**EDA and Classification of E-Mails as spam or Non-spam**

**Project Members**

Deeksha Chinthapatla

Sai Kumar Reddy Ramasahayam

Bhanu Pranav Pandirla

Purushotham Reddy Kankanala

# Abstract

Anyone with an active email today is likely to receive at least one spam email over time. Spam email, also known as junk email, is an unsolicited email that is sent in bulk to a large number of recipients. Spam emails may contain malicious links, scams, or other misleading concepts. Spam emails can also be sent by humans, but bots are far more common. Most email clients, such as Gmail and Microsoft Outlook, have built-in spam filters that scan incoming messages for specific words or phrases (Sharma & Sahni, 2011). Examples of spam emails include deceptive marketing materials, unwanted message forwarding, and spoofing attempts. These filters keep unwanted messages out of your inbox, but spam messages are so well hidden that they sometimes sneak into your main inbox instead of your spam folder. Opening spam email attachments or clicking links can put your computer and personal information at risk with a variety of malicious software (Alghoul et al., 2018).

With this in mind, this study seeks to improve the accuracy of existing models to significantly, if not completely, eliminate the remaining percentage of spam that reaches inboxes. is intended to develop It is important to take extra precautions to protect your device, especially when storing or processing personal data.

**Goal and Objectives :**

**Motivation**

Sending and receiving spam emails not only wastes time but also wastes disk space and bandwidth. In recent years, the problem of spam emails has become more serious. According to Rayan & Taloba (2021), more than 40% of all emails, or 15.4 billion emails per day, are considered spam, costing Internet users about $355 million annually. . These unwanted messages have already caused many problems such as B. Having to expend time and energy to delete received messages or prevent them from reaching users, clogging mailboxes and wasting network bandwidth. Due to the large number of issues related to spam emails, I was inspired to check spam emails to discover numerous approaches to research and fix issues. The purpose of spam filtering is to automatically remove unwanted e-mails from a user's mail stream. This can be achieved using a filter.

**Significance**

Given that a user's inbox can be filled with spam emails that do nothing, this leads to a waste of the user's time, energy, and various other basic resources. (Zhao & Zhang, 2015). Therefore, the significance of this work is to address significant resource waste by using different ensemble techniques in machine learning techniques to improve the accuracy of models such as Naive Bayes and Support Vector Machines. . After identifying each spam email individually, these improved algorithms help filter out spam emails with precision.

**Objectives**

This study presents a new approach to spam filtering based on preserving the chronological order of word occurrence with data mining techniques. Learn how to determine which messages are spam and how to provide a record for use in investigations. Various ensemble classification approaches have been investigated and practiced in the context of spam filtering to improve the accuracy of classification models. Next, we describe the corresponding classification model. The system consists of four parts: spam data collection, classification model, trained classification model, and classification results.

**Features**

For the classification and modeling of decision trees and other classification models, we will use the dataset titled "Spam-Base Dataset" published on the Kaggle website [here]. This data collection includes email messages and labels that define whether the messages are spam or ham. Full data sets are provided separately as test and training sets. The training set has 957 observations and the training set has 125 observations. The algorithm is based on word counting processes and dynamics. So, in its simplest sense, you can distinguish unique words based on the number of times they appear in your email inbox. Therefore, a special function could be used to distinguish between identical and repeated words. After reading the information contained in the dataset, we used the Python environment provided by Jupyter to view word and letter frequencies and implement various classification algorithms.

**Block Diagram:**

# Diagram Description automatically generated

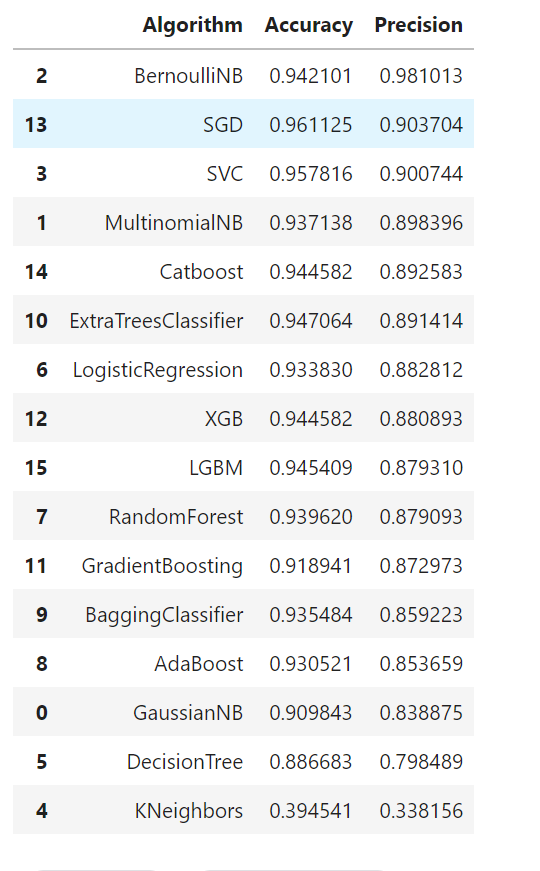
**Related Work (Background)**

Previously there have been many articles and models related to this work. These models are built using various techniques such as probabilistic models using basic Machine learning, Deep learning models and transformers etc.

We have gone through various articles and previous papers to understand the base functionality

As a result, we concluded that these methods lack clear preprocessing.

These are the results of the paper that we took:



**Dataset**

The dataset that has been used for this work was acquired from the public resource that is Kaggle.

The dataset can be found [here](https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset).

This dataset contains two attributes such as :

The body of the email and the label denoting that it is spam or ham

The total number of tuples is 6046.

**Detailed design of Features**

The feature that is present in the dataset is the body of the email. So we have decided to process the given data in various stages.

**Step 1: Contraction Mapping / Expanding Contractions**

Contractions are basically words that we normally write using an apostrophe. “ain’t” or “aren’t” are few examples of contractions. Since we are going to standardize our text, we need to expand these contractions. We will be adding a new column called “no\_contract” to our data frame and apply a lambda function to the "msg" field which results in expanding any contractions.

**Step 2: Tokenization**

Tokenization is the process we'll use to divide the corpus into a vocabulary of distinct terms. We can also able to tokenize individual terms and generate a bag of words model. You may have noticed that this model has a clear flaw: it cannot able to accurately represent the innate structure of human language. For the purpose of tokenization we can use NLTK, a platform to build programs on python to work with human language data

**Step 3: Noise Cleaning - spacing, special characters, lowercasing**

A lot is going on here: digits, gratuitous whitespace, and all varieties of punctuation. Some terms are randomly capitalized, and others are in all caps. Since these terms might show up in any one of the training examples in countless forms, we need a way to ensure each training example is on an equal footing via a preprocessing step called normalization. This form of noise-cleaning takes care of spacing and any special characters.

A common step in pre-processing is, converting all the words to lowercase. So here we will again add a new column called “lower” to our data frame which is going to convert all the tokenized words to lowercase. To iterate over multiple words, a for-loop within a lambda function is used to apply the “lower” function to each and every word.

we'll remove all punctuation since they add no value when we start to analyze our data. By following the previous steps we will create a new data column with all the punctuation removed. We are going to use a for-loop within a lambda function again to iterate over the tokens but this time we will be using an IF condition to output alpha characters

**Step 4: Spell Checking**

For spell-checking, we will use Microsoft's TextBlob, which is a simple spelling correction mechanism

**Step 5: ‘Stop Words’ Identification**

Some words in the English language, while necessary, don't contribute much to the meaning of a phrase. These words, such as "when", "had", "those" or "before", are called stop words and should be filtered out.

We add a new data column called “no\_stopwords” that removes all the stopwords from the “no\_punc” column since it has already been tokenized, converted to lowercase, punctuation was removed. Same as in previous steps we will use a for-loop within a lambda function that iterates the tokens in “no\_punc” and it only returns the tokens which are not available in our “stop\_words” variable.

**Step 6: Stemming/Lemmatization**

The main aim of stemming is to convert various forms of words to their root word. For example, “ride”, “rode”, “riding”, “rider” are the derivatives of the word “ride” and normally researchers try to remove this type of variability from the corpus. When compared to lemmatization, stemming is a bit the less complicated procedure but it not always provide a dictionary-specific root of the word. For example, if we want to stem a word called ‘leaves’ it will produce ‘leav’ which is not a correct word. whereas in lemmatization it will produce a word called ‘leaf’, which has meaning to it.

So Instead of stemming, we will apply lemmatization to our data but it requires few additional steps compared to stemming. First, we need to apply parts of speech tags, i.e parts of speech ( noun, verb, adverb, etc.) to each and every word

**Step 7: Tokenization/Vectorization (Countvectororiser/TFID)**

Tf-idf is one of the best metrics to determine how effective a term is to a text in a series or a corpus. tf-idf is a weighting system that assigns a weight to each word in a document based on its term frequency (tf) and the reciprocal document frequency (tf) (idf). The words with higher scores of weight are deemed to be more significant.

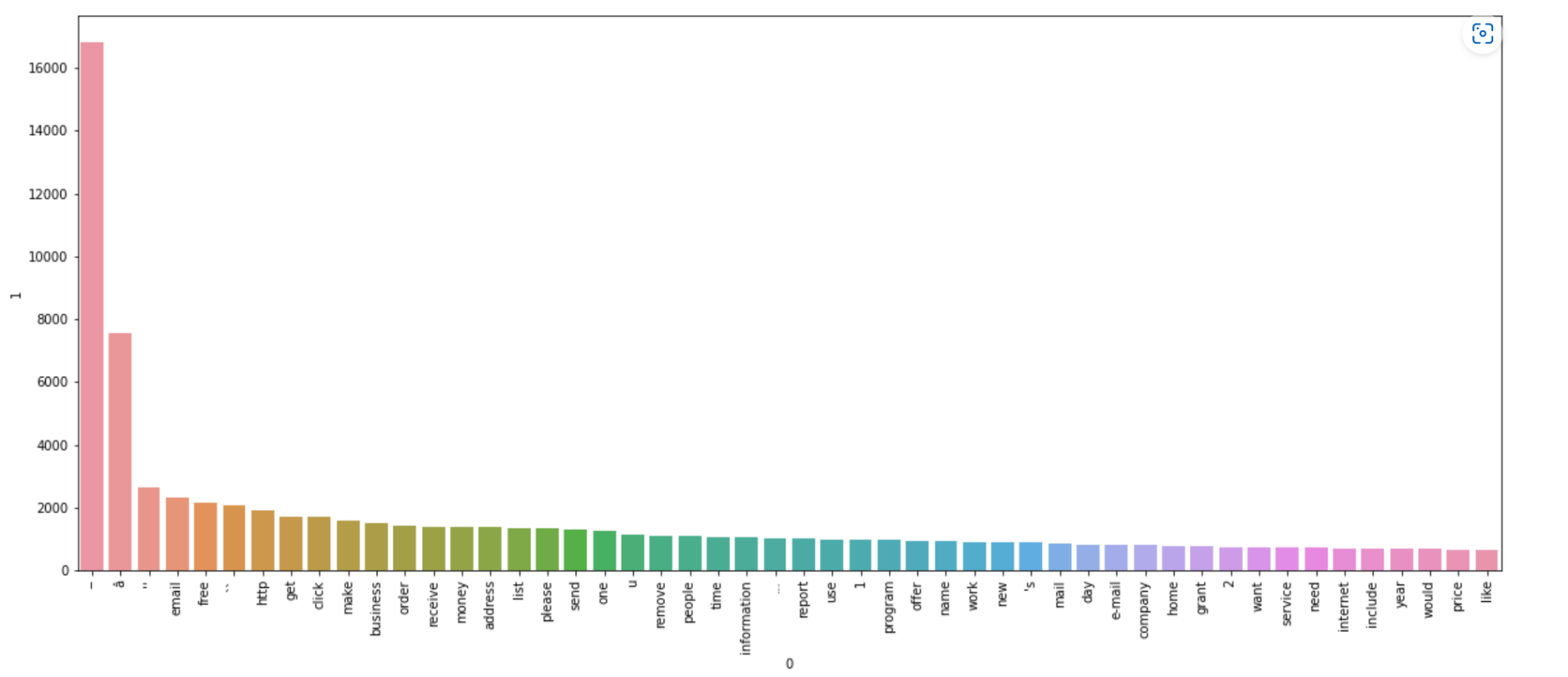
After the completion of the preprocessing.

We used various models and compared their results and tried to include hash vectorization and it worked to improve the accuracy of the model.

**Analysis**

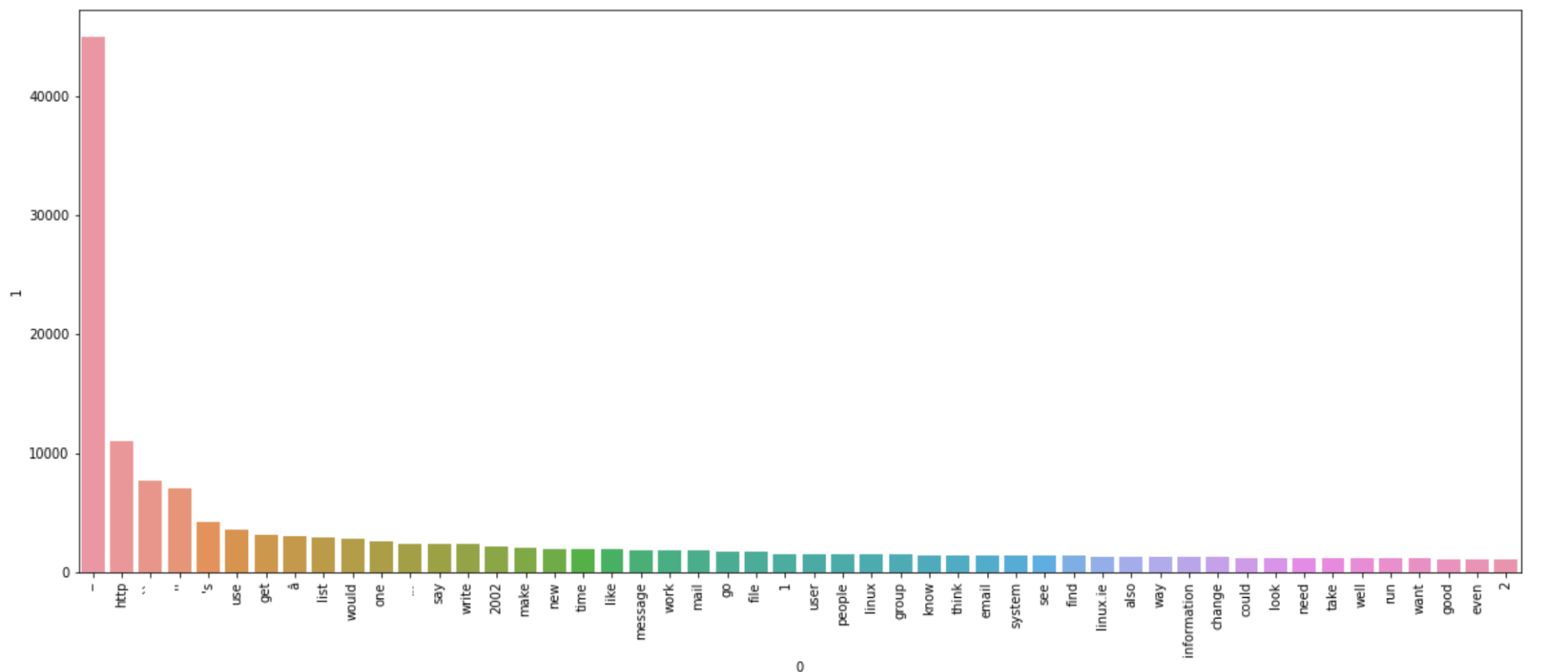
We analyzed the dataset and used various plots to visualize and interpret it.

These are some of the words which make your mail spam.



email, free, HTTP, get, click, make, business, order, receive, money, address, list, please, send, one,u, remove, people, time, information, report, use,1, program, offer, name, work, news, mail, day, e-mail, company, home, grant,2, want, service, need, internet, include, year, would, price, like”

**Similarly with ham.**

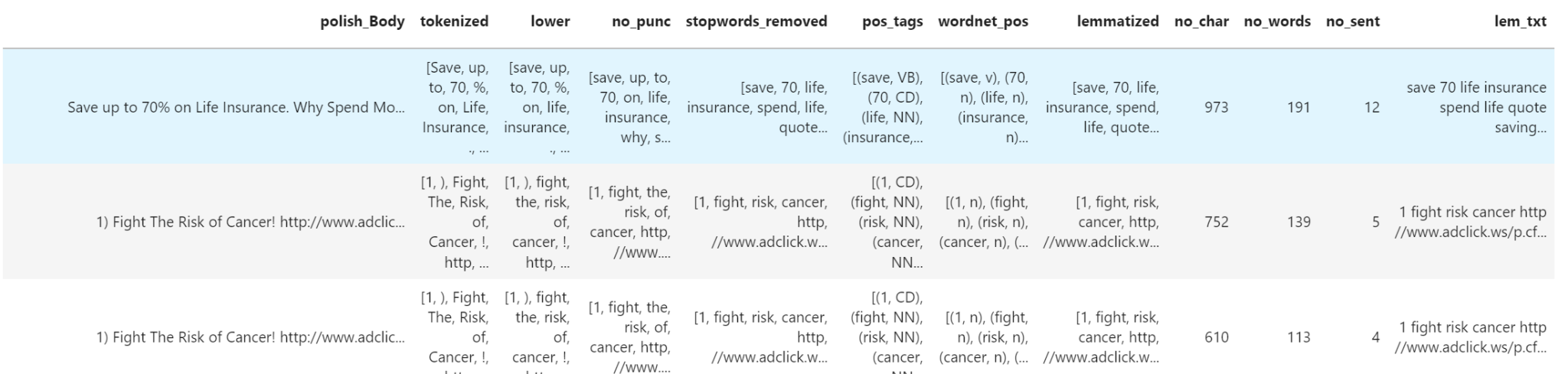


use, get,â, list, would, one, say, write,2002, make, new, time, like, message, work, mail, go, file,1, user, people,linux, group, know, think, email, system, see, find, Linux. ie, also, way, information, change, could, look, need, take, well, run, want, good, even

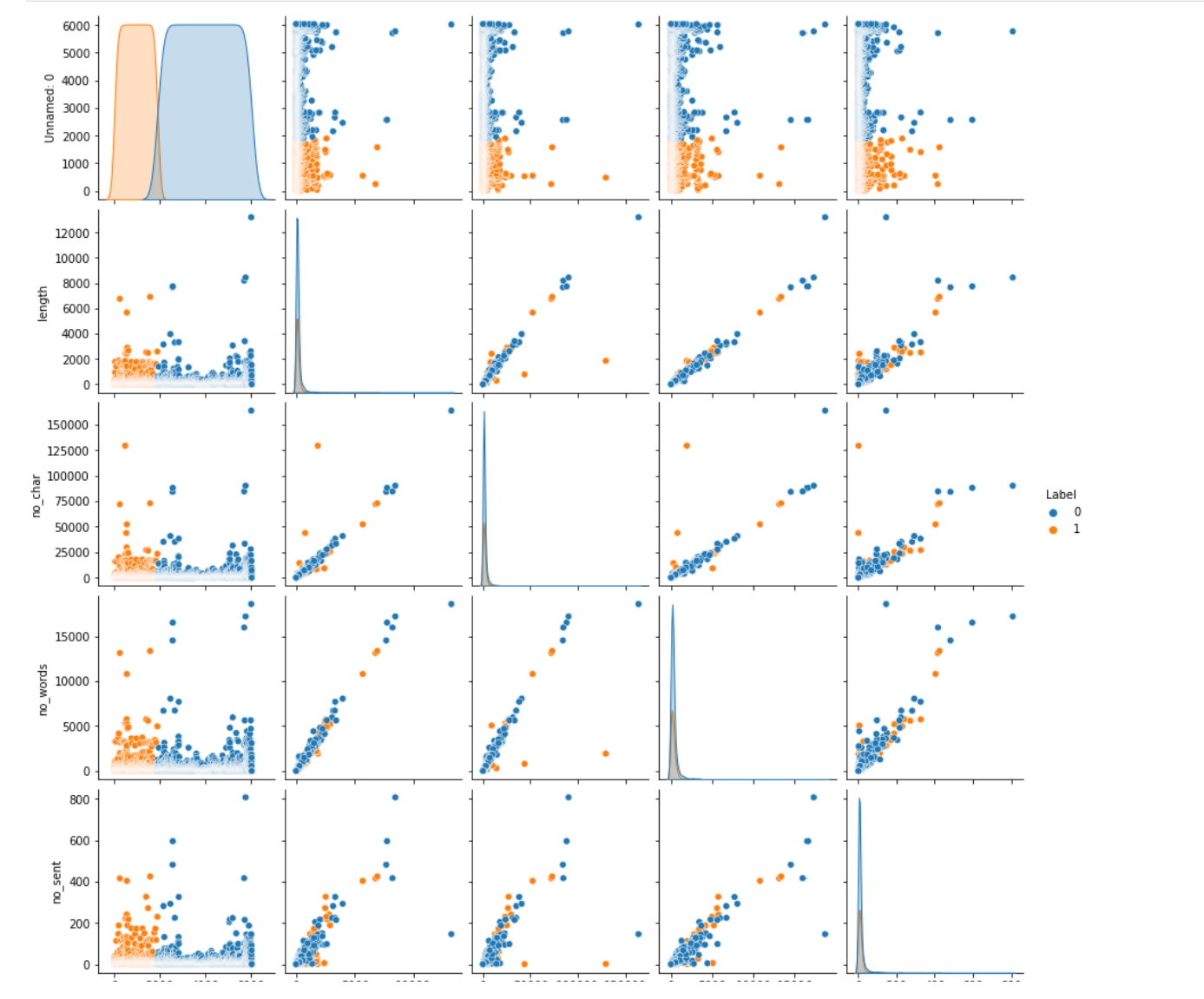
**Implementation**

We have implemented all the steps mentioned above.

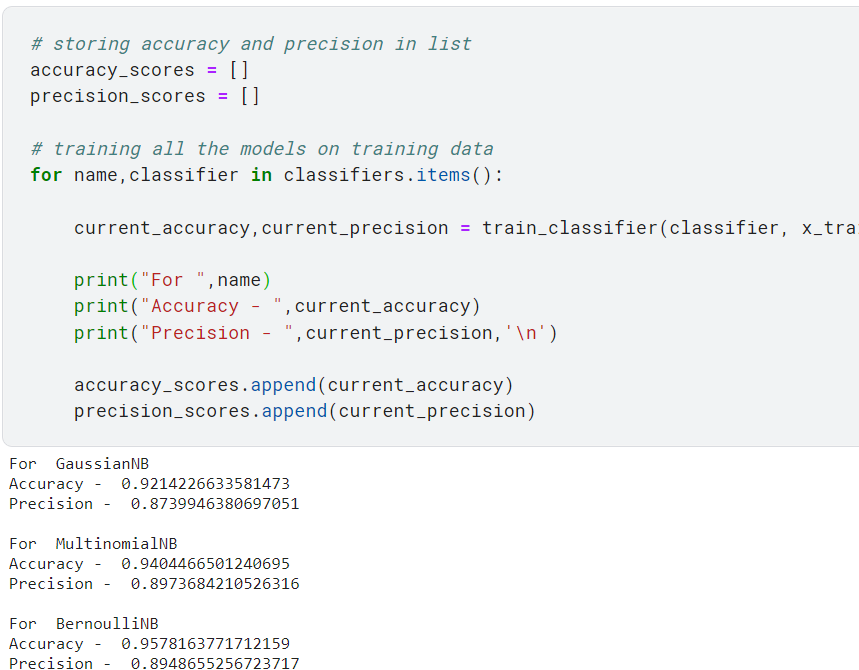
**Data Preprocessing :**



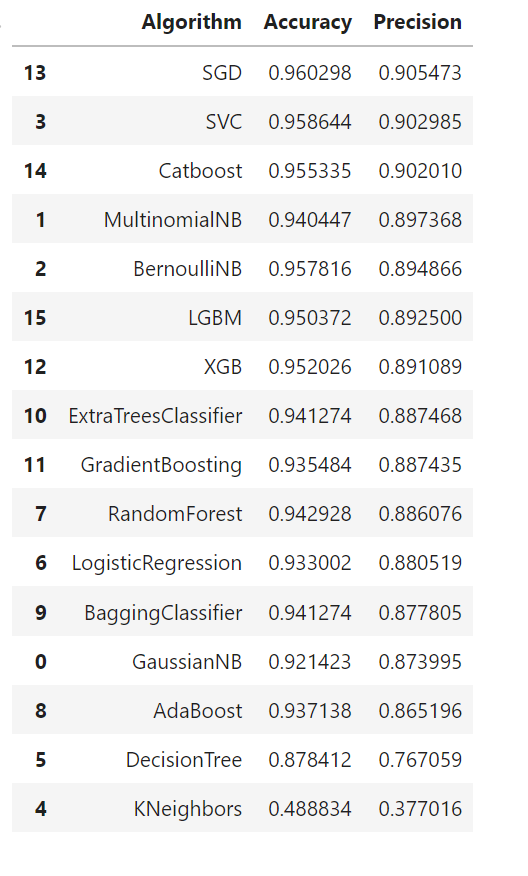
**Data interpretation :**



**Training and saving the model**



**Preliminary Results**



And we tried to add HashingVectorizer before TFID and it gave greater results than the base Decision tree model.



At last, we saved the model for future deployment.



**Project Management**

**Implementation status report ▪**

**Work completed:**

* **Description**

We have examined numerous datasets and found a suitable dataset for our project. We analyzed the whole dataset and performed pre-processing on it by using NLP techniques.

We nearly used 10 classifiers to compare our results. We also used ‘hash vectorization’ to increase the model’s accuracy.

* **Responsibility**

We have divided our project into various portions and stages which made it more effective.

The various stages include of :

1. Collected various articles and reports regarding the project and analyzed them**. [ All Members ]**
2. Explored various datasets and compared them to find the most relevant dataset. **[ Purushotham Reddy Kankanala]**
3. Then we analyzed the dataset and extracted useful details regarding the dataset. **[ Sai Kumar Reddy ramasahayam ]**
4. Performed necessary pre-processing functions to make the data more precise and clean**. [ Deeksha Chinthapatla, Bhanu pranav ]**
5. Explore and list out more than 10 classifiers to implement and compare the results- **[ Sai kumar reddy, Purushotham reddy Kankanala ]**
6. We have also included the method of hash vectorization to the pipeline to increase the accuracy of the model at the end. – **[ Bhanu pranav, Deeksha chinthapatla ]**
7. Stored the base model weights for deployment and prediction. – Bhanu Pranav

* **Contribution**

Purushotham Reddy Kankanala:23%

Sai Kumar Reddy Ramasahayam:25%

Deeksha Chinthapatla:26%

Bhanu Pranav:26%

**Work to be completed :**

**Description**

The future scope of this project is to analyze which part of the email was used to classify it as spam or ham.

And edit the email such that it converts the spam mail to unspam using Natural language processing. We need a rigorous amount of research using relevant articles and papers.

* **Responsibility**

**Purushotham Reddy Kankanala :** Still need some minor correction in the pre-processed text ,benchmark the current results and save them for future use.

**Sai kumar reddy ramasahaym:** Try to retrieve the weights from the ML model such that we can calculate which feature helped us to get current classification.

**Sai kumar reddy ramasahaym:** If the above method fails we will build an simple feed forward neural network and extract the weightage of each feature such as heatmaps in CNN.

**Deeksha Chinthapatla:** After retrieving try to use NLP process to reverse engineer the message and convert it to HAM if it is SPAM.

**Bhanu Pranav:** At last we will build an API using saved model and deploy it to an local webserver

* **Issues/Concerns**

The issue might be the limitation of Machine learning. And intercepting the intermediate layers to find the weightage of each word.

# References

[Ms.D.Karthika Renuka1 ,Dr.T.Hamsapriya2 , Mr.M.Raja Chakkaravarthi3 ,Ms.P. Lakshmi Surya4 Spam Classification based on Supervised Learning using Machine Learning Techniques](https://www.readcube.com/articles/10.21917%2Fijct.2011.0064)

[Alghoul, A., Al Ajrami, S., Al Jarousha, G., Harb, G., & Abu-Naser, S. S. (2018). Email classification using artificial neural network.](https://www.researchgate.net/publication/329307944_Email_Classification_Using_Artificial_Neural_Network)

[Rayan, A., & Taloba, A. I. (2021). Detection of Email Spam using Natural Language Processing Based Random Forest Approach.](https://www.researchgate.net/publication/354886499_Detection_of_Email_Spam_using_Natural_Language_Processing_Based_Random_Forest_Approach)

[Sharma, A. K., & Sahni, S. (2011). A comparative study of classification algorithms for spam email data analysis.](https://www.researchgate.net/publication/322129882_Comparative_Analysis_of_Classification_Algorithms_for_Email_Spam_Detection) *[International Journal on Computer Science and Engineering](https://www.researchgate.net/publication/322129882_Comparative_Analysis_of_Classification_Algorithms_for_Email_Spam_Detection)*[,](https://www.researchgate.net/publication/322129882_Comparative_Analysis_of_Classification_Algorithms_for_Email_Spam_Detection) *[3](https://www.researchgate.net/publication/322129882_Comparative_Analysis_of_Classification_Algorithms_for_Email_Spam_Detection)*[(5), 1890-1895.](https://www.researchgate.net/publication/322129882_Comparative_Analysis_of_Classification_Algorithms_for_Email_Spam_Detection)

[Zhao, W., & Zhang, Z. (2015). An email classification model based on rough set theory. In *Proceedings of the 2005 International Conference on Active Media Technology, 2005. (AMT 2005).* (pp. 403-408). IEEE.](https://ieeexplore.ieee.org/document/1505383)