**SMS Spam Detection**

**Project Documentation**

**📝 Overview**

The **SMS Spam Detection** project is a **Flask-based web application** that allows users to input an SMS message and receive a classification result indicating whether the message is **Spam** or **Ham** (Not Spam).

This system utilizes **Machine Learning (ML) models** trained on an SMS dataset to identify spam messages accurately. It integrates a **user-friendly web interface** where users can submit SMS messages and view real-time predictions.

**📂 Project Structure**

The project is structured as follows:

**📂 Root Directory**

The root directory contains the main scripts and essential files required to run the project.

**Main Scripts**

📌 **app.py** → The main Flask application that serves the web interface and handles API requests for spam detection.  
**📌 training.py** → Script for training various machine learning models using the spam.csv dataset.  
**📌 testing.py** → Script for testing the trained models with sample messages.  
📌 **SMS\_SPAN\_DETECTION.ipynb** → Jupyter Notebook for exploratory data analysis, visualization, and model training/testing.

**Configuration & Dependencies**

📌 **requirements.txt** → Contains a list of dependencies required for the project (Flask, scikit-learn, nltk, tensorflow, etc.).  
📌 **roadMap.txt** → A document outlining the project's development plan or progress tracking.

**📁 data/**

Contains raw and preprocessed datasets used for training the spam detection models.

📌 **spam.csv** → The dataset containing SMS messages labeled as **spam** or **ham**. This is used for training and testing the models.

**📁 models/**

Contains trained models and preprocessing files required for spam classification.

📌 **gru\_model.keras** → Trained GRU (Gated Recurrent Unit) deep learning model for spam detection.  
📌 **lr\_model.pkl** → Trained Logistic Regression model.  
📌 **lstm\_model.keras** → Trained LSTM (Long Short-Term Memory) deep learning model.  
📌 **nb\_model.pkl** → Trained Naïve Bayes model.  
📌 **rf\_model.pkl** → Trained Random Forest model.  
📌 **svm\_model.pkl** → Trained Support Vector Machine model.  
**📌 processed\_data.pkl** → Preprocessed dataset used for training.  
📌 **tfidf\_vectorizer.pkl** → Saved **TF-IDF vectorizer**, used for converting text messages into numerical form for model input.

**📁 plots/**

Contains graphical representations of model performance and comparisons.

📌 **Accuracies.png** → A bar chart or graph comparing the accuracy of different models.  
📌 **ML\_MetricsComparrision.png** → A performance comparison of various machine learning models based on evaluation metrics (precision, recall, F1-score, etc.).  
📌 **spamVSham.png** → A visualization showing the distribution of spam vs. ham messages in the dataset.

**📁 static/**

Contains **static assets** such as JavaScript and CSS files for the web interface.

📌 **script.js** → Handles frontend interactions, form submission, and API requests to the backend.  
📌 **style.css** → Provides styling for the **index.html** web page.

**📁 templates/**

Contains **HTML templates** used by the Flask application for rendering web pages.

📌 **index.html** → The main web page where users input SMS messages for spam detection.

**🔑 Key Components**

**1️. Flask Application (app.py)**

The **Flask backend** is responsible for handling web requests and performing spam classification.

**Endpoints:**

📌 **Home Page (/)**

* Serves the index.html file to the user.

📌 **Prediction API (/predict)**

* **Receives** SMS input via a **POST request**.
* **Processes** the text and sends it to the ML model.
* **Returns** a JSON response indicating **Spam or Ham**.

**Example API Response:**

{

"message": "This is a free lottery ticket!",

"prediction": "Spam"

}

**2️. Machine Learning Model (training.py)**

The ML model is trained using **Natural Language Processing (NLP)** techniques and **various classifiers**.

**Steps in Training:**

✅ **Data Preprocessing**

* Convert text to lowercase.
* Remove **stopwords, special characters, and numbers**.
* Perform **tokenization** (split text into words).
* Convert text into numerical vectors using **TF-IDF (Term Frequency-Inverse Document Frequency)**.

✅ **Training Different Models**

* **Naïve Bayes (NB)** → Fast and efficient for text classification.
* **Logistic Regression (LR)** → Works well with large datasets.
* **Random Forest (RF)** → Reduces overfitting and improves accuracy.
* **Support Vector Machine (SVM)** → Finds optimal decision boundaries.

✅ **Model Saving**

* The best-performing model is saved as **model.pkl** for future predictions.

**3️. Frontend - User Interaction**

**HTML (index.html)**

* A simple form where users enter an SMS message.
* A button to submit the message for spam detection.

**CSS (style.css)**

* Custom styles for an **intuitive** and **responsive** UI.
* Ensures a **clean and professional layout**.

**JavaScript (script.js)**

* Handles **form submission** asynchronously using the **Fetch API**.
* Prevents the page from reloading when submitting data.
* Displays the **spam detection result dynamically**.

**🚀 How to Run the Application**

**1️. Install Dependencies**

Ensure that the required Python libraries are installed:

pip install flask nltk scikit-learn imbalanced-learn tensorflow

**2️. Train the Model**

Run the training script to train and save the ML model:

python training.py

**3️. Start the Flask Application**

Run the Flask server:

python app.py

After starting, you will see an output like:

\* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

**4️. Access the Web Interface**

Open a web browser and visit:  
🔗 http://127.0.0.1:5000/

* **Enter an SMS message** in the input box.
* **Click "Check"** to see whether the message is **Spam or Ham**.

**⚙️ Troubleshooting**

**1️. API Not Working (/predict not responding)**

🔍 **Solution:**

* Ensure that app.py is running correctly.
* Open the **browser console (F12 → Console tab)** and check for JavaScript errors.

**2️. Model Not Found (model.pkl missing error)**

🔍 **Solution:**

* Make sure **training.py** has been executed successfully.
* Check if **model.pkl** is present in the project directory.

**3️. Incorrect Predictions**

🔍 **Solution:**

* Try using a **different machine learning model** for better accuracy.
* **Increase the dataset size** for better training results.

**🛠️ Future Enhancements**

🚀 **To further improve the project, consider adding:**  
✅ **Deep Learning Models** (LSTMs, Transformers) for better spam detection.  
✅ **User Authentication** (Login system to track SMS history).  
✅ **Real-time SMS Filtering API** for integration with messaging apps.  
✅ **Mobile App Version** to allow SMS classification on smartphones.

**📌 Conclusion**

This project demonstrates **how Machine Learning can be used for SMS Spam Detection** through a simple **Flask web application**. It incorporates:

✔ **A trained ML model** for text classification.  
✔ **A Flask backend** to process and serve predictions.  
✔ **A dynamic frontend** for user interaction.