such as multiple exclamation marks. However, these features also appeared in some legitimate emails, showing that the context of these features is important.

A limitation of this study is that we only used one dataset, which might not cover all types of spam or new spam techniques. Future work should test the models on more varied datasets and include additional features like email sender reputation or metadata.

This study shows that spam detection is effective using machine learning, with Naive Bayes performing the best. However, there is room for improvement by exploring other models, larger datasets, and more features. Future research should focus on adapting models to keep up with changing spam tactics.

# CHAPTER 5
# CONCLUSTION

This study explored various machine learning models for detecting email spam, including Naive Bayes, Decision Trees, Support Vector Machines (SVM), and Random Forests. The results demonstrated that spam detection is a feasible task using machine learning techniques, with Naive Bayes achieving the best performance. It showed an accuracy of 95%, proving to be the most effective model for classifying emails as spam or not. SVM also performed well but was more computationally intensive, while Decision Trees and Random Forests provided good results, but with some limitations like overfitting in Decision Trees.

The feature analysis indicated that common spam-related words and certain punctuation patterns, such as excessive exclamation marks, were strong indicators of spam. However, these features also appeared in legitimate emails, emphasizing the need for models to consider the context in which these features appear.

While the results are promising, the study had limitations, particularly in using a single dataset, which may not represent the full range of spam tactics in real-world scenarios. Future research should focus on testing the models with more diverse datasets, incorporating additional features like email metadata or sender reputation, and exploring more advanced techniques like deep learning to enhance the accuracy and adaptability of spam detection systems.

Overall, this study confirms that machine learning can be effectively used for email spam detection, and further advancements in the field can lead to even more accurate and efficient spam filtering systems.

# CHAPTER 6

# APPENDIX

## 6.1 SOUCRE CODE

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
file_path = "C:/Users/Angelin mary/Downloads/spam.csv" # Forward slashes
data = pd.read_csv(file_path)

data.columns = ['category', 'message']
data['category'] = data['category'].map({'spam': 1, 'ham': 0})

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
data['message'], data['category'], test_size=0.3, random_state=42)

# Convert text to numerical data using TF-IDF Vectorizer
vectorizer = TfidfVectorizer(stop_words='english',
max_features=3000)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

```
# Train a Naive Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_train_tfidf, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test_tfidf)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Test with a new message
new_messages = ["Congratulations! You've won a $1,000 gift card. Click here to claim your prize.",
"Don't forget our meeting at 10 AM tomorrow."]
new_messages_tfidf = vectorizer.transform(new_messages)
predictions = classifier.predict(new_messages_tfidf)

# Display predictions
for msg, pred in zip(new_messages, predictions):
print(f"Message: {msg} | Prediction: {'Spam' if pred == 1 else 'Ham'}")
```

## 6.2 SCREENSHOTS

| | Category | Message |
|---|---|---|
| 1 | ham | ine there got amore wat... |
| 2 | ham | Ok lar... Joking wif u oni... |
| 3 | spam | ply 08452810075over18's |
| 4 | ham | r... U c already then say... |
| 5 | ham | lives around here though |
| 6 | spam | chgs to send, £1.50 to rcv |
| 7 | ham | treat me like aids patent. |
| 8 | ham | py your friends Callertune |
| 9 | spam | L341. Valid 12 hours only. |
| 10 | spam | o FREE on 08002986030 |
| 11 | ham | ? I've cried enough today. |
| 12 | spam | dCs apply Reply HL 4 info |
| 13 | spam | OX 4403LDNW1A7RW18 |
| 14 | ham | ind a blessing at all times. |
| 15 | ham | )N SUNDAY WITH WILL!! |
| 16 | spam | om?n=QJKGIGHJJGCBL |
| 17 | ham | Oh k...i'm watching here:) |
| 18 | ham | aughty make until i v wet. |
| 19 | ham | That□s the way its gota b |
| 20 | spam | 3OXox36504W45WQ 16+ |
| 21 | ham | how you spell his name? |
| 22 | ham | months ha ha only joking |
| 23 | ham | when is da stock comin... |

Figure 6.1: Input given through CSV

```
Accuracy: 0.9808612440191388

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1448
           1       0.99      0.87      0.92       224

    accuracy                           0.98      1672
   macro avg       0.98      0.93      0.96      1672
weighted avg       0.98      0.98      0.98      1672

Message: Congratulations! You've won a $1,000 gift card. Click here to claim your prize. | Prediction: Spam
Message: Don't forget our meeting at 10 AM tomorrow. | Prediction: Ham
```

Figure 6.2: Output

# CHAPTER 7

# REFERENCES


1. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron, 2019 (2nd Edition), O'Reilly Media.
2. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili, 2019 (3rd Edition), Packt Publishing.
3. "Pattern Recognition and Machine Learning" by Christopher M. Bishop, 2006, Springer.
4. "Machine Learning Yearning" by Andrew Ng, 2018, Deeplearning.ai.
5. "Introduction to Machine Learning with Python" by Andreas C. Müller and Sarah Guido, 2016, O'Reilly Media.
6. "Data Science for Business" by Foster Provost and Tom Fawcett, 2013, O'Reilly Media.
7. "Data Mining: Concepts and Techniques" by Jiawei Han, Micheline Kamber, and Jian Pei, 2011 (3rd Edition), Elsevier.
8. "Spam Detection Using Machine Learning" by Dinesh Samuel Christoper and K. S. Rajasekaran, 2018, Wiley.