SMS Spam Detection — Project Template

1. Title Page

- **Project Title**: SMS Spam Detection Using Machine Learning
- **Your Name:** Rajumenan B
- **Institution / Department :** University college of engineering ,Tindivanam / B.E CSE
- **Date :** 17-10-2025
- **Github :** https://github.com/Rajumenan/SMS-Spam-Detection

---

2. Table of Contents

3. Project Overview

- **Background / Motivation**:
  Spam messages are a major concern for users and mobile networks, as they often contain fraudulent, phishing, or promotional content. Detecting spam SMS automatically improves user experience, prevents scams, and strengthens communication security.

- **Scope**:
  This project focuses only on SMS text messages, classifying them into two categories: Spam or Ham (Not Spam) using machine learning techniques. The project does not cover emails or multimedia spam.

- **High-level Description**:
  The system reads raw SMS messages, preprocesses them, converts text into numerical features, trains multiple ML models, and predicts whether a new message is spam or not.

- **Dataset / Data Source Summary**

  https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

---

4. Objectives & Problem Statement

4.1 Problem Statement

Given a short SMS text message, classify it as **Spam** or **Ham** using machine learning.
Challenges include:

- Short informal text
- Misspellings, abbreviations, and slang
- Class imbalance (fewer spam than ham)

4.2 Objectives

List specific goals your project will achieve.

Examples:

- Build a predictive model with high accuracy and recall for spam detection.
- Preprocess and clean text effectively.
- Compare multiple ML algorithms.
- Visualize model performance.
- Develop a reusable prediction pipeline

---

5. Proposed Solution

- **Approach / Methodology**:
  A supervised machine learning approach was used. The main pipeline:
  Raw SMS → Preprocessing → Feature Extraction → Model Training → Prediction.

- **Pipeline Overview**: raw SMS → preprocessing → feature extraction → model → prediction

- **Justification of Choices**:

  - TF-IDF Vectorizer to represent text efficiently.
  - Multinomial Naive Bayes model for fast and accurate text classification.
  - Alternative models (Logistic Regression, SVM) tested for comparison.

---

6. Features

6.1 Functional Features

- Accepts raw SMS text and outputs “Spam” or “Ham”.
- Provides classification probability.

- Can process multiple messages at once.
- Easy to extend into a GUI or API interface.

6.2 Non-Functional Features

- High accuracy (above 95%).
- Fast and lightweight.
- Reliable and easy to maintain.
- User-friendly for non-technical users.

---

7. Technologies & Tools

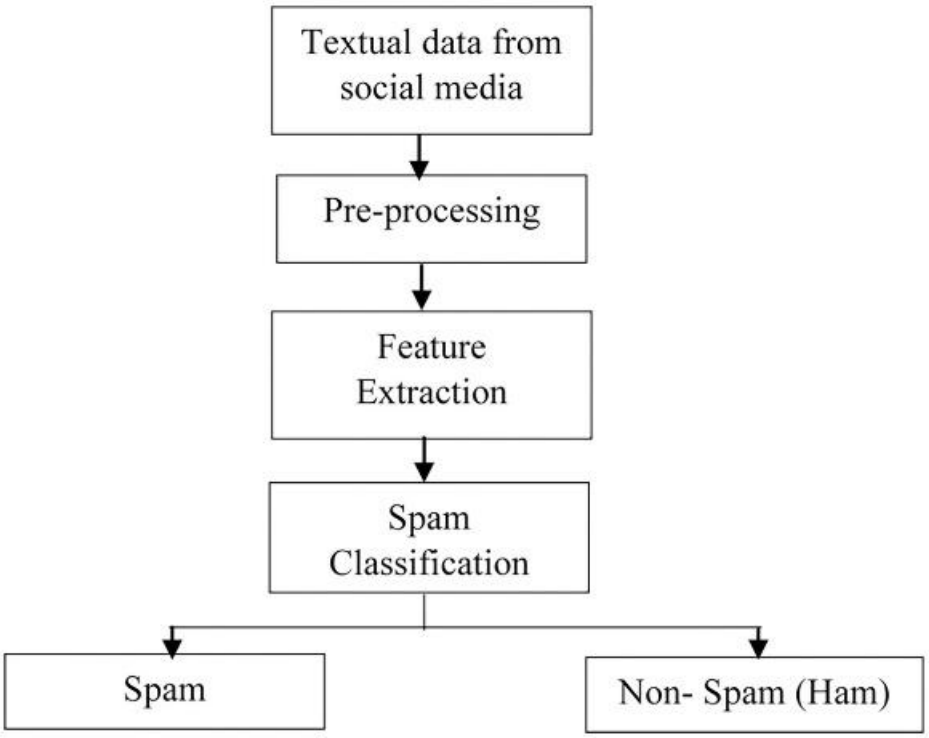
List all software, libraries, frameworks used. Examples:

- **Language**: Python
- **Data Handling**: pandas, numpy
- **Visualization**: matplotlib, seaborn
- **Text / NLP**: nltk, re (regex), spaCy (if used)
- **Feature extraction**: scikit-learn's CountVectorizer, TfidfVectorizer
- **Machine Learning**: scikit-learn (Naive Bayes, Logistic Regression, Random Forest, SVM etc.)
- **Model persistence**: pickle, joblib
- **Environment / Tools**: Google colab / GitHub

---

8. System Architecture

- **Architecture Diagram** (block diagram)



- **Component Descriptions**:
  - **Preprocessing Module:** Cleans and tokenizes SMS text.
  - **Feature Extraction:** Converts cleaned text into TF-IDF vectors.
  - **Model Training:** Learns spam/ham patterns using ML.
  - **Prediction Layer:** Outputs spam or ham for new messages.

- **Data Flow**: How data moves through the system (during training, and during inference)

---

9. Implementation Steps

Step-by-step how you built the system. Example:

1. Load dataset (spam.csv).

2. Perform EDA — check spam vs ham count, message lengths, word clouds.
3. Preprocess text:
    - Lowercasing
    - Removing punctuation and stopwords
    - Tokenization and stemming
4. Convert to numeric features using TF-IDF.
5. Split data into train (80%) and test (20%).
6. Train models:
    - Multinomial Naive Bayes
    - Logistic Regression
    - SVM
7. Evaluate using Accuracy, Precision, Recall, F1-score.
8. Visualize confusion matrix and word frequencies.
9. Save final model and vectorizer using pickle.
10. Build prediction function / UI for new messages.

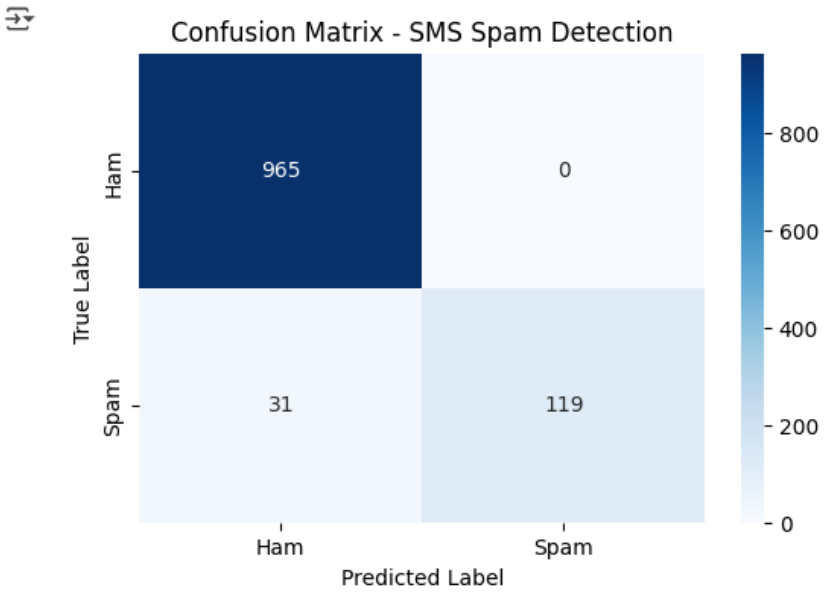
---

10. Output / Screenshots

- Sample inputs & outputs (SMS text → prediction)

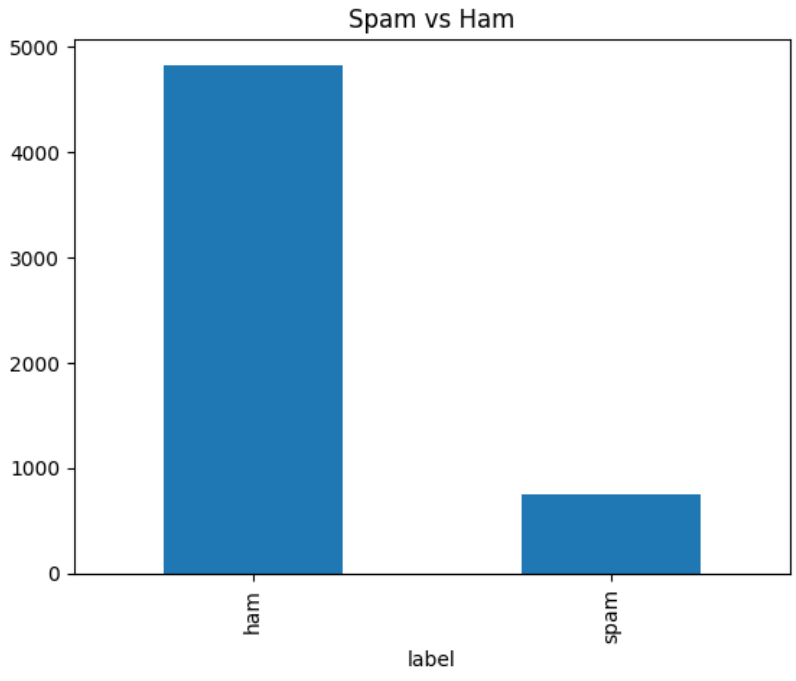
	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

- Confusion matrix, classification report



- Data Exploration

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   label    5572 non-null   object
 1   message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

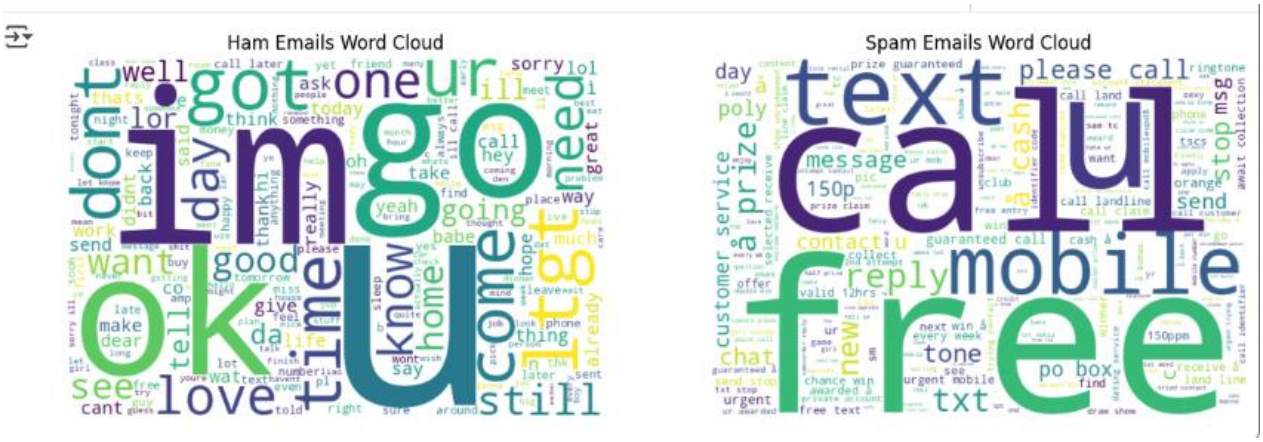


- Model Training

```
model = MultinomialNB()
model.fit(X_train_tfidf, y_train)
```

```
MultinomialNB()
```

- Any feature importances / word clouds / insights visuals



- Evaluation

```
Accuracy: 0.9721973094170404
[[965   0]
 [ 31 119]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       965
           1       1.00      0.79      0.88       150

    accuracy                           0.97      1115
   macro avg       0.98      0.90      0.93      1115
weighted avg       0.97      0.97      0.97      1115
```

11. Advantages

- Automatic and fast SMS classification.
- High accuracy and low false positives.
- Scalable for real-time detection.
- Lightweight and easy to deploy.
- Extendable for other text sources (emails, chat).

---

12. Future Enhancements

- Use Deep Learning models (LSTM, BERT).
- Support multilingual SMS.
- Apply SMOTE for class balancing.
- Deploy as web API or mobile app.
- Integrate with real-time spam filtering systems.

---

13. Conclusion

This project successfully demonstrates the use of machine learning for SMS spam detection.

The final model achieves high accuracy and reliability using simple NLP pre-processing and TF-IDF features.

It provides a foundation for advanced spam detection systems, with potential for deployment as an online service or integrated app.