SMS Spam Detection — Project Template

- **Project Title:** SMS Spam Detection Using Machine Learning
- **Your Name:** Deepika KR
- **Institution:** Mahendra College of Engineering
- **Department:** Computer Science and Engineering
- **Date:** October 2025

❖ Table of Contents

1. Project Overview

Spam detection plays a vital role in digital communication systems. With the rapid increase of mobile users, SMS has become a primary medium for exchanging messages. However, many of these messages are unwanted or malicious in nature, posing threats to user privacy and financial security. Detecting such spam messages manually is inefficient, hence automated systems are necessary.

The primary motivation of this project is to enhance user experience and security by automatically identifying and filtering spam messages using machine learning. By training algorithms on historical SMS data, the system learns to differentiate between spam and legitimate (ham) messages based on text patterns.

Scope: The system is designed exclusively for SMS text classification and not for email or social media spam. It handles only two categories: spam and ham. The project covers data collection, preprocessing, model building, evaluation, and basic deployment demonstration.

High-level Description: The model pipeline includes text cleaning, tokenization, vectorization using TF-IDF, training models such as Naive Bayes and Logistic Regression, evaluating them, and finally predicting the label for new SMS input.

2. Objectives & Problem Statement

2.1 Problem Statement

The challenge addressed in this project is to classify SMS messages as spam or ham automatically using machine learning. The messages are typically short and informal, often containing slang or typos, which makes accurate classification difficult. Additionally, spam messages are fewer than normal ones, causing a class imbalance problem that affects model training.

2.2 Objective

- To build an efficient SMS spam detection model using machine learning algorithms.
- To preprocess and clean text data effectively using NLP techniques.
- To evaluate and compare models based on various performance metrics.
- To create a user-friendly system for predicting whether an SMS is spam or ham.

To ensure the system is accurate, fast, and scalable for large datasets

3. Proposed Solution

This project employs a supervised learning approach where a labeled dataset of SMS messages is used to train a classification model. Each message is labeled as 'spam' or 'ham'. The overall

process includes data preprocessing, feature extraction, model training, and evaluation.

Pipeline Steps:

- Data preprocessing: Removing punctuation, converting to lowercase, and tokenizing text.
- Feature extraction: Converting text to numeric vectors using TF-IDF or CountVectorizer.
- Model training: Training algorithms such as Naive Bayes, Logistic Regression, Random Forest, and SVM.
- Model evaluation: Assessing results using accuracy, precision, recall, F1-score, and confusion matrix.
- Deployment: Saving the model and creating a prediction interface (optional web/CLI app).

Justification: Machine learning models are chosen because of their ability to identify complex patterns from text data automatically. Naive Bayes is particularly effective for text classification due to its simplicity and speed.

4. Features

4.1 Functional Features

- Accepts SMS text input from the user.
- Classifies input messages as Spam or Ham.
- Displays probability/confidence score.
- Performs batch classification for multiple inputs.
- Provides visualizations such as confusion matrix or word importance.

4.2 Non-Functional Features

- Fast inference speed with low computational cost.
- High accuracy and reliability.
- User-friendly interface with simple interaction.
- Scalable to handle large datasets efficiently.

- Maintainable for future updates and improvements.

5. Technologies & Tools

- Programming Language: Python
- Libraries: Pandas, NumPy, Scikit-learn, NLTK, Matplotlib, Seaborn
- Feature Extraction: CountVectorizer, TF-IDF Vectorizer
- Models: Naive Bayes, Logistic Regression, Random Forest, Support Vector Machine
- Model Persistence: pickle, joblib
- Interface: Streamlit or Flask for user interaction
- Development Environment: Jupyter Notebook / VS Code

6. System Architecture

The architecture consists of several modules that process data sequentially from input to output. The structure ensures modularity, making it easy to debug and maintain.

Modules: Input → Preprocessing → Feature Extraction → Model Training → Evaluation → Output Interface

Data Flow: During training, raw text is cleaned, transformed into numeric vectors, and passed into machine learning models. During inference, new messages follow the same preprocessing pipeline, and the trained model predicts spam or ham.

7. Implementation Steps

- 1. Data acquisition and loading.
- 2. Exploratory Data Analysis (EDA) to understand spam/ham distribution.
- 3. Text preprocessing (lowercasing, removing punctuation, stopword removal).
- 4. Tokenization and lemmatization using NLTK.
- 5. Feature extraction using TF-IDF Vectorizer.

- 6. Splitting the dataset into training and testing sets (80:20 ratio).
- 7. Model training using Naive Bayes as the baseline model.
- 8. Testing additional models such as Logistic Regression and Random Forest.
- 9. Evaluating performance metrics: accuracy, precision, recall, F1-score.
- 10. Plotting confusion matrix and ROC curve for visualization.
- 11. Performing hyperparameter tuning using GridSearchCV.
- 12. Saving best model using pickle/joblib.
- 13. Developing a simple web UI using Streamlit for predictions.
- 14. Testing system on unseen data and documenting results.

8. Output / Screenshots

```
Sample Input: 'Congratulations! You have won a $1000 Walmart gift card. Click here to claim your pri
Prediction: Spam

Sample Input: 'Hey, I will be late to class today.'
Prediction: Ham
```

9. Advantages

- Automates spam detection effectively, saving time and effort.
- Achieves high accuracy using optimized ML algorithms.
- Lightweight and efficient, suitable for real-time use.
- Can be retrained easily with updated datasets.
  Interpretability through key word analysis or feature importance

10. Future Enhancements

- Incorporate deep learning models such as LSTM and BERT for better accuracy.
- Support multiple languages for wider usability.

- Implement online learning to update models dynamically.
- Integrate with mobile messaging apps for real-time filtering.
- Deploy the model as a cloud API for large-scale use.

11. Conclusion

In this project, we developed an SMS Spam Detection system using machine learning techniques. The system successfully classifies messages as spam or ham with high accuracy. It leverages NLP methods to preprocess text data and employs supervised learning models for prediction. The experimental results demonstrate that models like Naive Bayes and Logistic Regression perform efficiently for short text classification.

The project not only showcases the power of machine learning in text analytics but also contributes to user security by automatically filtering out malicious or irrelevant messages. Future improvements can focus on deep learning approaches and real-time implementation.