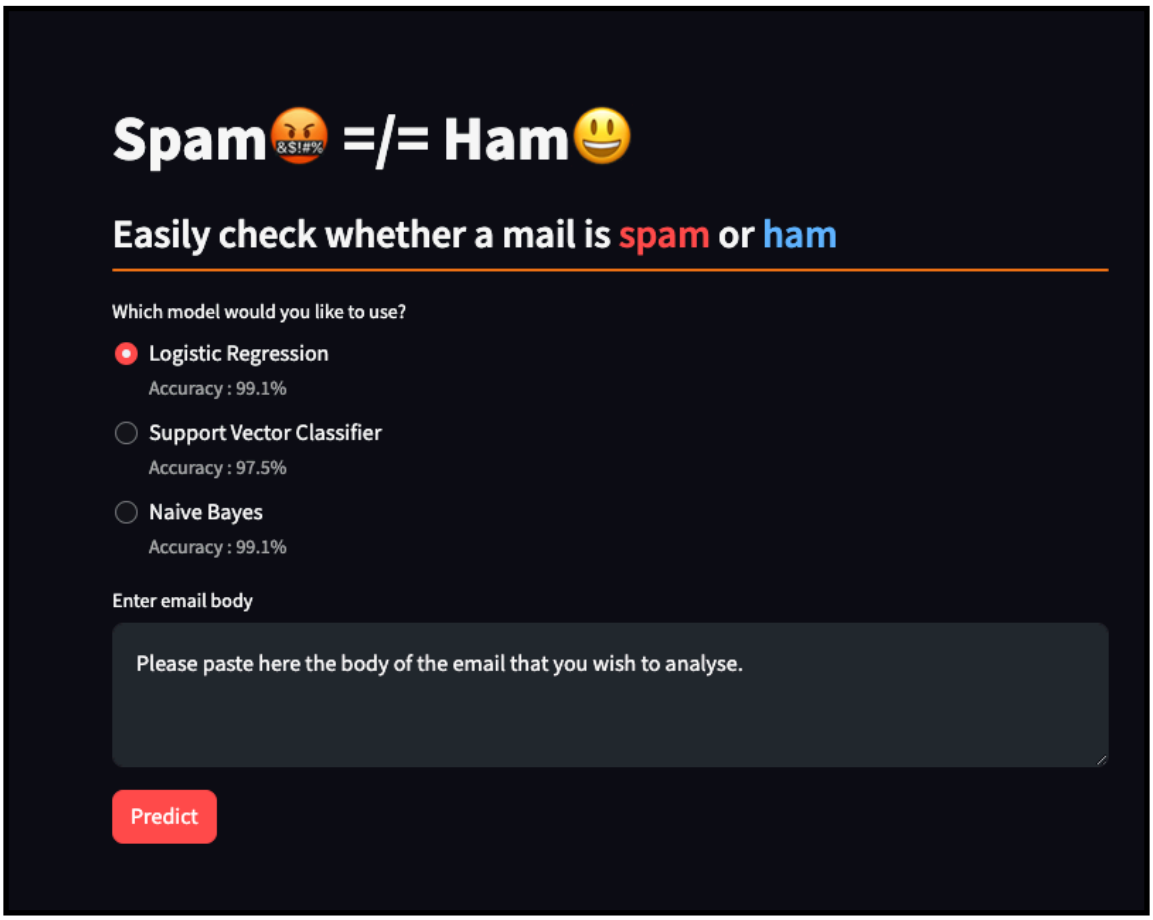
Project Report

Spam or Ham

Natural Language Processing Project



Akshat Sharma
B.Tech Hons. CSE AIML
500097291, R2142211317

Introduction

Email has become an integral part of modern communication, enabling individuals and organisations to exchange information efficiently. However, the prevalence of email has also given rise to the problem of unsolicited bulk emails commonly known as spam. Spam emails are not only a nuisance but can also pose serious security threats, such as phishing attempts, malware distribution, and other malicious activities.

The ability to accurately classify incoming emails as either legit(ham) or spam is crucial for maintaining a clutter-free inbox and mitigating potential security risks. This project aims to develop an effective email classification system using NLP techniques to automatically detect and filter out spam emails.

Dataset and Preprocessing

The dataset used for this project is a csv file containing a collection of 5728 emails, each labelled as either “spam” or “ham”.

Preprocessing the raw email data is crucial step in NLP tasks, as it hep s to remove noise and transform the text into a more suitable format for machine learning model. I have applied several preprocessing steps all of which have been explained below.

1. Punctuation Removal

Punctuation marks such as commas, periods, and exclamation marks carry little semantic value for text classification tasks. To remove punctuation marks, we iterated through each character in the email text and filtered out

any character that belonged to the string.punctutation set. This step helped to reduce noise and simplify text.

2. Stop Word Removal

Step words are common words like “the”, “and” etc. that occur frequently in text but contribute little to the overall meaning. We used the NLTK library to remove stop words. This step is helpful in reducing the dimensionality of the feature space and focus on more meaningful words for classification.

3. Stemming

Stemming is the process of reducing words to their base or root form, known as the stem. To perform this step, we employed the Porter Stemmer algorithm from the NLTK library to perform stemming on the Emil text. Stemming can help to group together semantically related words, potentially improving the model’s performance and reducing the feature space.

4. Text Transformation

Once the above 3 preprocessing steps are complete, we have to transform the email text into a numerical representation suitable for machine learning algorithm. For this we used the CountVectorizer from scikit-learn library. This library implements the bag of words model which represents each email as a vector of word counts, where each dimension corresponds to to a unique word in the corpus.

Methodology

To build and evaluate our models for spam detection, we follow the usual methodology of data splitting, model training and performance evaluation.

1. Data splitting

We divide the dataset into separate training and testing sets. This step is crucial to assess the model's performance on unseen data and prevent overfitting. We used scikit-learn's train_test_split function to create an 80-20 split, allocating 80% of the data for training and the remaining 20% for testing.

2. Machine Learning Models

- **Logistic Regression**

Logistic regression is a popular linear model that models the probability of an instance belonging to a particular class. Despite it's name, it can be used for classification tasks by applying a decision threshold to the predicted probabilites. The logistic regression model was able to achieve an accuracy of around 99% for our task.

- **Naive Bayes**

The naive bayes algorithm is a probabilistic classifier based on the Baye's theorem. It assumes independence among features and is often employed in text classification due to it's simplicity and efficiency. We used the multinomial Naive Bayes since we are dealing with textual data.

- **Support Vector Machine(SVM)**

SVMs are robust to high dimensional data can can capture complex decision boundaries, making em well suited for text classification tasks. However, they can be computationally expensive, especially for large datasets, and require careful parameter tuning.

3. Model Training and Evaluation

Each of the three models were trained using the training data, which consisted of the preprocessed email text and the corresponding labels. To evaluate the performance of the trained models, we used the testing data and calculated the accuracy score, which measures the proportion of

correctly classified instances. All three models achieved impressive accuracy scores ranging from 97% to 99% on the test data.

UI Development

To provide an interactive and user friendly way of utilising the trained email classification model, we developed a web based user interface using the Streamlit library. Streamlit is an open-source app framework that allows creation of beautiful and interactive data applications with minimal coding effort. The primary goal of the UI was to enable users to input the text content of an email and receive a prediction from the trained model, classifying the email as either “spam” or “ham”. The UI not only enhances the usability of the email classification system but also serves as a demonstration of how the model can be integrated into a real world application.