Email Spam or Ham Checker

Author: H. Patel

Date: October 2025

Institution: CDI College – Artificial Intelligence Specialist Program

1. Introduction

The main goal of this project is to build a program that can automatically detect whether an email is spam or not. Spam emails are unwanted messages that often contain advertising, scams, or harmful links. The project uses machine learning to study examples of emails and then predict if a new message is spam or ham (not spam). This helps improve safety and saves time by reducing junk messages.

2. Methodology

The project follows a step-by-step workflow. First, two datasets were combined into one large file using Python and Pandas. Then, the text data was cleaned and changed into numbers using TF-IDF (Term Frequency–Inverse Document Frequency). Three machine learning models were tested: Logistic Regression, Linear Support Vector Classifier (LinearSVC), and Random Forest Classifier. These models were trained using scikit-learn. MLflow was used to track all the experiments and results automatically.

Below is the area where a screenshot of MLflow can be placed to show how the models and hyperparameter tuning were logged:

(Insert MLflow Screenshot Here)

Finally, a Streamlit web app was made so that users can type or paste any message and check if it is spam or ham. A Sphinx documentation folder was also created to explain each part of the project and generate an easy-to-read report about the code.

3. Results

4. Personal Reflection

Working on this project was a great learning experience. I learned how to combine data, clean text, train machine learning models, and build a web app. It was exciting to see the project go from raw data to a working spam detection system. Using MLflow and Streamlit helped me understand how real projects are tracked and presented. The modular folder structure made it easier to organize everything and fix problems faster.

5. Conclusion

In conclusion, the Email Spam Verifier project was successful in building a working spam detection system. It showed how machine learning can be used to solve real-world problems like spam filtering. The Random Forest Classifier was found to be the most accurate model. The project also demonstrated how tools like MLflow, Streamlit, and Sphinx can be used together for training, testing, and documentation.

6. Challenges Faced

During this project, several problems were faced and solved along the way:

• Setting up the Python environment was sometimes difficult because some libraries like MLflow, Streamlit, and scikit-learn had version conflicts.

• Import errors happened when running the project in VS Code. This was fixed by adding the correct folder path to the code using sys.path.

• The Sphinx documentation tool gave errors such as 'conf.py missing' or 'source directory not found'. These issues were fixed by reconfiguring the docs folder and adding a proper conf.py file.

• MLflow showed many warnings like 'google.protobuf.service module is deprecated' and 'pkg_resources is deprecated as an API'. These warnings did not stop the program but made the output messy. To fix this, warnings were hidden using warnings.filterwarnings('ignore').

• The model accuracy was not always stable at first. Logistic Regression sometimes gave lower accuracy. After cleaning the data and adjusting the TF-IDF settings, the accuracy improved to around 95–96%. Random Forest gave the best and most consistent results.

• Training time was long because GridSearchCV tested many parameter combinations. The process was made faster by using fewer folds and smaller grids.

• In the Streamlit app, at first, there was no way to type a custom email message. Later, a text box was added so the user can type or paste any content for prediction.

All these problems helped build better understanding of debugging, improving accuracy, and working with real machine learning environments.

3. Results (Updated Final Version)

All three models gave good results after training. The Linear SVM model performed best with 96.65% accuracy and an F1 score of 0.9224. The Logistic Regression model followed closely with 95.04% accuracy, while the Random Forest model achieved 94.27% accuracy.

Below is the complete model comparison table from the final training run on the merged dataset (merged_spam.csv, 44,003 samples):

Model	Accuracy	F1 Score
Logistic Regression	0.9504	0.8901
Linear SVM	0.9665	0.9224
Random Forest	0.9427	0.8543

Best Model: Linear SVM — It produced the highest accuracy and F1 score, offering the most balanced precision and recall.

The results were automatically logged and visualized in MLflow, including precision, recall, accuracy, and F1 metrics. Each experiment recorded the hyperparameters used, the training time, and the resulting performance metrics. A screenshot of the MLflow tracking interface can be placed below to illustrate the logged runs and hyperparameter tuning:

(Insert MLflow Screenshot Here)

All trained models (logreg_model.joblib, linearsvc_model.joblib, and rf_model.joblib) were stored in the model/ folder along with the TF-IDF vectorizer (vectorizer.joblib) for later use in the Streamlit prediction app.