

Machine Learning: From Beginning to Master

by NASCENIA

Report on Final Assessment Project

Submitted by:

Md. Zayed Al Masud

Project Description:

Title: SMS Spam Classifier using Machine Learning

Objective:

The goal of this project is to build an end-to-end system that can automatically classify SMS messages as either Spam or Not a Spam.

Dataset Details:

Dataset: SMS Spam Collection Dataset (UCI/Kaggle)

Size: 5,574 SMS messages labeled as either ham (legitimate) or spam.

Format: Two columns – label (spam/ham) and text (message body).

SMS spam classifier project addresses the widespread issue of spam messages, which can be annoying for users and sometimes malicious. Spam wastes time, clutters inboxes, and can be used for phishing attacks. Automatic detection improves user experience and security.

Solution Overview

Data Preprocessing

- Loaded dataset from Kaggle.
- Cleaned text: lowercasing, removing punctuation, trimming spaces.
- Encoded labels: ham → 0, spam → 1.
- Split data: 80% training, 20% testing.

Model Architecture

- Used Hugging Face Transformers pipeline for text-classification.
- Base model: 'distilbert-base-uncased' (pretrained on English text).
- Fine-tuned on SMS Spam dataset with 5 epochs.

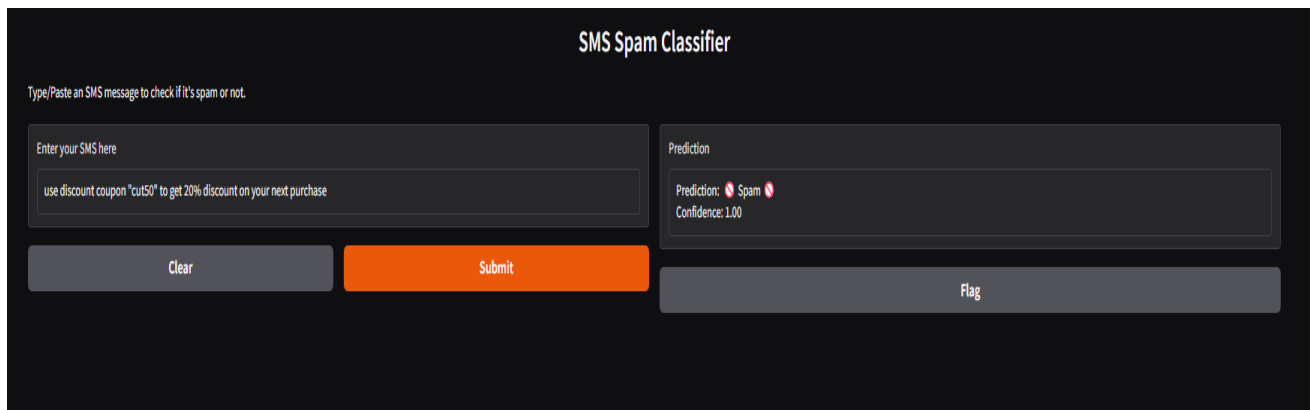
Training

- Used Hugging Face Trainer API.
- Learning rate = $2e-5$.
- Evaluation: Accuracy and loss measured per epoch.
- Results: Achieved ~97% validation accuracy.

Deployment

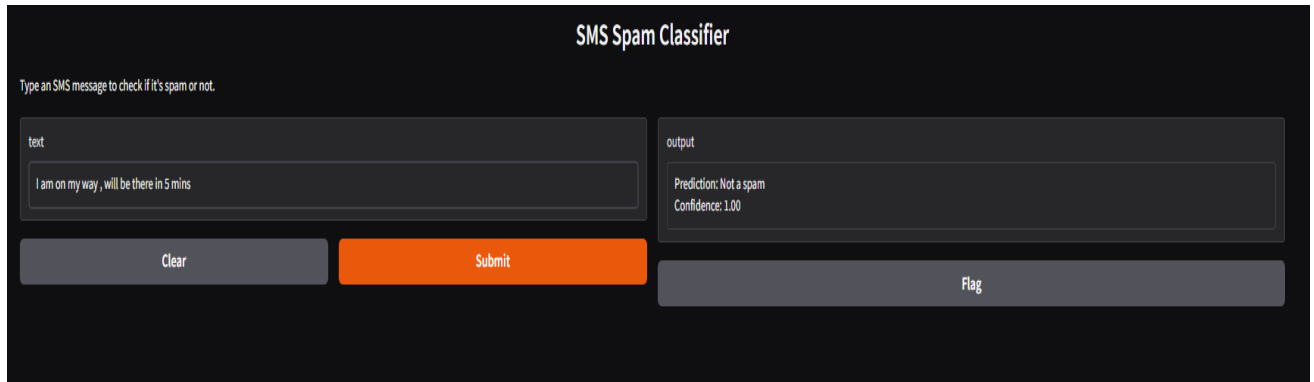
- Built frontend using Gradio Interface.
- Users can input SMS text and get prediction instantly.
- Hosted locally.

Demo Screenshots:



The screenshot shows the 'SMS Spam Classifier' web interface. At the top, it says 'Type/Paste an SMS message to check if it's spam or not.' Below this is a text input field labeled 'Enter your SMS here' containing the message: 'use discount coupon "cut50" to get 20% discount on your next purchase'. To the right of the input field is a 'Prediction' box showing 'Prediction: 🚫 Spam 🚫' and 'Confidence: 1.00'. At the bottom, there are three buttons: 'Clear' (grey), 'Submit' (orange), and 'Flag' (grey).

Figure 1: Example of a Spam SMS (Zoom in For better visualization)



The screenshot shows the 'SMS Spam Classifier' web interface. At the top, it says 'Type an SMS message to check if it's spam or not.' Below this is a text input field labeled 'text' containing the message: 'I am on my way , will be there in 5 mins'. To the right of the input field is an 'output' box showing 'Prediction: Not a spam' and 'Confidence: 1.00'. At the bottom, there are three buttons: 'Clear' (grey), 'Submit' (orange), and 'Flag' (grey).

Figure 2: Example of Not a Spam SMS (Zoom in For better visualization)

Challenges and Learnings

Challenges Faced:

- Library version mismatches (Transformers TrainingArguments keyword errors).
- Dependency conflicts in Kaggle (e.g., PyTorch CUDA mismatches).
- Styling issues in Gradio when trying to customize background dynamically.

How They Were Overcome:

- Checked Hugging Face version compatibility and used correct arguments.
- Reinstalled matching versions of PyTorch/Transformers.
- Default styling is kept.

Key Learnings:

- Fine-tuning pretrained models is far more efficient than training from scratch.
- Hugging Face + Gradio makes end-to-end ML deployment simple.
- Debugging environment conflicts is a crucial. And also fun when every problem is resolved.

References

- **Dataset:** [UCI SMS Spam Collection Dataset](#)
- **Hugging Face Transformers:** <https://huggingface.co/transformers/>
- **Gradio Documentation:** <https://gradio.app/docs/>
- **PyTorch Documentation:** <https://pytorch.org/docs/stable/>