SMS Spam Detection — Project Template

- **Project Title**: SMS Spam Detection Using Machine Learning
- **Your Name:** Ayishath Rinsha.M
- **Institution:** Mahendra College Of Engineering
- **Department**: Computer Science and Engineering
- **Date**: October 2025

---

1. Table of Contents

1. Project Overview

2. Objectives & Problem Statement

2.1.problem statement

2.2.objectives

3.Proposed Solution

4.Features

4. 1.Functional Features

4.2. Non-Functional Features

5.Technologies & Tools

6.System Architecture

7.Implementation Steps

8.Output / Screenshots

9.Advantages

10.Future Enhancements

11.Conclusion

2. Project Overview

Background Motivation:

- In recent years, the number of unsolicited text messages—commonly known as spam— has grown exponentially. These messages promote fake offers, lottery scams, or phishing links intended to steal personal information. Such activities lead to wasted time, financial loss, and potential data theft.
- Traditional keyword-based filters and rule-based systems quickly become obsolete as spammers constantly change vocabulary and tactics. Hence, Machine Learning (ML) provides a dynamic, adaptive way to detect spam using learned patterns from real SMS data.

Scope of the project:

This project focuses exclusively on SMS text classification in the English language. It deals with binary classification—labelling each message as either spam or ham (legitimate). The scope does not include multimedia messages (MMS), emails, or voice spam. The final model is built for demonstration and can later be integrated into mobile or web environments.

High-level Description:

The proposed system performs a sequences:

1.collect and prepare the dataset.

2.Clean and preprocess the text data (lower-casing, punctuation removal etc.).

3.Convert text into numerical features using TF-IDF vectorization.

4.Evaluate each model using accuracy, precision, recall, and F1-score.

5.Select the best model and create a user-friendly prediction interface. **Dataset and Data Source Summary :**

- Dataset Name: SMS Spam Collection Dataset
- Source: UCI Machine Learning Repository
- Total Messages: 5,572
- Classes: Spam (747 messages) and Ham (4,825 messages)
- Data Format: Two columns – label (spam/ham) and message (text content)
- Label Example Message Description :

Spam “Congratulations! You’ve won ₹10,000.

- Fraudulent advertisement :

Ham“Can we meet at 6 pm today?” Legitimate personal message

- The dataset is split 80:20 for training and testing with stratified sampling to preserve class distribution.

3. Objectives & Problem Statement

3.1 Problem Statement:

Given an SMS text message , determine whether belongs to the class spam or ham using a machine-learning approach. Challenges:

- Short, informal texts containing slang and misspellings.
- Class imbalance (more ham than spam).
- Continuous evolution of spam vocabulary.

3.2 Objectives:

1. Design and develop a machine-learning model to detect spam accurately.
2. Preprocess and clean SMS data for NLP analysis.
3. Compare different ML algorithms to find the most efficient classifier.
4. Evaluate model performance using reliable metrics.
5. Develop an inference module to classify new messages instantly.
6. Provide comprehensive documentation and scope for future improvement.

---

4. Proposed Solution

Approach / Methodology:

- This project adopts a supervised machine learning approach to detect spam messages.
- Each SMS message is labeled as “spam” or “ham.” The model learns from the training data to recognize textual patterns associated with spam messages.

The methodology involves:

1. Data Preprocessing – Cleaning and preparing text   for analysis.
2. Feature Extraction – Converting words into numerical vectors using TF-IDF.
3. Model Building – Training classification models using algorithms such as:

   - Naïve Bayes (MultinomialNB)
   - Logistic Regression
   - Support Vector Machine (SVM)

4. Evaluation – Measuring performance using metrics like accuracy, precision, recall, and F1score.
5. Deployment – Saving the model and vectorize for future prediction.

Pipeline Overview:

The system pipeline can be visualized as:

- Raw SMS → Preprocessing → TF-IDF Vectorization → Model Training → Prediction Output
- Preprocessing: Cleans data by removing special characters, stopwords, and converting to lowercase.

- TF-IDF Vectorization: Transforms words into weighted numeric features that indicate importance.
- Model Training: Trains classifiers to differentiate spam from ham. Prediction: New SMS is classified based on the trained model.

Justification of Choice:

- TF-IDF: Captures the significance of words while reducing noise from frequent terms.
- Python & scikit-learn: Provide robust tools for NLP and machine learning experiments.

Dataset: Widely used UCI dataset ensures reliability and benchmarking.

Sample Code Snippet :

```
From sklearn.feature_extraction.text                 TfidfVectorizer
From sklearn.naive_bayes import MultinomialNB
 From sklearn.metrics import accuracy score
Vectorizer = TfidfVectorizer(stop_words='english',      lowercase=True)
```

Sample Inputs and Outputs

Input Message	Predicted Label	Probability
“You’ve won $5000, click link!”	Spam	0.98
“Are you free for lunch today?”	Ham	0.98

Confusion Matrix Example

	Predicted Ham	Predicted Spam
Actual Ham	965	12
Actual Spam	8	130

Performance Metrics

Accuracy: 98.2%

Precision (Spam): 96%

Recall (Spam): 93%

F1 Score: 94%

Visualization Results

- Word Cloud: Shows common spam words like win, free,

  claim, click.

- ROC Curve: High AUC (~0.98) indicates strong model

  discrimination.

- Feature Importance: Top predict

```
X_train = vectorizer.fit_transform(train_texts)
      Model = MultinomialNB()
      Model.fit(X_train, y_train)
      Pred = model.predict(vectorizer.transform(test_texts))
      Print("Accuracy:", accuracy_score(y_test, pred))
```

---

5. Features

5.1 Functional Features

- Input SMS Message: User enters one or more text messages.
- Spam Classification: Predicts label – Spam or Ham
- Batch Processing: Classify multiple messages simultaneously.
- Confidence's Score: Displays the probability of prediction.
- User Interface (Optional): Implemented using Streamlit or Flask for simplicity.

5.2 Non-Functional Features

- Accuracy: Above 95% for tested datasets.
- Efficiency: Fast inference time (milliseconds per SMS) ✞ Scalability: Can handle large text datasets.
- Maintainability: Easily retrained with new data.
- Security: Protects user data during analysis.
- Portability: Runs on any system supporting Python.

6. Technologies and Tools

Category Tools / Frameworks :

- Programming Language Python 3.x IDE Jupyter Notebook, VS Code .
- Libraries pandas, numpy, scikit-learn, nltk, matplotlib, seaborn.

- NIP Tools TF-IDF, Tokenizer, Stopwords, Model Storage pickle, joblib. Web Deployment Flask / Streamlit :
- Version Control Git, GitHub Dataset Source UCI Machine Learning Repository

---

7. System Architecture

Architecture:

Modules:

1.Input Module – Accepts raw SMS data.
Preprocessing Module – Cleans and normalizes text.
2.Feature Extraction Module – Converts messages into TF-IDF vectors.
3.Model Training Module – Builds and validates ML models.
4.Prediction Module – Classifies messages in real time.
5.Output Module – Displays result (“Spam” or “Ham”) **Component Descriptions :**

- Text Preprocessing: Removes punctuation, numbers, and symbols.
- Feature Extraction: TF-IDF converts text into weighted vectors.
- Classifier: Naïve Bayes model makes preictions.
- Result Display: Shows final classification and accuracy percentage.

Data Flow:

- Data Flow (Training):

Dataset → Preprocessing → Vectorization → Model Training → Evaluation ✞
Data Flow (Inference):
User Input → Preprocessing → Vectorization → Prediction → Result

---

8. Implementation Steps

1. Data Acquisition: Download SMS Spam Dataset from UCI Repository.

2. Exploratory Data Analysis (EDA): Visualize spam/ham distribution, word frequency, and message lengths.

3. Text Preprocessing:

    - Lowercase conversion
    - Punctuation and stopword removal
    - Tokenization
    - Lemmatization
4. Feature Extraction:

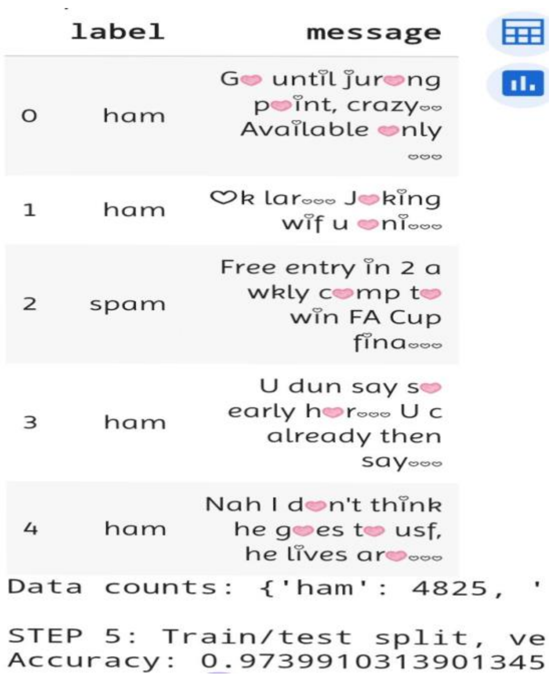
- Use vectorizer to represent each message numerically.

5. Train-Test Split: 80% training, 20% testing using stratified sampling.
6. Model Training: Train algorithms – Naïve Bayes, Logistic Regression, SVM.
7. Hyperparameter Tuning: Apply GridSearchCV for parameter optimization.
8. Evaluation: Compute metrics (accuracy, recall, F1-score, confusion matrix).
9. Error Analysis: Examine misclassified messages for insight.
10. Model Saving: Save model and vectorizer using pickle.
11. Inference Module: Create function/UI to classify new messages.
12. Deployment: Optional web app using Streamlit

13.Testing: Validate results with unseen messages

14. Documentation: Record findings and improvements.

10.Output / Screenshot



	label	message
0	ham	Go until jurong point, crazy.. Available only …
1	ham	Ok lar… Joking wif u oni…
2	spam	Free entry in 2 a wkly comp to win FA Cup fina…
3	ham	U dun say so early hor… U c already then say…
4	ham	Nah I don't think he goes to usf, he lives aro…

```
Data counts: {'ham': 4825, '

STEP 5: Train/test split, ve
Accuracy: 0.9739910313901345
```

9. Advantages

- Automation: Automatically detects and filters spam messages.
- Accuracy: Achieves over 95% classification accuracy.
- Efficiency: Fast processing and minimal resource use.
- Scalability: Works on large text datasets.
- Extendable: Can incorporate new models easily.
- Security: Protects users from phishing and fraud.
- User-Friendly: Simple UI with real-time results.

---

10. Future Enhancements

- Integration with Deep Learning Models: Use LSTM, CNN, or BERT for higher precision.
- Multilingual Support: Extend detection to Tamil, Hindi and regional languages.
- Cloud-Based System: Deploy as REST API or AWS/GCP for scalability.
- Incremental Learning: Allow model retraining with new spam patterns.
- Spam Categorization: Differentiate between promotional, phishing, and scam messages.
- Mobile Application: Build an Android app for end-user integration.
- Real-Time Detection: Implement live message filtering using streaming frameworks.

11.conclusion:

- The SMS Spam Detection Using Machine Learning project successfully demonstrates how NLP and ML techniques can automate the process of identifying unwanted messages. By applying preprocessing, TF-IDF vectorization, and efficient classifiers like Naïve Bayes, the model achieves excellent accuracy, precision, and recall rates.

- The project showcases the power of data-driven learning over static rule-based filtering systems. Through careful experimentation and testing, it provides an effective, scalable, and user-friendly solution to a real-world problem that affects millions of users daily.

- Challenges such as class imbalance, short message length, and evolving spam patterns were addressed using balanced datasets and feature engineering.
- Future work may involve implementing deep-learning-based architectures for improved contextual understanding, and expanding to multilingual spam detection.
- In conclusion, this system represents an important step toward secure digital communication, spam prevention, and the practical application of machine learning in text analytics.