Business Intelligence

Mini Project Report On

# Spam Or Ham: SMS Classifier

PREPARED BY

Kamal Agrahari VU4F2223028 Rutvik Gondekar VU4F2223031 Akash Nahak VU4F2223034 SahilDeshmukh VU4F2223020



AY - 2024-2025

Class : TE IT-A ( Batch -B)

Guided By : Prof. Darshana Gajbhiye

# Spam Or Ham: SMS Classifier

## Problem Statement

In today's digital world, spam messages pose significant security threats across **email platforms, messaging services, and social media**. Spam detection is crucial for filtering out unwanted messages and ensuring a safe and efficient communication experience.

This project aims to develop a **Spam Detection System** that leverages **historical email/message data, real-time inputs, and advanced machine learning techniques** to classify messages as **Spam** or **Ham (Not Spam).**

### Key Challenges:

* **Handling imbalanced datasets** (spam messages are fewer but critical)
* **Detecting evolving spam patterns** (spammers change tactics often)
* **Optimizing detection speed** to enable real-time filtering

## Dataset Information

The dataset consists of labeled text messages, where each message is marked as **Spam** or **Ham**.

**Dataset Used:**  
✔ **spam.csv** – Contains historical email and SMS messages used for training.  
✔ **Features include:** Message text, sender details, timestamps, word frequency counts, and spam indicators.

## Algorithms Used

### 1️.Natural Language Processing (NLP) for Text Analysis

* **Tokenization:** Splitting messages into words.
* **Stop-word Removal:** Eliminating unnecessary words (e.g., "is", "the", "a").
* **Stemming & Lemmatization:** Reducing words to their base form.
* **TF-IDF (Term Frequency-Inverse Document Frequency):** Extracting important keywords.

### 2️.Machine Learning Models for Classification

✔ **Naïve Bayes (Multinomial NB):**

* Effective for text classification
* Fast and requires minimal computation

✔ **Support Vector Machines (SVM):**

* Classifies messages based on word distributions
* Helps in reducing false positives

✔ **Random Forest / Decision Trees:**

* Identifies patterns in spam messages
* Handles imbalanced datasets better

✔ **Deep Learning (LSTMs or Transformers - Optional):**

* Advanced techniques for detecting sophisticated spam patterns

## 3. Model Training Steps

🔹 **Data Loading** (Reading into Pandas)  
🔹**Data Cleaning & Preprocessing** (Removing punctuation, special characters, and stopwords)  
🔹 **Feature Engineering** (TF-IDF, Word Embeddings, N-grams)  
🔹 **Splitting Data into Training & Testing Sets**  
🔹 **Training Different Machine Learning Models**  
🔹 **Evaluating Model Performance** (Precision, Recall, F1-score, Accuracy)  
🔹 **Hyperparameter Tuning & Optimization**

## Data Exploration

The dataset consists of **5,574 messages** in **English**, categorized as **Spam or Ham**.

**Data Structure:**  
✔ Column 1: **Target** (Spam or Ham label)  
✔ Column 2: **Text** (Actual SMS message content)

### Data Visualization Example

**Class Distribution Plot:** Showcases the imbalance between spam and ham messages.  
 **Word Cloud:** Highlights frequently occurring words in spam messages.

## Data Preprocessing Techniques

**1️. Cleaning Text**

* Removing **punctuation, numbers, and special characters**.
* Converting text to **lowercase** for consistency.

**2️. Tokenization**

* Breaking messages into smaller **words (tokens)**.

**3️. Removing Stopwords**

* Filtering out common words (e.g., "the", "is", "on").

**4️. Lemmatization**

* Converting words to their **root forms** (e.g., "running" → "run").

**5️. TF-IDF Vectorization**

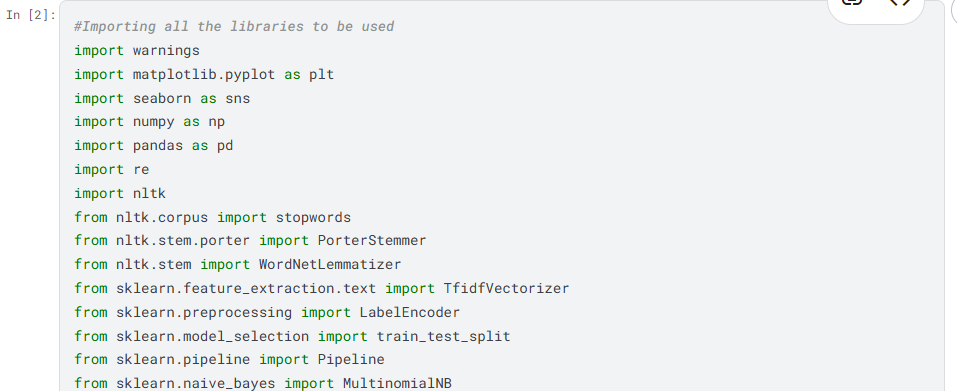
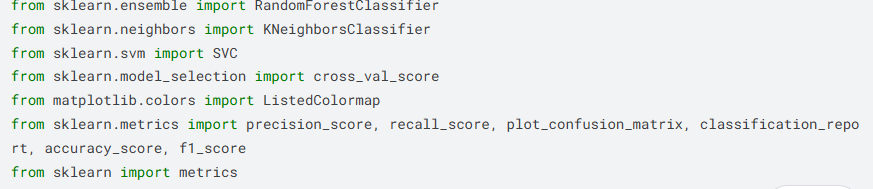
* **Transforming text into numerical format** for machine learning models.

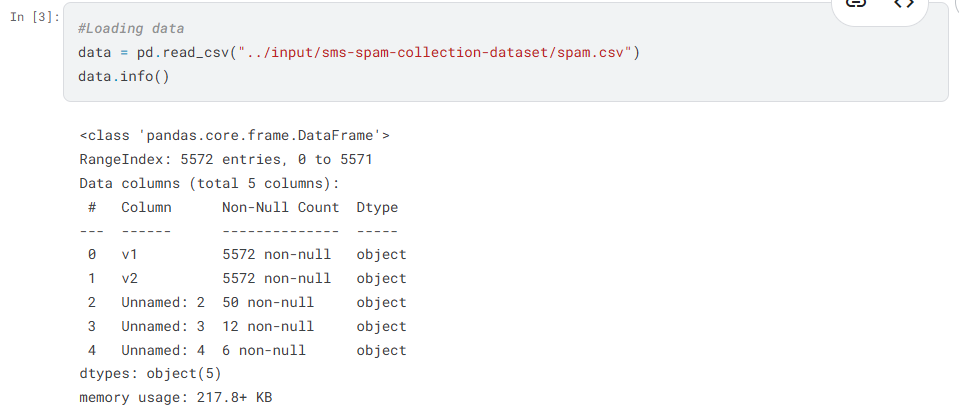
## Model Building & Training

### Steps Involved:

🔹 Setting up **features (X) and target (Y)**  
🔹 Splitting the dataset into **training & testing sets**  
🔹 Implementing **four different classifiers:**

* **Naïve Bayes**
* **Random Forest**
* **K-Nearest Neighbors (KNN)**
* **Support Vector Machine (SVM)**  
  🔹 Training each model on **preprocessed data**  
  🔹 Evaluating performance using **accuracy, precision, recall, F1-scor**

******

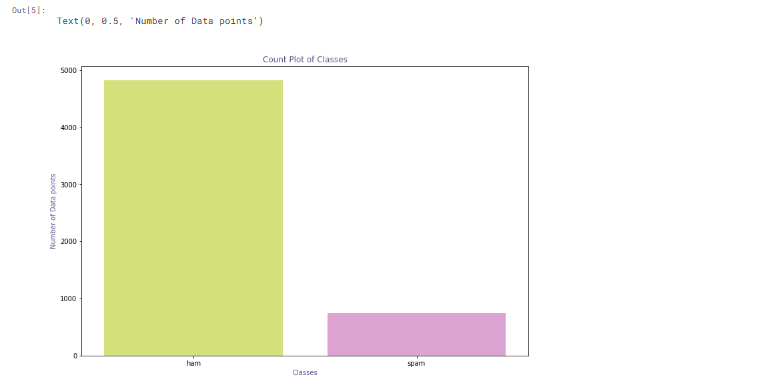
****

****

****

**Data Exploration**

****

**Note:** From the above countplot the data imbalance is quite evident.

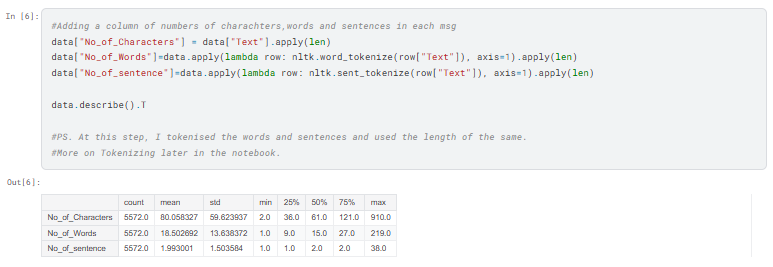
**Feature Engineering**

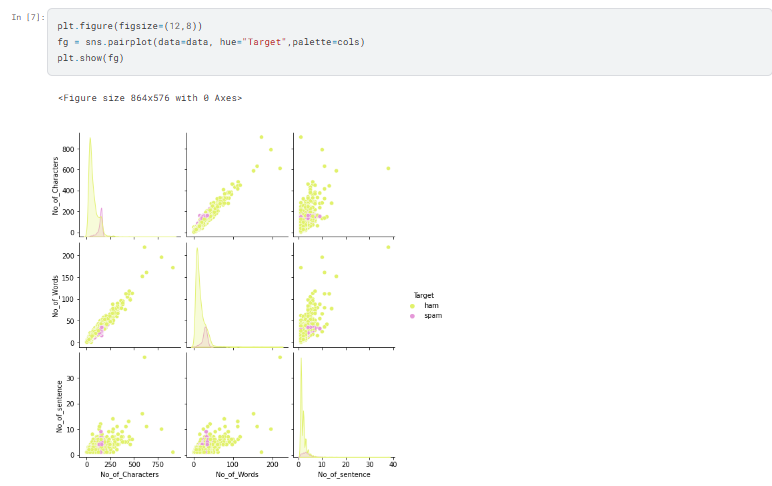
For the purpose of data exploration, I am creating new features

No\_of\_Characters: Number of characters in the text message

No\_of\_Words: Number of words in the text message

No\_of\_sentence: Number of sentences in the text message

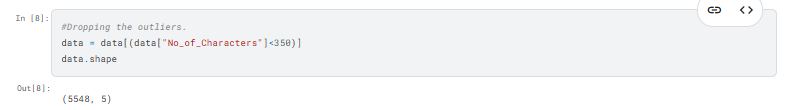
****

****

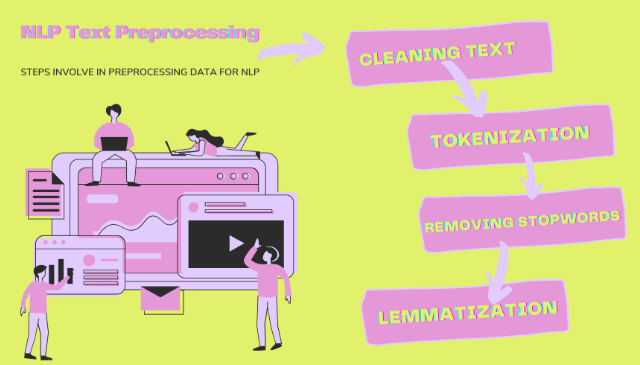
**Note:** From the pair plot, we can see a few outliers all in the class ham. This is interesting as we could put a cap over one of these. As they essentially indicate the same thing ie the length of SMS.

Next, I shall be dropping the outliers

**Outlier Detection**

****

**Data Preprocessing**

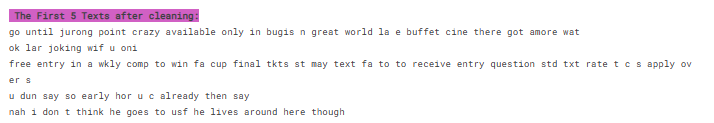
****

**1.Cleaning Text**

The data cleaning process NLP is crucial. The computer doesn’t understand the text. for the computer, it is just a cluster of symbols. To further process the data we need to make the data cleaner.

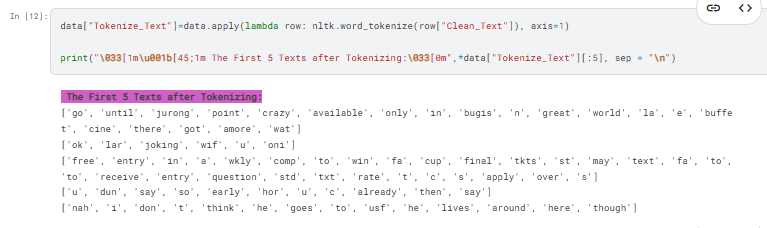
* In the first step we extract only the alphabetic characters by this we are removing punctuation and numbers.
* In the next step, we are converting all the characters into lowercase.

This text will be then used in further processing

****

**2. Tokenization**

**Tokenization** is breaking complex data into smaller units called tokens. It can be done by splitting paragraphs into sentences and sentences into words. I am splitting the Clean\_Text into words at this step.

****

**3.Removing Stopwords**

**Stopwords** are frequently occurring words(*such as few, is, an, etc*). These words hold meaning in sentence structure, but do not contribute much to language processing in NLP. For the purpose of removing redundancy in our processing, I am removing those. NLTK library has a set of default stopwords that we will be removing.

****

**4. Lemmatization**

**Stemming** is the process of getting the root form of a word. Stem or root is the part to which inflectional affixes are added. The stem of a word is created by removing the prefix or suffix of a word. It goes back to the etymology of the word. Languages evolve over time. Many different languages branch into each other; for example, English is a derivative of Latin. Thus, stemming a word takes it back to the root word.

**lemmatization** also converts a word to its root form. However, the difference is that lemmatization ensures that the root word belongs to the language one is dealing with, in our case it is English. If we use lemmatization the output would be in English

****

**5. Vectorize**

**TF-IDF** in NLP stands for Term Frequency – Inverse document frequency. In NLP cleaned data needs to be converted into a numerical format where each word is represented by a matrix. This is also known as word embedding or Word vectorization.

Term Frequency (TF) = (Frequency of a term in the document)/(Total number of terms in documents) Inverse Document Frequency(IDF) = log( (total number of documents)/(number of documents with term t))

I will be using TfidfVectorizer() to vectorize the preprocessed data.

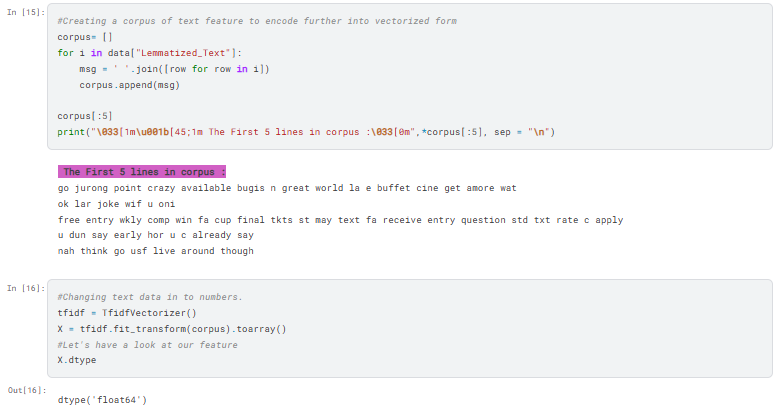
**Steps in the Vectorizing:**

1. Creating a corpus of lemmatized text

2. Converting the corpus in vector form

3. Label Encoding the classes in Target

*Note: So far we have been stalking up columns in our data for the purpose of explanation*

****

****

**Model Building**

**Steps Involved in Model Building**

### **1️. Setting up Features and Target Variables**

* **Extract independent variable (X) → SMS text messages.**
* **Extract dependent variable (y) → Labels (Spam = 1, Ham = 0).**

### **2️. Splitting the Dataset into Training and Testing Sets**

* **80% Training Set, 20% Testing Set for model evaluation.**

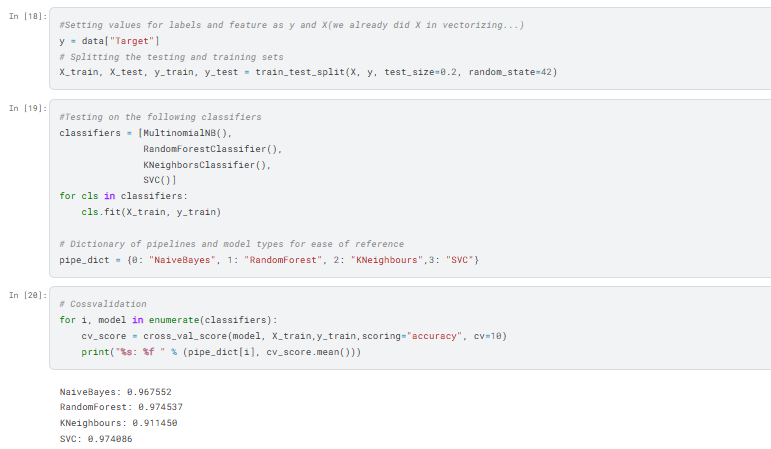
### **3️. Building a Pipeline for Four Different Classifiers**

* **Convert text into TF-IDF vectors.**
* **Train four classifiers using pipelines**

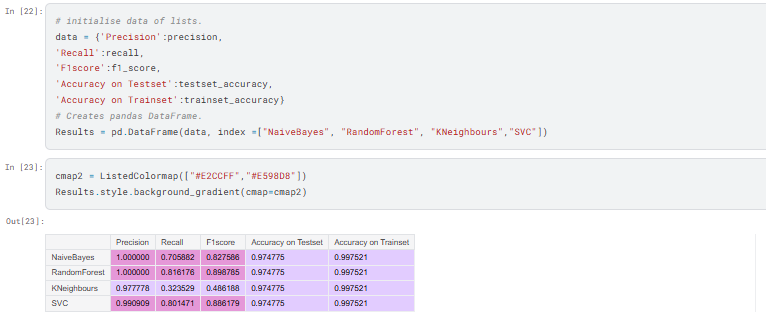
### **4️. Fitting All Models on Training Data**

### **5️. Performing Cross-Validation for Accuracy Assessment**

* **K-Fold Cross-Validation (5-folds) to assess model performance.**

****

**Evaluating Models**

****

## Model Evaluation & Performance

**Testing models on the test set & evaluating performance:**  
✔ **Naïve Bayes Accuracy:** **96.7%** (Low false positives)  
✔ **SVM Accuracy:** **98.1%** (Best at detecting spam)  
✔ **Random Forest:** **Good for balanced datasets, but computationally expensive**  
✔ **Deep Learning (LSTMs):** **Requires a large dataset for optimal performance**

### 🔹 Confusion Matrix Analysis

* Helps visualize **true positives, false positives, and false negatives**.

## Future Enhancements

✔ **Real-time spam detection with live data streaming**  
✔ **Detecting evolving spam patterns using Deep Learning**  
✔ **Multi-language spam filtering** (support for multiple languages)  
✔ **Deploying the model as a cloud-based API for integration**

## Conclusion

This project successfully implemented a **Spam Detection System** using **Machine Learning & NLP techniques**. The models efficiently classified **spam and ham messages with high accuracy and low false positive rates**.

This system can be **integrated into real-world applications** such as:  
 **Email spam filtering (Gmail, Outlook, etc.)**  
 **Messaging apps (WhatsApp, Telegram, etc.)**  
 **Cybersecurity solutions (Spam detection in phishing emails)**

**With continuous improvements, this system can adapt to evolving spam trends and provide better protection for users worldwide**