Spam Email Detection

4AI3 Project Final Report

Janpreet Singh(400467612)
Bachelor of Technology – Software Engineering Tech.
McMaster University
Hamilton, ON Canada
Email: janpreej@mcmaster.ca

Shivam Sachdeva (400422583)
Bachelor of Technology – Software Engineering Tech.
McMaster University
Hamilton, ON Canada
Email: sachds8@mcmaster.ca

Kamaljit Mahal (400467671)
Bachelor of Technology – Software Engineering Tech.
McMaster University
Hamilton, ON Canada
Email: mahalk1@mcmaster.ca

Hemandeep Singh (student number)
Bachelor of Technology – Software Engineering Tech.
McMaster University
Hamilton, ON Canada
Email: @mcmaster.ca

Steven Osanmaz (student number)
Bachelor of Technology – Software Engineering Tech.
McMaster University
Hamilton, ON Canada
Email: @mcmaster.ca

Table of Contents

Abstract

In today's digital landscape, the constant flood of spam emails poses a significant threat to both user experience and online security. This report delves into the development of a spam email detection model utilizing the Multinomial Naive Bayes classifier. By focusing on the intricacies of the algorithmic

framework, data acquisition, preprocessing, and model training, this project aims to provide a comprehensive solution to mitigate the impact of unwanted and potentially harmful emails. The report emphasizes the practicality and real-world applicability of the developed model, showcasing its potential to enhance user security and overall email experience.

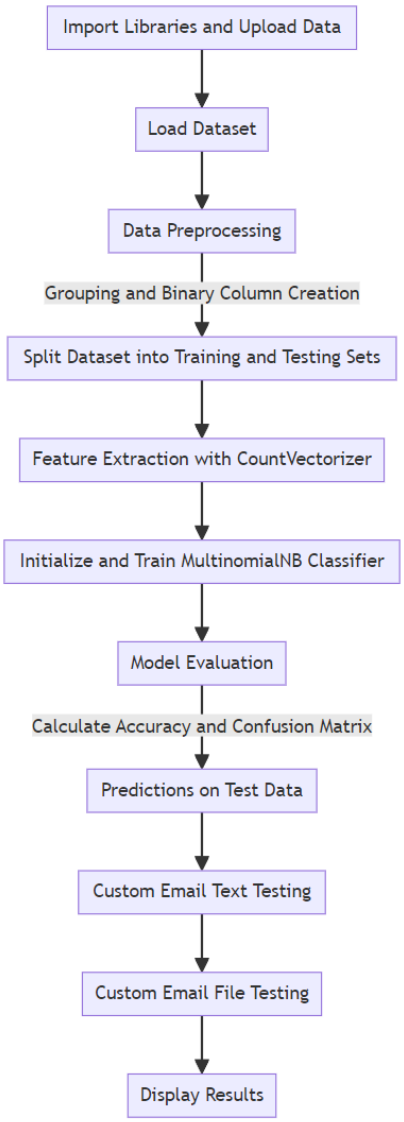
Introduction

Spam emails have become a pervasive issue in the digital age, posing challenges to user convenience and online safety. This report outlines the creation of a spam email detection model leveraging the Multinomial Naive Bayes classifier. The algorithm's effectiveness stems from its ability to discern between spam and ham emails through a meticulous process. This includes data acquisition and preprocessing, text vectorization, model training, and real-world testing. By understanding the nuances of each step, we can appreciate the model's capability to accurately classify emails, thereby improving user security and experience.

How The Algorithm Works

Our spam detection system relies on the Multinomial Naive Bayes classifier, known for its efficiency in text classification. The algorithm distinguishes between spam and ham emails through a sequence of calculated steps:

Algorithmic Framework



Data Acquisition and Preprocessing

The process starts by acquiring a dataset labeled "spam.csv," containing email categorized as 'spam' or 'ham' (non-spam). This dataset serves as the foundation for the algorithm's learning phase. After loading the data into a Pandas DataFrame, a binary column 'is_spam' is created to indicate whether emails are spam (1) or ham (0). This simplifies subsequent processing steps for optimal algorithm performance.

	Label	Message	is_spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0
...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	1
5568	ham	Will ü b going to esplanade fr home?	0
5569	ham	Pity, * was in mood for that. So...any other s...	0
5570	ham	The guy did some bitching but I acted like i'd...	0
5571	ham	Rofl. Its true to its name	0

5572 rows × 3 columns

Text Vectorization

Data Splitting and Text Vectorization:

The dataset is divided into a 75% training set and a 25% testing set for robust model validation. By employing the Count Vectorizer, email texts are converted into a numerical format, producing a sparse matrix of token counts. Each email is transformed into a vector, capturing the frequency of individual words. This process serves as the foundation for our model's interpretation.

Enhanced Feature Extraction through Stop Word Exclusion:

To refine feature extraction, common English stop words are intentionally excluded during vectorization. This deliberate removal of commonplace language elements allows the model to concentrate on more meaningful patterns within the text. This strategy significantly contributes to the overall efficacy of the model, enabling it to discern valuable information from the dataset.

```
[ ]  # Initialize the CountVectorizer
     vectorizer = CountVectorizer(stop_words='english')
     # Store word count data as a matrix
     x_train_count = vectorizer.fit_transform(x_train.values)
```

```
2515          Ok ill send you with in  &lt;DECIMAL&gt;  ok.
674     Ditto. And you won't have to worry about me sa...
996       Change again... It's e one next to escalator...
3337                          Then u go back urself lor...
2503    Ola would get back to you maybe not today but ...
                                  ...
312     Think ur smart ? Win £200 this week in our wee...
2381    If i let you do this, i want you in the house ...
1078                         Yep, by the pretty sculpture
2938    Lol yep did that yesterday. Already got my fir...
2544                        Package all your programs well
Name: Message, Length: 4179, dtype: object
```

```
count                         4179
unique                        3928
top       Sorry, I'll call later
freq                            22
Name: Message, dtype: object
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Hello my name is Shivam.													
This is an excel document for Shivam													
The name Shivam is in the excel document													
With stop words	Hello	my	name	is	shivam	this	an	excel	document	for	the	in	
0	1	1	1	1	1	0	0	0	0	0	0	0	
1	0	0	0	1	1	1	1	1	1	1	0	0	
2	0	0	1	1	1	0	0	1	1	0	1	1	
Without stop words	Hello	name	shivam	excel	document								
0	1	1	1	0	0								
1	0	0	1	1	1								
2	0	1	1	1	1								

Model Training

The core of our spam email detection system relies on the Multinomial Naive Bayes classifier. Naive Bayes is a probabilistic algorithm based on Bayes' Theorem, which calculates the probability of a hypothesis based on prior knowledge of conditions that might be related to the event. In our case, the hypothesis is whether an email is spam or ham, and the conditions are the words present in the email.

General Bayes' Theorem is expressed as:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

In the context of our model:

$$P(\text{spam}|\text{words}) = \frac{P(\text{words}|\text{spam}) \cdot P(\text{spam})}{P(\text{words})}$$

$$P(\text{ham}|\text{words}) = \frac{P(\text{words}|\text{ham}) \cdot P(\text{ham})}{P(\text{words})}$$

P(spam|words): is the probability that an email is spam given the words in the email.

P(ham|words): is the probability that an email is ham given the words in the email.

P(words|spam) and **P(words|ham):** are the likelihoods of observing the words given the class.

P(spam) and **P(ham)**: are the prior probabilities of spam and ham.

P(words): is the probability of observing the words in the email.

The naive assumption in Naive Bayes is that the presence of a particular word in an email is independent of the presence of other words.

Multinomial Naive Bayes Framework:

1. **Prior Probability Calculation:**

   Calculate the prior probabilities P(spam) and P(ham) based on the proportions of spam and ham emails in the training dataset.

2. **Likelihood Calculation:**

   For each word in the vocabulary, calculate the likelihood of observing that word given the class (spam or ham).

   P(word|spam) is the probability of observing the word in a spam email.

   P(word|ham) is the probability of observing the word in a ham email.

3. **Posterior Probability Calculation:**

   In this step, we use Bayes' Theorem to calculate the probability that an email belongs to a specific class (spam or ham) given the observed words in the email. This is done for both spam and ham classes.

4. **Class Assignment:**

   The class label (spam or ham) is then assigned based on the higher posterior probability. For example:

   If P(spam|words)>P(ham|words), the email is classified as spam.

   If P(spam|words)<P(ham|words), the email is classified as ham.

```
# Initialize the Multinomial Naive Bayes classifier
classifier = MultinomialNB()
# Train the classifier using the training data
classifier.fit(x_train_count, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

Prediction and Evaluation

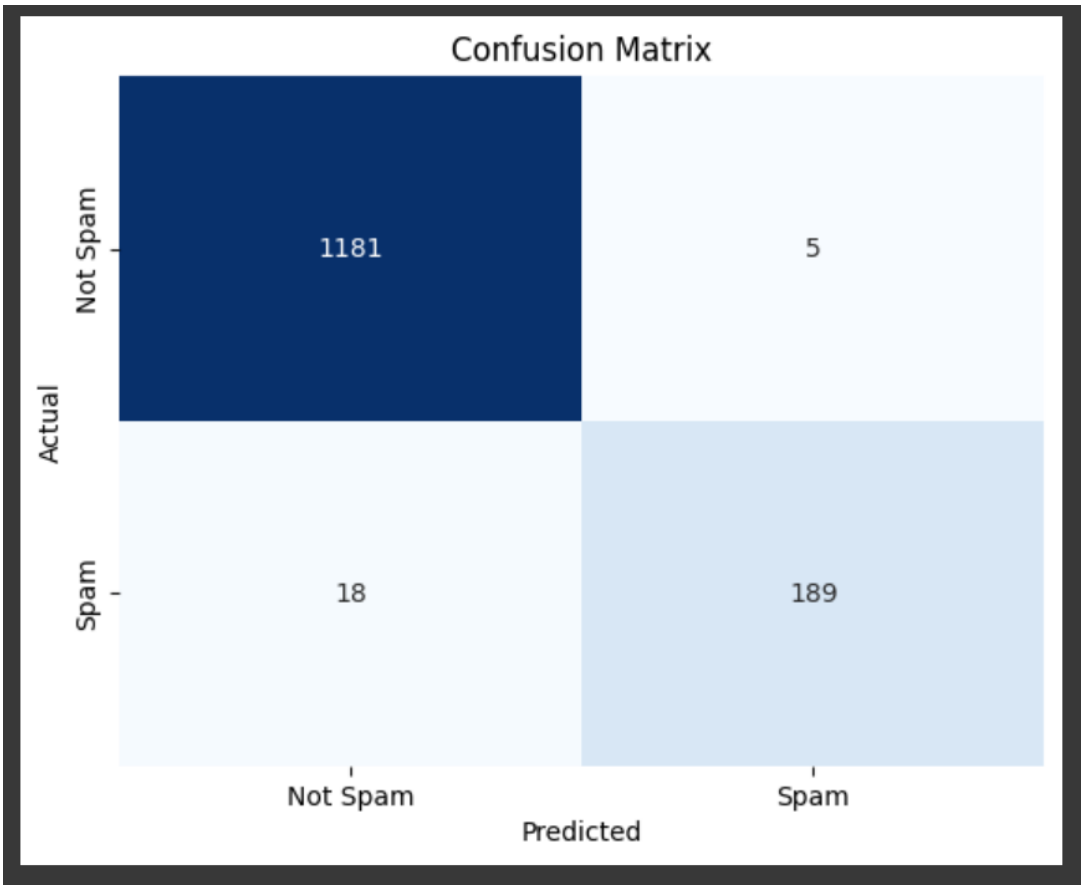
Following the training phase, the model uses the testing set to make predictions, and its performance is evaluated using the classifier.score() method. This method takes the transformed test data, represented as 'x_test_count,' predicts labels using the trained classifier, and compares these predictions with the true labels ('y_test'). The resulting accuracy score reflects the proportion of correctly predicted instances among the total number of instances in the test set. This evaluation approach provides a quantitative measure of the model's proficiency in correctly classifying previously unseen data, contributing to a holistic assessment of its predictive capabilities.

```
# Calculate and print the accuracy of the classifier based on the test data
x_test_count = vectorizer.transform(x_test)
accuracy = classifier.score(x_test_count, y_test)
print(f"Accuracy: {accuracy:f}")
```

```
Accuracy: 0.983489
```

Confusion Matrix Visualization

To gain insights into the model's performance, a confusion matrix is generated and visualized using seaborn and matplotlib. It makes predictions on the test data, calculates the confusion matrix by comparing predictions with actual labels, and creates a heatmap for a clear representation. This resulting visualization provides insights into the model's accuracy in classifying spam and non-spam instances by detailing true positives, false positives, true negatives, and false negatives. The detailed breakdown of these components in the confusion matrix allows for a nuanced evaluation of the classifier's ability to discern between spam and non-spam instances, offering a comprehensive view of its strengths and weaknesses.



Real-world Testing

The code provided below evaluates the model's practicality by testing its performance on both custom email texts and files, assessing how well it performs on data it has not encountered during the training phase. The custom email text and file content undergo the same transformation process used in training, enabling the model to predict whether they are spam or not. This step gauges the model's generalization to diverse input sources, providing a crucial assessment of its effectiveness on previously unseen data and reinforcing its utility in real-world scenarios.

```
# Test the model with a custom email text
test_email = ["Hello! How are you?"]
test_email_count = vectorizer.transform(test_email)
test_email_prediction = classifier.predict(test_email_count)
# Print the prediction
if test_email_prediction[0] == 1:
    print("Spam detected!")
else:
    print("Not spam")
```

```
Not spam
```

```
# Test the model with a custom email text
test_email = ["Congratulations! Your credit score entitles you to a no-interest Visa credit card. Click here to claim"]
test_email_count = vectorizer.transform(test_email)
test_email_prediction = classifier.predict(test_email_count)
# Print the prediction
if test_email_prediction[0] == 1:
    print("Spam detected!")
else:
    print("Not spam")
```

```
Spam detected!
```

```
File: email1.txt
Subject: Meeting Confirmation

Dear Michael,

I hope this email finds you well. I am writing to confirm our scheduled meeting on November 7, 2023 at 7:00pm in Toronto. Please let me know if there are any changes or if you have any specific agenda items you'd like to cover during the meeting.

Looking forward to our discussion.

Best regards,
Trevor Phillips
Not spam
```

```
File Content:
['Subject: Urgent Action Needed\n\nDear Michael,\n\n We have detected suspicious activity on your Wells Fargo account. Log in at [Link] to update your account preferences and protect your information.

File: email2.txt
Spam detected!
```

Conclusion

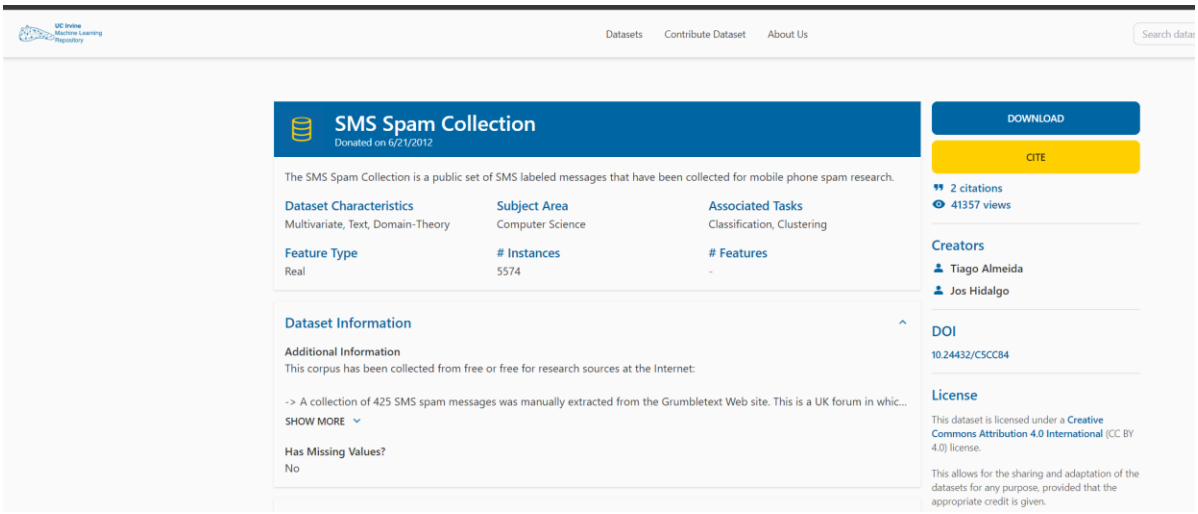
In conclusion, the developed spam detection model showcases a pragmatic approach to categorizing emails as spam or ham, demonstrating its potential for real-world applications. By systematically addressing each phase, from data acquisition to model training, the model offers an efficient solution for filtering unwanted emails in users' inboxes. The project underscores the importance of robust algorithms in enhancing cybersecurity and user experience in the face of evolving digital threats.

References

Dataset Source:

Almeida,Tiago and Hidalgo,Jos. (2012). SMS Spam Collection. UCI Machine Learning Repository. https://doi.org/10.24432/C5CC84.

https://archive.ics.uci.edu/dataset/228/sms+spam+collection



Naïve Bayes:

Ray, S. (2023, November 6). *Naive Bayes classifier explained: Applications and practice problems of naive Bayes classifier*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#:~:text=It%20is%20a%20classification%20technique,presence%20of%20any%20other%20feature

Count vectorization:

Bhalala, P. (n.d.). Countvectorizer in NLP - Pianalytix: Build real-world tech projects. https://pianalytix.com/countvectorizer-in-nlp/

scikit-learn.org. (n.d.). *Sklearn.feature_extraction.text.CountVectorizer*. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Jain, P. (2021, June 1). *Basics of countvectorizer*. Medium. https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c

Confusion matrix:

Narkhede, S. (2021a, June 15). *Understanding confusion matrix*. Medium. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

scikit-learn.org. (n.d.-b). *Sklearn.metrics.confusion_matrix*. scikit. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html