Project Report
on
Email/SMS Spam Classifier: Using Machine Learning to Detect Spam Messages

(CSE V Semester Mini project)

2024-2025

Submitted To:

Mr. Nitin Thapliyal

(CC-CSE-D2-V-SEM)

Submitted By:

Kajal Bisht

Roll No. 2218908

CSE-D2-V-Sem

Session- 2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

CERTIFICATE

Certified that Kajal Bisht(Roll No.- 2218908) has developed mini project on "**Email/SMS Spam Classifier: Using Machine Learning to Detect Spam Messages**" for the CSE V Semester Mini Project Lab in Graphic Era Hill University, Dehradun. The project carried out by Students is their own work as best of my knowledge.

DATE:18/01/2025

(Mr. Nitin Thapliyal)
Class Coordinator
CC-CSE-D2-V-Sem
CSE Department GEHU,
DEHRADUN

ACKNOWLEDGMENT

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our project Co-Ordinator Mr. Nitin Thapliyal for his patience, support and encouragement throughout the completion of this project and having faith in us.

At last, but not the least We greatly indebted to all other persons who directly or indirectly helped us during this work.

Kajal Bisht

Roll no.22198908

CSE-D2-V-Sem

Session: 2024-2025

GEHU, Dehradun

1. INTRODUCTION

About the Application

The Email/SMS Spam Classifier application aims to automatically detect and classify messages as spam or non-spam (ham) based on their content. Spam messages are typically unsolicited and unwanted, often used for advertising or malicious purposes. The classifier uses machine learning models to analyze message content and determine whether it is spam or ham. The system helps filter out unwanted messages, enhancing productivity and security.

Technologies Used

The project utilizes a combination of the following technologies:

- **Python Libraries**: Pandas, Numpy, Scikit-learn, NLTK, Seaborn, Matplotlib
- **Machine Learning Models**: Naive Bayes, Logistic Regression, SVM, KNN, Decision Tree, Random Forest
- **Deployment Framework**: Streamlit
- **Dataset**: SMS Spam Collection Dataset

Purpose and Objectives

The purpose of this application is to build a reliable machine learning-based spam classifier that can detect spam messages in real-time. The objectives of the project include:

- Developing a machine learning model to classify SMS or email messages as spam or not spam.
- Utilizing various feature extraction techniques such as TF-IDF and CountVectorizer for text data.
- Deploying the model as a web application to allow users to classify messages easily.

2. SYSTEM DESIGN

Modules and Libraries

The system is built using various modules and libraries to handle different tasks:

- **Data Preprocessing**: Pandas and Numpy are used for data manipulation and cleaning.
- **Feature Extraction**: Scikit-learn's CountVectorizer and TF-IDF Vectorizer are used for converting text into numerical features.
- **Machine Learning Models**: Scikit-learn is used to implement different models such as Naive Bayes, SVM, and Random Forest.
- **Visualization**: Seaborn and Matplotlib are used to generate visualizations such as histograms and confusion matrices.
- **Deployment**: Streamlit is used to deploy the trained model as a web application, allowing users to input messages and receive classifications in real-time.

Application Workflow

1. **Data Input**: The user inputs an SMS or email message into the application interface.
2. **Preprocessing**: The message is preprocessed by converting text to lowercase, removing stopwords and punctuation, and performing tokenization and stemming.
3. **Feature Extraction**: The preprocessed text is converted into numerical features using TF-IDF Vectorizer.
4. **Prediction**: The processed data is fed into the trained machine learning model (e.g., Naive Bayes), which classifies the message as spam or not spam.
5. **Result Display**: The result ("Spam" or "Not Spam") is displayed to the user.

3. IMPLEMENTATION

User Interface (UI)

The user interface is designed using **Streamlit**. It provides an interactive platform where users can input messages and get predictions in real-time. The UI includes:

- A text input box where users can enter the message.
- A button to submit the message for classification.
- A display area showing the result (Spam or Not Spam).

File Handling

The application saves and loads models using the **Pickle** library. This allows the machine learning model to be stored after training and loaded later for inference:

- **Model Storage**: After training, the model is serialized using pickle and saved to disk.
- **Model Loading**: On each prediction request, the saved model is loaded from disk and used to classify the input message.

4. TESTING AND RESULTS

User Review

User testing showed that the application performs well in identifying spam messages. User found the interface simple and intuitive, allowing them to classify messages with ease. Feedback highlighted that the classification results were accurate, and the system was responsive.

Error Handling

The application has been designed to handle errors gracefully:

- **Invalid Input**: If the user inputs a message that is too short or contains special characters, the system provides an appropriate error message.
- **Model Failures**: In case of any failure while loading the model or during prediction, a fallback message is shown, ensuring the system doesn't crash unexpectedly.
- **Edge Cases**: The application also handles edge cases, such as when the message contains non-English characters or slang.
- **Performance Metrics:**

  Best Model: Multinomial Naive Bayes

  Accuracy: 97.5%

  Precision: 96.8%

5. FUTURE ENHANCEMENTS

Feature Extensions

1. **Multilingual Support**: The system can be extended to classify messages in multiple languages by training the model on multilingual datasets.
2. **Deep Learning Models**: Future work will explore using deep learning techniques such as LSTM or BERT for more sophisticated text understanding and improved classification accuracy.
3. **User Feedback Loop**: Incorporating a user feedback loop to continually improve the model by retraining it on new data, thus adapting to evolving spam trends.

Performance Improvements

1. **Model Optimization**: Further performance improvements can be achieved by tuning the hyperparameters of the models or using ensemble techniques to combine the predictions of multiple models.
2. **Real-time Updates**: Implementing real-time updates for model retraining based on new incoming data can help the system remain effective as spam trends change.
3. **Parallel Processing**: For large-scale deployments, implementing parallel processing and load balancing techniques will ensure faster predictions, even when the system handles a large volume of requests.

6. CONCLUSION

The Email/SMS Spam Classifier project demonstrates the successful use of machine learning techniques to automatically classify messages as spam or non-spam. The application achieved high accuracy and precision using models like Naive Bayes and Logistic Regression, making it highly effective for real-world usage. The user-friendly interface built with Streamlit makes it easy for users to classify messages in real time.

Future work includes exploring deep learning models, adding multilingual capabilities, and continuously improving the classifier based on user feedback. The project highlights the practical application of machine learning in enhancing user experience by reducing spam-related distractions and increasing security in communication.

REFERENCES

1. SMS Spam Collection Dataset on Kaggle
2. Python Streamlit Documentation
3. Natural Language Toolkit (NLTK)
4. Scikit-learn User Guide
5. Spam Detection with Machine Learning