**Email Spam Classifier Using Random Forest & SVM**

**Step 1: Project Overview**

**Objective:**

To build a machine learning model that classifies emails as **Spam** or **Not Spam**, using both **Random Forest** and **Support Vector Machine (SVM)**, and evaluate their performance.

**Dataset:**

* Source: [ Kaggle Spam Dataset]
* Features: Text-based email content (e.g., words, characters, patterns)
* Label: Spam or Not Spam

**Step 2: Task Instructions**

**A. DATA UNDERSTANDING & PREPROCESSING**

**✅ Tasks Performed:**

**i. Importing the Required Libraries**

All necessary Python libraries were imported at the start of the project. These include:

* **pandas** for data manipulation,
* **numpy** for numerical operations,
* **sklearn** for model building, vectorization, evaluation, and preprocessing,
* **re** for regular expressions used in text cleaning,
* **imblearn** for applying the SMOTE balancing technique,
* and others as required.

A requirements.txt file was also created to list all the modules used in the project for easy installation and reproducibility.

**ii. Loaded Dataset**

The dataset was loaded using pandas.read\_csv() into a DataFrame for further processing. The dataset contains email messages and their labels (spam or ham).

**iii. Checked for and Handled Missing Values**

The dataset was checked for missing values using isnull().sum(). Fortunately, there were no missing values; otherwise, appropriate handling such as filling or dropping would be applied.

**iv. Removed Duplicate Records to Avoid Data Leakage**

Duplicate rows were removed using drop\_duplicates() to ensure that the same message does not appear in both training and test sets, which could lead to data leakage and artificially high accuracy.

**v. Converted the 'label' Column from Categorical to Binary**

The target column 'label' was converted from categorical text:

* 'spam' → 1
* 'ham' → 0  
  This was necessary for feeding the data into machine learning models which expect numerical labels.

**vi. Cleaned Text of Dataset Using re Module**

Text cleaning was performed using the re module to:

* Convert all text to lowercase,
* Remove punctuation and special characters,
* Normalize white spaces.  
  This step reduces noise and prepares the text for vectorization.

**vii. Used TF-IDF Vectorization to Convert Text into Numerical Features**

TF-IDF (Term Frequency-Inverse Document Frequency) was used to transform cleaned text into numerical vectors. This helps capture the importance of words relative to all documents and prepares data for model training.

**viii. Saved Vectorizer to tfidf\_vectorizer.pkl File**

The fitted TF-IDF vectorizer was saved using joblib to a file named tfidf\_vectorizer.pkl, so it can be reused later during prediction or deployment without needing to retrain it.

**ix. Split Data into Training and Testing**

The dataset was split into **training** and **testing** sets using train\_test\_split(), typically with an 80-20 ratio. This helps evaluate the model's generalization performance on unseen data.

**x. Balanced Data Using SMOTE Technique**

Since the dataset was **imbalanced**, with far more 'ham' messages than 'spam', **SMOTE (Synthetic Minority Over-sampling Technique)** was applied to balance the classes.

* **Before SMOTE:** Counter({0: 3851, 1: 630})
* **After SMOTE:** Counter({1: 3851, 0: 3851})

Balancing improves the model's ability to correctly identify spam messages and reduces bias toward the majority class.

**B. EXPLORATORY DATA ANALYSIS (EDA)**

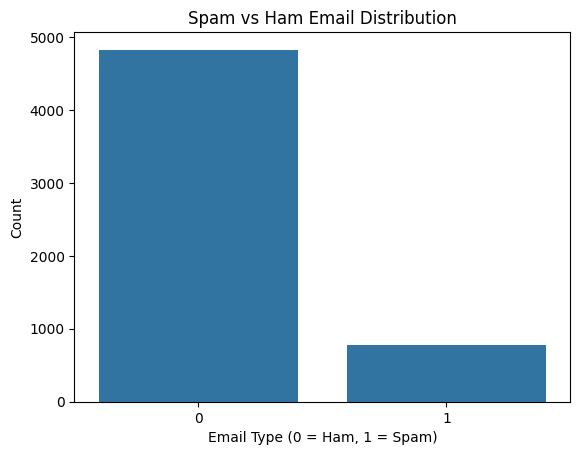
**✅ Tasks Performed:**

**📊 Visualizing Relationships Between Features**

**1. Class Distribution – Bar Plot (Spam vs Ham)**

To understand the distribution of classes in the dataset, a **bar plot** was created showing the number of messages labeled as **ham (0)** and **spam (1)**.

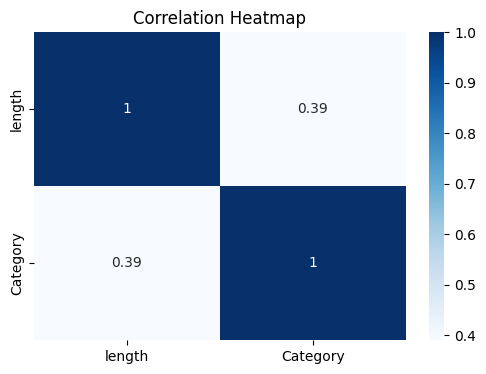
* This is a crucial step in exploratory data analysis (EDA) to assess class **imbalance**, which can significantly affect model performance.
* In this case, the plot revealed that the dataset was **highly imbalanced**, with a majority of the messages labeled as **ham** and a relatively small portion as **spam**.
* Addressing this imbalance is important to avoid a model that is biased toward the majority class. This observation later justified the use of the **SMOTE** technique for balancing the data.



**2. Correlation Heatmap (Message Length vs Label)**

Although the dataset primarily consists of textual data, we engineered a simple numerical feature: **message length**, i.e., the number of characters in each email.

* A **correlation heatmap** was generated to visualize the relationship between this feature and the target label (spam or ham).
* This helped to identify whether spam messages tend to be longer or shorter than ham messages.
* The heatmap showed the **Pearson correlation coefficient**, where values close to +1 or -1 indicate strong relationships.
* While message length alone is not enough for classification, such features can still provide useful **additional signals** for the model.



**C. FEATURE SELECTION**

**✅ Tasks Performed:**

* TF-IDF automatically assigns weights to words based on importance
* Analyzed **feature importance** in Random Forest model

**D. MODEL BUILDING & EVALUATION**

**Models Trained:**

* **Random Forest Classifier**
* **Support Vector Machine (SVM)**

**Evaluation Metrics Used:**

* Accuracy
* Precision
* Recall
* F1-Score
* Confusion Matrix

**Results Summary:**

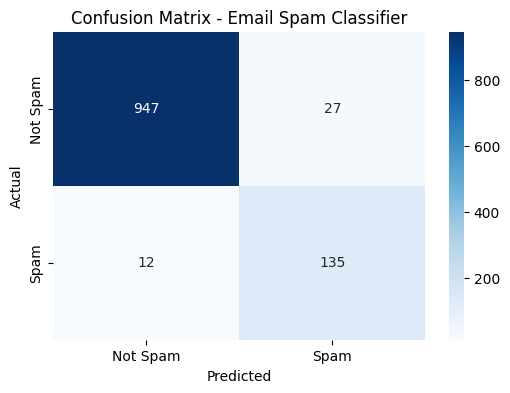
1. **SVM**

Accuracy: 96.5210

**CLASSIFICATION REPORT TABLE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Metric | Class 0 (Ham) | Class 1 (Spam) | Accuracy | Macro Avg | Weighted Avg |
| Precision | 0.99 | 0.83 | 0.97 | 0.91 | 0.97 |
| Recall | 0.97 | 0.92 |  | 0.95 | 0.97 |
| F1-Score | 0.98 | 0.87 |  | 0.93 | 0.97 |
| Support | 974 | 147 | 1121 | 1121 | 1121 |

**CONFUSION MATRIX**



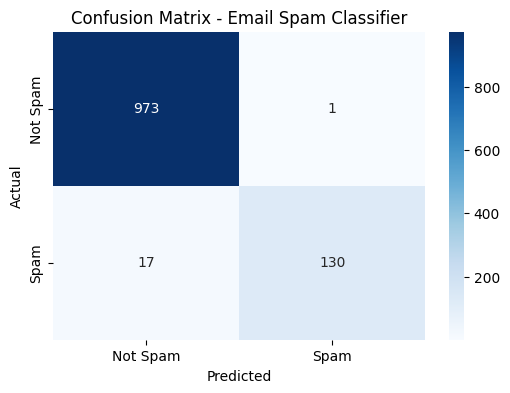
**ii- Random Forest**

**Accuracy: 98.3943**

**CLASSIFICATION REPORT TABLE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Metric | Class 0 (Ham) | Class 1 (Spam) | Accuracy | Macro Avg | Weighted Avg |
| Precision | 0.98 | 0.99 | 0.98 | 0.99 | 0.98 |
| Recall | 1.00 | 0.88 |  | 0.94 | 0.98 |
| F1-Score | 0.99 | 0.94 |  | 0.96 | 0.98 |
| Support | 974 | 147 | 1121 | 1121 | 1121 |

**CONFUSION MATRIX**

****

**E. DEPLOYMENT**

* A simple **Streamlit app** is deployed to allow real-time spam detection from user input.

**📂 Deliverables**

* Jupyter Notebook: Email\_Spam\_Classifier.ipynb
* Source Code Repo: [GitHub Repository](https://github.com/sohaibafzal165/Email-Spam-Classifier-using-Random-Forest-SVM)
* This Brief Report
* (Optional) Video Presentation
* (Optional) Streamlit App Link