**RV UNIVERSITY**

**School of Computer Science and Engineering**

**Bengaluru – 560059**



**Machine Learning for Cybersecurity**

**VII Semester B.Sc/B.Tech (HONS.)**

|  |  |
| --- | --- |
| **Name** | **Name** |
| **USN** | **USN** |

### 1. Introduction

Spam email is a prevalent security concern in digital communication, often containing phishing links, scams, and malicious attachments. Automated spam detection is crucial for security and user experience.

In this project, we designed a machine learning spam email classifier. The process includes raw email data handling, text preprocessing, feature extraction using TF-IDF weighted n-grams, and training models such as Support Vector Machine, Naïve Bayes, and Logistic Regression. The trained model was integrated into a web application for near real-time spam detection.

### 2. Tools Used

**Programming Language:** Python

**Data Handling:** pandas, numpy

**Text Processing:** nltk, BeautifulSoup, re, string (tokenization, stopword removal, stemming, lemmatization)

**Feature Extraction:** scikit-learn – TfidfVectorizer (1–3 n-grams)

**Machine Learning Models:**

* LinearSVC (Support Vector Machine)
* MultinomialNB (Naïve Bayes)
* LogisticRegression
* (Optional) RandomForestClassifier

**Model Persistence:** joblib (save trained models and vectorizers)

**Evaluation Metrics:** Accuracy, Precision, Recall, F1-score, Confusion Matrix

### 3. Preprocessing and Training

#### 3.1 Data Preprocessing

* **Parsing and Cleaning:** Extracted subject and body, removed HTML tags, scripts, styles, extra spaces, and punctuation, and converted text to lowercase.
* **Tokenization & Normalization:** Tokenized words, applied stemming and lemmatization, and removed stopwords.
* **Label Mapping:** Emails labeled as ham (0) or spam (1). Dataset saved as processed\_emails\_cleaned.csv.
* **Feature Extraction:** TF-IDF Vectorizer (1–3 n-grams) captured spam patterns. Vectorizer saved as vectorizer.pkl.
* **Data Splitting & Balancing:** Split dataset 80/20 (train/test) and balanced training set by downsampling spam samples.

#### 3.2 Model Training

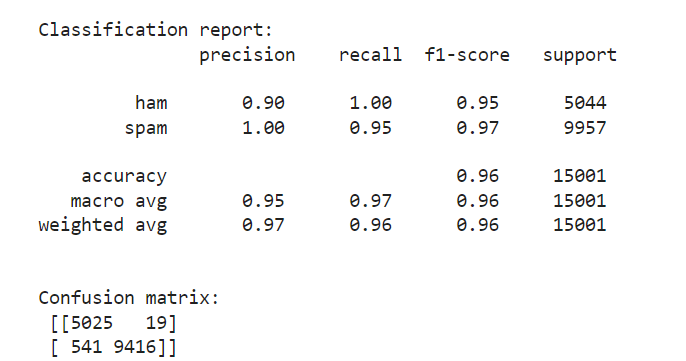
* **Naïve Bayes (MultinomialNB):** Effective with sparse TF-IDF vectors; slightly higher false positives.
* **Evaluation:** Metrics included accuracy, precision, recall, F1-score, and confusion matrices. SVM and Logistic Regression performed best overall.

### 4. Workflow

1. **Data Collection:** Raw emails and labels provided; each file mapped to a class.
2. **Data Preprocessing:** Extracted text, removed noise, tokenized, applied stemming/lemmatization, removed stopwords, and saved cleaned dataset.
3. **Feature Extraction:** TF-IDF vectorization (uni-, bi-, tri-grams) to capture spam-specific patterns; vectorizer persisted.
4. **Dataset Splitting & Balancing:** Train/test split (80/20) and balanced training set.
5. **Model Training:** Trained SVM, Naïve Bayes, and Logistic Regression; stored trained pipelines.
6. **Evaluation:** Measured accuracy, precision, recall, F1-score, and confusion matrices; tested on custom emails.
7. **Deployment:** Integrated model into Flask web app; users upload PDFs, backend predicts SPAM/HAM, and results display clearly.

### 5. Conclusion

Naïve Bayes achieved the highest accuracy (96% on the test set, 93% on custom examples). The web application successfully accepts PDF emails and classifies them as ham or spam.



**6. Screenshots**

