SMS Span Classifier

CONTENTS

Introduction to SMS Spam Classifier Using Machine Learning

In today's fast-moving digital world, **SMS (Short Message Service)** remains an important way for people and businesses to communicate. However, with the convenience of SMS comes an annoying problem — **spam messages**. These are unwanted, irrelevant, or even dangerous texts sent to your phone. To tackle this, **SMS Spam Classifiers** powered by **Machine Learning (ML)** have become very important.

Let's dive into what an SMS Spam Classifier is, how it works, and why it is so useful.

What is Classification?

Before we jump into SMS spam classification, let's first understand what **classification** means.
Classification is a type of **supervised machine learning** task where the goal is to **sort data into categories or groups**. For example, an email can be classified as "Important" or "Not Important". Similarly, in the case of SMS, a message can be classified as either "Spam" or "Ham" (Ham means not spam).

In simple words, classification teaches a computer to recognize patterns in data and **make decisions** about which category a new item belongs to.

What is SMS Spam Classification?

SMS Spam Classification is the process of using machine learning models to **detect** and **separate spam messages from normal (ham) messages**.
A classifier is trained on a large number of labeled SMS messages — some marked as "spam" and some as "ham".
After training, the model can predict whether a new incoming SMS is spam or not.

The main aim is to **protect users** from fraud, scams, and unwanted ads, improving their SMS experience.

Uses of SMS Spam Classification

SMS spam classification has several important uses:

- **Security**: Blocks scam or phishing messages that could steal sensitive information.
- **Better User Experience**: Keeps the user's inbox clean and organized.
- **Business Protection**: Prevents important business messages from getting lost among spam.
- **Automation**: Reduces the need for manual filtering of unwanted messages.
- **Cost Saving**: Helps organizations save money by protecting communication channels.

In short, it makes communication **safer**, **faster**, and **more reliable**.

Type of Classification Problem: Binary or Multi-class?

SMS Spam Classification is usually a **binary classification problem**.
This means there are only **two classes**:

1. **Spam** — unwanted or harmful message.
2. **Ham** — safe, normal message.

Since the classifier needs to make a choice between two options, it falls under binary classification.
However, in some advanced cases (like detecting different types of spam — e.g., ads, scams, frauds), it can become a **multi-class problem**, but that is less common in basic SMS spam filtering.

Recommended Machine Learning Algorithms for SMS Spam Classification

Several machine learning algorithms work very well for SMS spam detection. Here are some popular and recommended ones:

1. **Naive Bayes Classifier**
   - Very effective for text classification problems.
   - Simple, fast, and works well even with smaller datasets.
2. **Logistic Regression**
   - A powerful algorithm for binary classification.
   - Easy to understand and implement.
3. **Support Vector Machine (SVM)**
   - Very good for separating two classes with a clear margin.
   - Often used when higher accuracy is needed.
4. **Decision Trees and Random Forests**
   - Easy to visualize.
   - Random Forests, being an ensemble of trees, improve accuracy and reduce overfitting.
5. **Deep Learning (LSTM, GRU)**
   - For very large datasets, deep learning models like LSTM (Long Short-Term Memory) can give excellent results.
   - However, they require more computation power and time.

In general, for beginners or small projects, **Naive Bayes** and **Logistic Regression** are great starting points.

SMS Spam Message Classification — Full Project Approach

1. Data Collection

Reference:
The dataset was downloaded from the link provided in the assignment PDF.

Goal:
Before diving into full model development, the first objective was to understand the quality of the dataset.

2. Quick Inference using an Automated ML Framework

Tool Used:
PyCaret — an automated machine learning library that simplifies the entire machine learning lifecycle, including preprocessing, feature engineering, model training, and evaluation.

Purpose:

- Quickly assess the quality of the dataset.
- Understand how various models perform without manually coding everything.
- Detect early signs of data issues (poor models or unrealistic performance like 1.0 accuracy).

Insight Gained:
If the dataset quality were poor, models would either show very low accuracy or suspiciously high (1.0) accuracy. A good dataset would reflect accuracies typically between 0.8 to 0.9 for general classifiers.

Result:

- **Support Vector Machine (SVM) with Linear Kernel** performed outstandingly, achieving an accuracy of approximately **0.98**.
- This confirmed that the dataset was of good quality.

3. Starting the Project from Scratch

After gaining quick insights, the project was restarted from scratch to follow a proper data science workflow.

4. Data Preprocessing

Steps Performed:

- Inspected the shape of the dataset.
- Renamed columns for better clarity and readability.
- Removed unnecessary and redundant columns.
- Handled missing values and duplicate entries to maintain data integrity.

Handling Textual Data:
Since the dataset contained textual data, it was important to manage memory usage

carefully. To prevent errors or crashes during model training, the dataset size was **reduced by approximately 33%** while ensuring no critical information was lost.

Class Balance Check:
Verified the distribution of spam and ham messages using value_counts(). This ensured there was no major imbalance in the target variable.

5. Exploratory Data Analysis (EDA)

Visualization:

- Created **Word Cloud graphs** separately for Spam and Ham messages.
- This helped visually identify important and distinguishing words used in spam vs ham texts.

6. Feature Engineering

Techniques Used:

- Applied **TF-IDF Vectorization** on the 'Description' feature.
- Applied **Bag of Words (Count Vectorizer)** separately on the same feature.

Target Label Mapping:

- Mapped textual labels: **Spam → 0**, **Ham → 1**.

Why Both TF-IDF and Bag of Words?:

- To analyze and compare the performance of over **20+ binary classifiers** on two different feature transformations.
- This enabled a more thorough understanding of how different feature engineering techniques impact model performance.

7. Model Selection and Implementation

First Approach:

- Implemented **20+ binary classifiers** (including ensemble models, naive techniques, tree-based methods, k-Nearest Neighbors, etc.) on the **TF-IDF transformed** dataset.
- Evaluated models using metrics like **accuracy**, **precision**, and **recall**.

Second Approach:

- Repeated the entire process using **Bag of Words transformed** data.
- Again, assessed model performance based on **accuracy**, **precision**, and **recall**.

8. Model Evaluation

Evaluation Metrics:

- **Accuracy**: Overall correctness of the model.
- **Precision**: Correctness of positive (spam) predictions.
- **Recall**: Ability of the model to find all positive (spam) cases.

Objective:

- Compare how different models performed across TF-IDF and Bag of Words features.
- Identify the best-performing classifier among all tried models.

9. Model Download

Saving the Model:

- Exported the best-performing model as a **.pkl** file.
- This allows for seamless reuse and deployment without retraining.

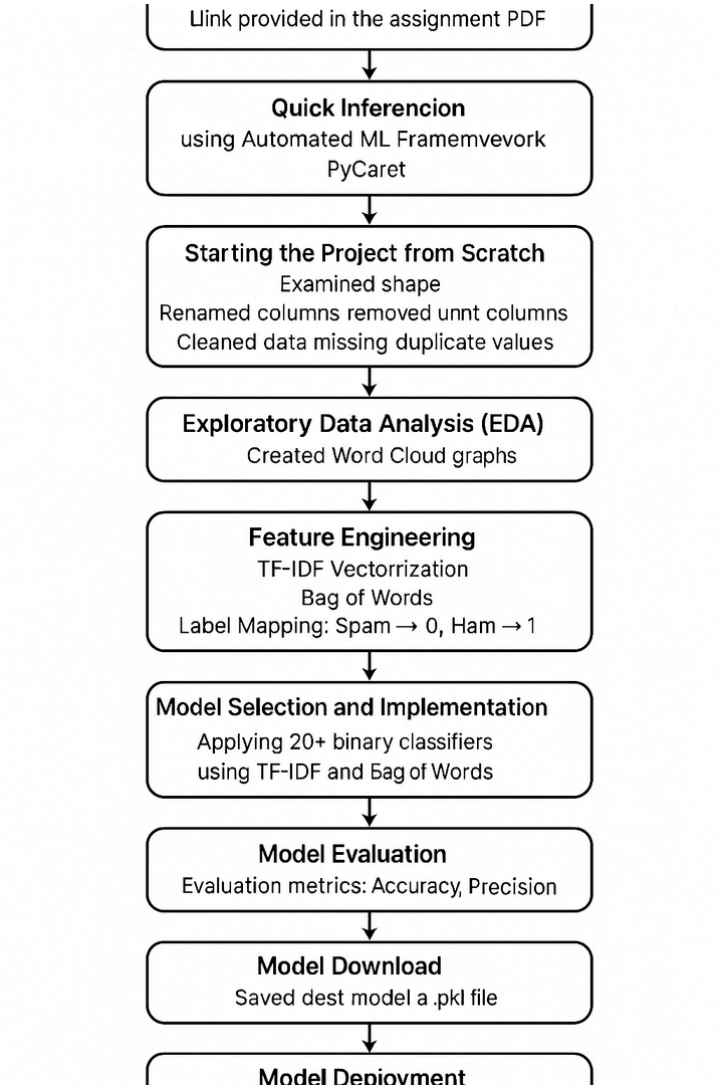
10. Model Deployment

Deployment Platform:

- Deployed the finalized spam classification model on **Hugging Face Spaces**.

Goal:

- Make the model easily accessible for end-users to classify SMS messages into Spam or Ham.



Code Cab Be Accessed Here : SPAM SMS CLASSIFICATION SYSTEM [Kaggle Notebook]

Deployment Files :

SMS-Spam-Classifier/CODEFILEs/deploymentFIles at main · Palyam72/SMS-Spam-Class...

Tech Stack Used For Deployment : FastAPI, HTML5, TAILWIND CSS

Deployment Service Used : Hugging Face Spaces Free Tier, Docker Container