Abstract

Below report shows the in detail project report and implementation of our project spam email detection.



Pratikshay Pradip Awadhoot

**Presented By:**

**Avanti Nipunge - 2886120**

**Myna Nereti - 2884801**

**Pratikshay Awadhoot - 2881830**

Project Report on Spam Email Detection

1. **Platform/System Setup Procedures**

The notebook is designed to be executed in **Google Colab** environment with the following setup steps:

1. **Environment Preparation**:

* Install dependencies:
* Required Libraries:
  + - scikit-learn: For machine learning models.
    - pandas: For data manipulation.
    - matplotlib: For visualization.
    - Below screenshot shows all the required libraries that need to import for data preprocessing, data cleaning, word embedding, visualization, UI and prediction.

A screenshot of a computer program

Description automatically generated

Figure: Screenshot of importing libraries

1. **File Preparation**:

* Upload the dataset Combined\_Spam\_Ham.zip to the directory.
* Unzip the file and ensure it is extracted into a folder named Combined\_Spam\_Ham.
* Check the files as they contain encoded emails and they’re not in .txt file, so verify the files are properly extracted

**Verification**:

* Test the environment by running
* Check the screenshot below for verification. On the left tab you can see Combine\_Spam\_Ham folder that got extracted and in the output window you can see one of the raw emails get extracted

A screenshot of a computer program

Description automatically generated

Figure: Screenshot of extraction of Raw files

1. **Load Data and extract features**

Now, we have loaded the extracted folder to one data frame for extracting features, as we have presented ppt in class, we have shown that we have removed html tags, css feature, message id and all the other unnecessary features that we don’t need for spam email detection.

Most important feature that we extracted from Raw email data.

* File name
* Sender
* Receiver
* Email Subject
* Body of the email

Check the screenshot below for the feature extraction from raw data, where we have shown loading the data and then only important features extraction for next process

A screenshot of a computer program

Description automatically generated Figure: Load and extract features

1. **Putting all the data into csv format for next process of data cleaning**

* Saved the dataframe to CSV file called email\_features.csv
* Check the zip folder for CSV file
* Below Screenshot shows the data types of features as well.

A screenshot of a computer program

Description automatically generated

Figure: Converting data frame to CSV file

1. **Removing all the unnecessary tags and creating cleaned body**

* Remove HTML tags, non-alphanumeric character, extra spaces, numbers, URLs
* We used STOPWORDS feature in English language for removing unknowing words in the email body that we don’t need.
* The cleaned body is ready now and converted to CSV preprocessed\_email.csv (check the zip folder)

A screen shot of a computer program

Description automatically generated

Figure: Cleaned body with features

1. **Bootstrapping to create a larger dataset**

As we had 6000 emails, we need a larger dataset (at least 10000, suggested by professor) we must apply the bootstrapping method for larger dataset. After bootstrapping we made 15000 rows of data.

A black screen with white lines

Description automatically generatedA screenshot of a computer

Description automatically generated Figure: Bootstrapping to create larger dataset

Figure: 15000 rows of data

1. **Adding class label (spam/ham) based on spam keywords**

To process for the training of the model to detect ham or spam email, we need to provide class label in binary form. 0 is ham and 1 is spam

Below screenshot show the number of spam keywords that can predict on the email that it is ham or spam.

We searched all spam and ham emails manually to do so and searched the search engine what kind of keywords or indicators could be helped to verify that email is spam or non-spam.

A screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated Figure: Adding class label on spam indicators

Figure: CSV file with class label on spam indicators

1. **Using Word2Vec model for our project**

**Why Use Word2Vec for Spam Email Detection?**

1. **Semantic Understanding**:

Spam emails often use synonyms or rephrased content to bypass keyword-based filters. For example, like this for understanding:

"Congratulations! You've won!" vs. "Congrats! You're a winner!"

Word2Vec captures the semantic similarity between words like "congratulations" and "congrats," allowing the model to detect spam content even if the wording changes.

1. **Contextual Features**:

Word2Vec uses the **context** of a word (neighboring words) to generate embeddings. For instance, in "Limited time offer," the word "offer" in this context may strongly indicate spam.

This helps classify emails based on subtle contextual differences.

1. **Dimensionality Reduction**:

Instead of using a high-dimensional sparse matrix (e.g., bag-of-words or TF-IDF), Word2Vec represents words in a low-dimensional vector space, reducing memory and computational overhead while preserving meaningful information.

Below screenshot shows how we used Word2Vec model in our project for spam analysis using indicators.

A screenshot of a computer program

Description automatically generated

Figure: Word2Vec model for word embedding to predict spam

1. **Training and testing**

The dataset is divided into two parts:

* **Training Set** (70%): This subset is used to train the model so it can learn patterns from the data.
* **Testing Set** (30%): This subset is used to evaluate the model's performance on unseen data.

The splitting process ensures that both subsets represent the original dataset's distribution. This is essential for reliable training and evaluation. Using a random\_state ensures consistency. Without it, the data would be split randomly every time, leading to different results.

A screenshot of a computer program

Description automatically generated

Figure: Split data for training and testing

1. **Train and evaluation model**

We use three main models as it helps with word embedding and prediction of emails like spam and no-spam emails, SVM, Logistic regression and random forest.

Below screenshot shows the implementation of random forest, logistic regression, SVM.

A screenshot of a computer program

Description automatically generated

Figure: Evaluation major for spam email detection

1. **Visualization**

Below screenshot shows our visualization of all three models and accuracy, precision, recall and F1 score of these models. We used the best model, which has the best accuracy for spam detection.

A screen shot of a graph

Description automatically generated

Figure: Visualization of all three models with accuracy, precision, F1 and recall

1. **Future predictions and basic UI for spam classification**.

Check the below two screenshots of UI code

A screenshot of a computer program

Description automatically generated

Figure: UI code for spam classification

A screenshot of a computer program

Description automatically generated

Figure: Email Classification using best model

1. **Problems/Errors Encountered and Resolutions**

* **Error 1: File Encoding Issues**
* Problem: Some email files caused encoding errors when read.
* Resolution: Added errors='ignore' in open().
* **Error 2: Data Parsing Challenges**
* Problem: Not all emails had structured headers (e.g., From, To, Subject).
* Resolution: Used regex with fallback mechanisms to handle missing data.
* **Error 3: Not able to process CSV file after bootstrapping**
* Problem: After bootstrapping we were getting null values in some records.
* Resolution: Handles null values using the below code
* # Handle missing values by removing rows with null values
* print("Handling missing values by removing rows with null values...")
* # Check for any null values in required columns
* print("Before dropping null values:")
* print(bootstrapped\_df.isnull().sum())
* **Error 4: Why spam indicators? Why not tf/idf or other vectorizing method**
* Problem: Using tf Idf or other vectorizing method having problem because of word weightage. It gave wrong predictions
* Resolution: Used spam indicators.
* **Error 5: Class Imbalance in Dataset**
* Problem: Spam and ham labels were imbalanced.
* Resolution: Applied stratified sampling during train\_test\_split.

**Note: There are attached CSV output files in the Zip file. With CSV files there is Combined ham and spam zip folder for RAW email data files.**