**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

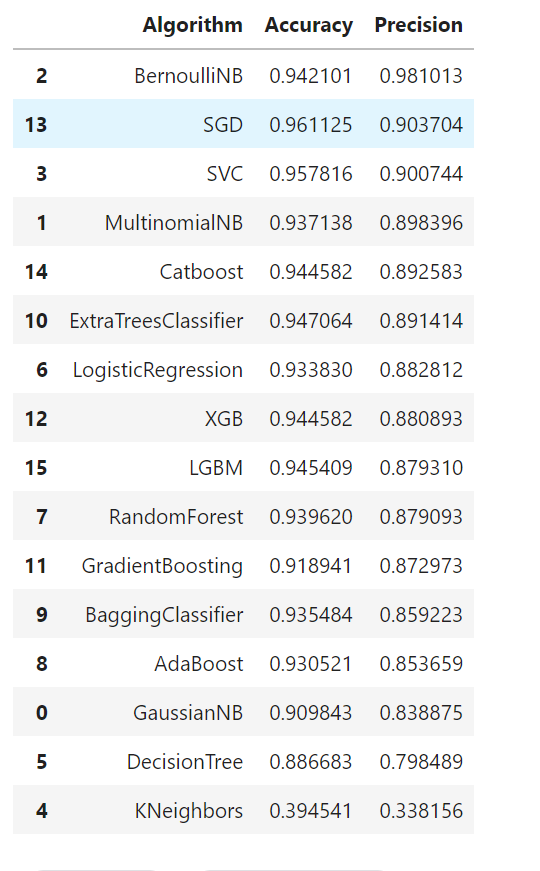
**Related Work (Background)**

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.

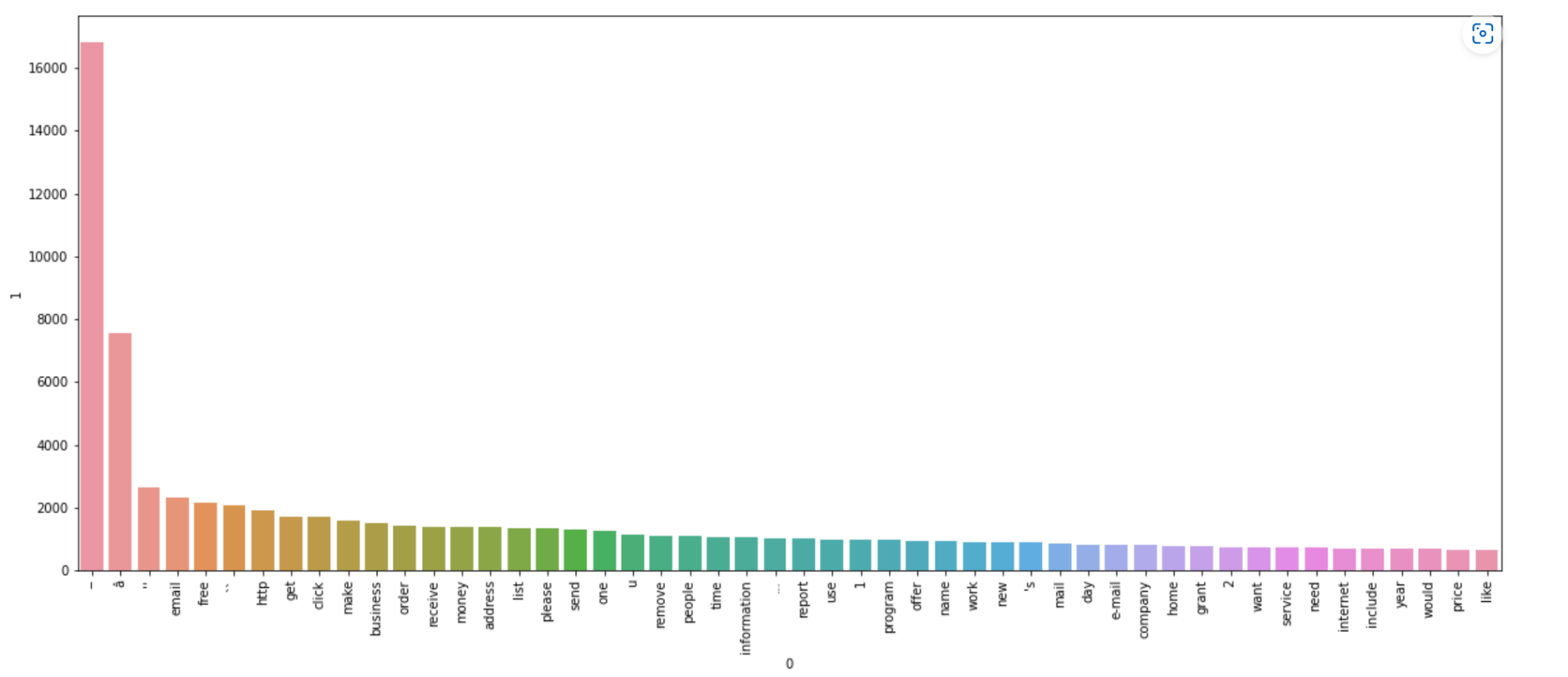
After the completion of the preprocessing.

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**

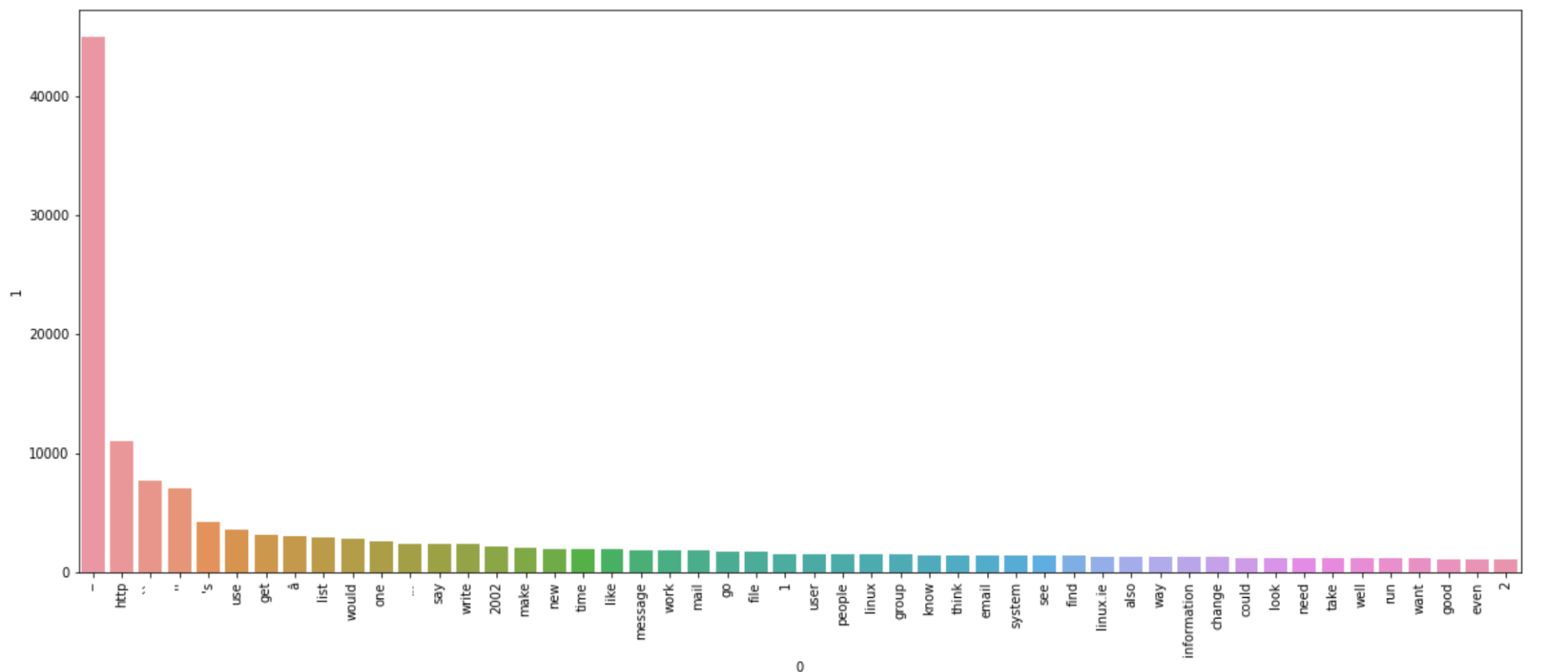
We analyzed the dataset and used various plots to visualize and interpret it.

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.**

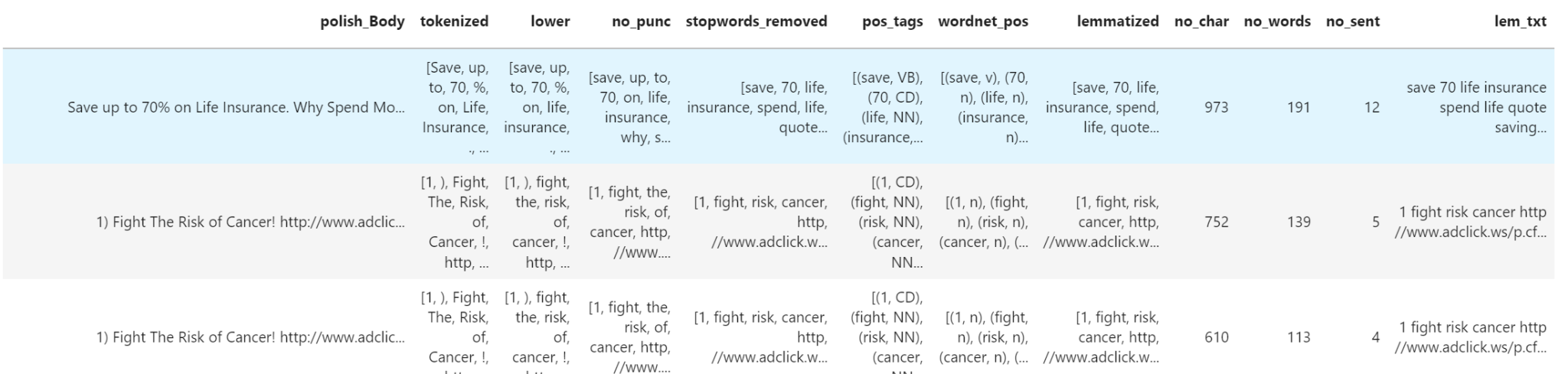


use, get,â, list, would, one, say, write,2002, make, new, time, like, message, work, mail, go, file,1, user, people,linux, group, know, think, email, system, see, find, Linux. ie, also, way, information, change, could, look, need, take, well, run, want, good, even

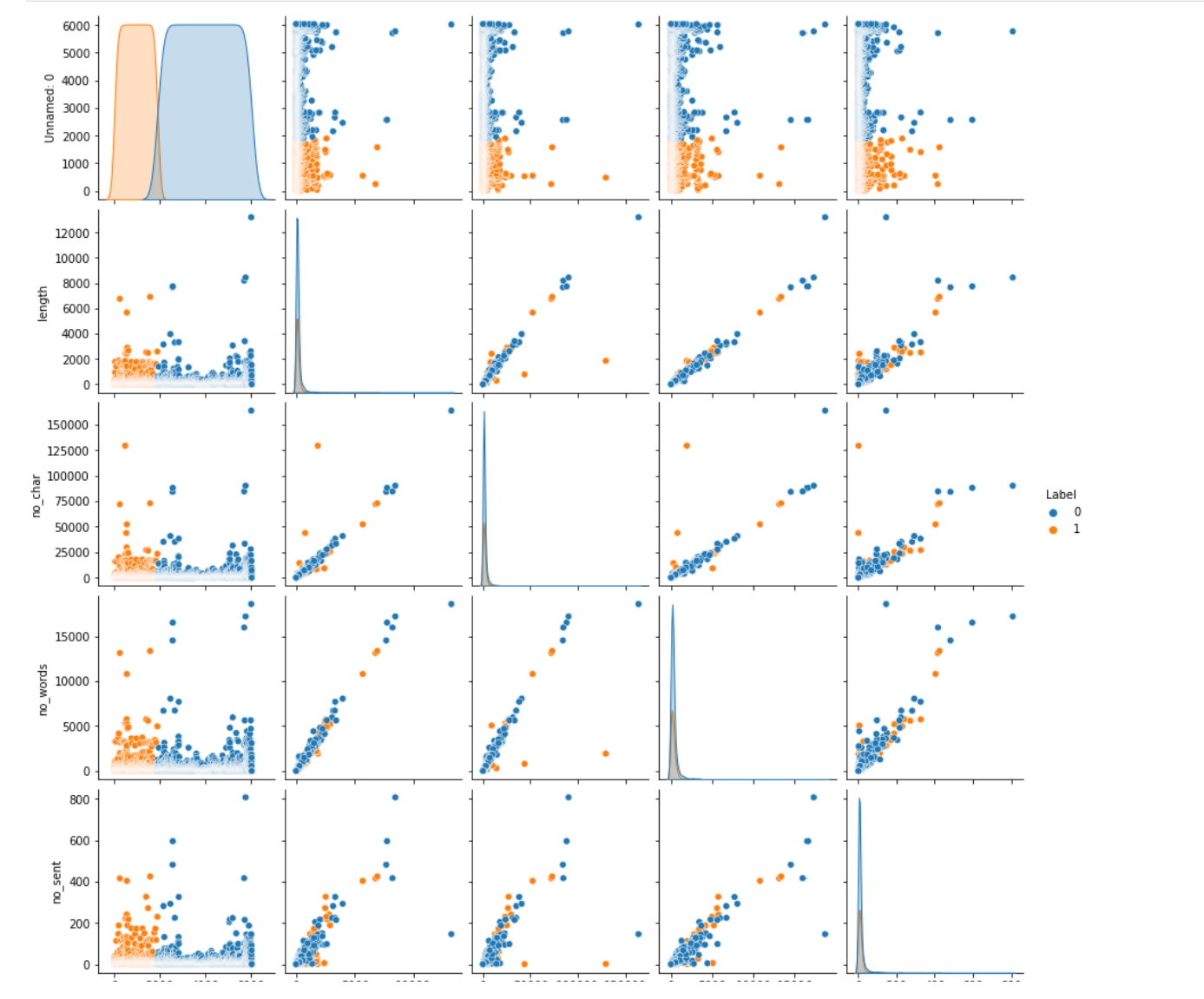
**Implementation**

We have implemented all the steps mentioned above.

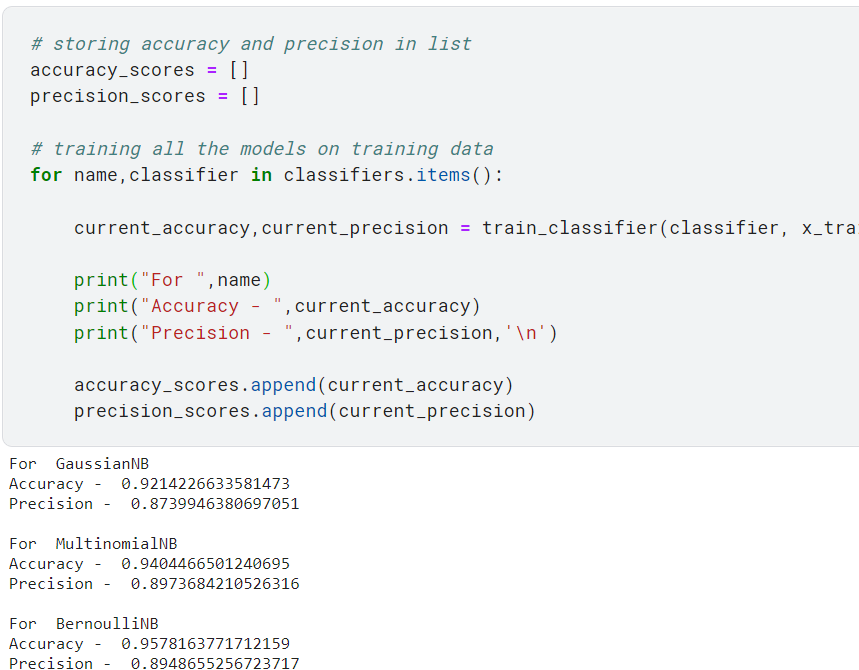
**Data Preprocessing :**



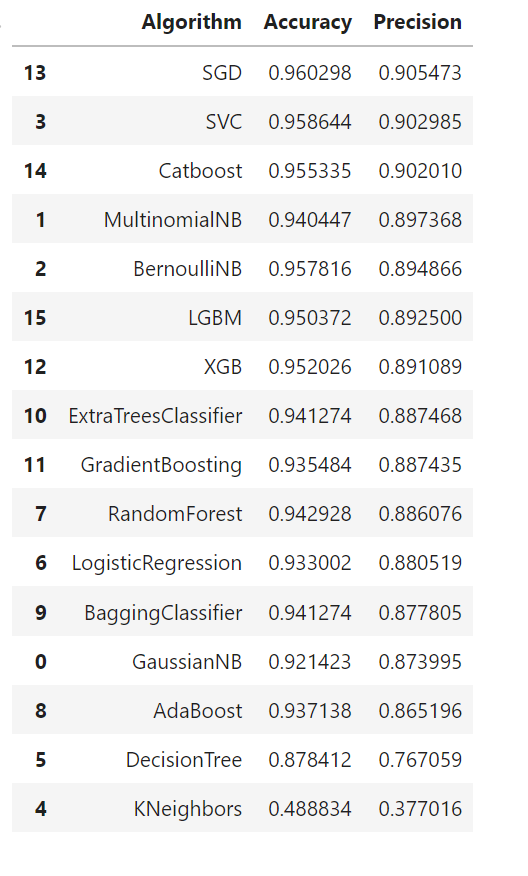
**Data interpretation :**



**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.

# Increment 2

# Related Work (Background)

No previous work was found on this idea. But we got a concept of finding the most important words that cause the email to be spam.

The main idea of extracting the main features from the input was gained from the CNN models. We use the Grad-Cam algorithm,

Ex:

Graphical user interface, application

Description automatically generated

In the above example.

We provide a normal image to the model now using the Grad-Cam algorithm we highlight the pixels that are used to classify the current image as cat.

The above example also shows the highlighted pixels that result in this prediction.

# Text, website Description automatically generated with medium confidence

Similar to the grad cam algorithm we have:

# Saliency Map

A saliency map is an explanatory method used to interpret the predictions of a convolutional neural network (CNN). This is probably the oldest and most commonly used interpretation method in deep learning. Essentially, the saliency map of the input image specifies the parts that contribute most to determining the activity of a particular layer in the network or the network as a whole. There are three main ways to obtain the saliency map of the input image for CNN. The first proposed approach uses a deconvolution network introduced by [Zeiler and Fergus 2013]. To identify the features (essentially pixels) in the input image that the hidden layers of the network are looking for, the authors of this paper proposed a deconvolution network that reconstructs the input from its layer activations. To do this, the operations performed between the input and that particular level are reversed.

# 

# Grad-CAM

Class Activation Map (CAM) is another description method used for CNN [Zhou et al. 2016]. The authors of this paper examined networks with the similar architectures to networks of network architecture. In these networks, the stack of fully connected layers at the end of the model has been replaced with a layer called Global Average Pooling (GAP). GAP simply averages the activations of each feature map, concatenates these averages and outputs them as a vector. The weighted sum of this vector is then fed to the final softmax loss layer. Using this architecture, we can highlight important regions of the image by back projecting the output weights onto the convolutional feature map. The following diagram illustrates this process.

Graphical user interface, application

Description automatically generated

We would try to extract the most important features like in grad cam and remove them to decrease the confidence of the model while classifying the model as spam.

At last, we got two implementations to go with:

1. Collecting all the words that cause a message to be spam or ham. And try to experiment with that.
2. Finding the weightage of all the words in the message which causes the following message to be spam or ham

**Your Model**

Architecture Diagram with Explanation

Diagram

Description automatically generated

The model mentioned above states that:

1. The data used in this model is a regular email text.
2. Then the message is passed through a preprocessor.
3. Then remove all the words from the email that causes the mail to be spam.
4. Prior to this we train a model to on train data.
5. Feed the modified email to the model to observe the prediction.

Workflow diagram with explanation:

**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

**Analysis of data**

o **Data Pre-processing.**

The feature that we have used here is the body of an email, so we have processed the given emails i.e the data into various stages.

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

Contractions are the words that are written with an apostrophe. Didn’t, don’t are the examples of contractions. As we are standardizing our text, the contractions need to be expanded. A new column named “no\_contract” is added to the data frame and we apply a lambda function has been applied to the msg field which inturn expands the contractions.

**Step 2: Tokenization**

Tokenization is used to divide a given corpus into a set of individual terms of vocabulary. Here we generally tokenize the individual terms into a list of words. This method has a clear flaw i.e., this model cannot accurately represent the innate structure of the human words . To fix this, we use NLTK, a platform based 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.

We have also added a step in preprocessing:

According to previous results still, there is a lot of symbols inside the pre-processed text.

Like:

A picture containing text, antenna

Description automatically generated

We also removed the extra symbols in the emails that do not contribute much in the process of classification.

**Implementation**

**o Algorithms / Pseudocode**

Diagram

Description automatically generated

**Chart

Description automatically generated**

**Algorithm:**

The model mentioned above states that:

1. The data used in this model is a regular email text.
2. Then the message is passed through a preprocessor.
3. Then remove all the words from the email that causes the mail to be spam.
4. Prior to this we train a model to on train data.

Feed the modified email to the model to observe the prediction.

**Pseudocode:**

**l=[]**

**For i in spam\_emails:**

**l.append(most\_used\_spam\_words)**

**For j in spam email processed text:**

**For j in l:**

**Remove j from i**

**Which will result in spam emails to reduce the confidence. And misclassify it as a ham email.**

**Result & Explanation of implementation:**

Graphical user interface, text, application

Description automatically generated

It is 96% accurate.

Not we try to misclassify the spams to hams which will result in a decrease of accuracy.

**Step 1:**

Train the model on currently available data.

**Step 2:**

Now we Build a preprocessor that will help us in pre-processing the data and ease the process of feeding the data to the model for prediction.

**Step 3:**

We examine the original msg and check its prediction:

**Input:**

Text

Description automatically generated with medium confidence

**Output:**

Model output 1 indicates that the input email is spam.

Model output 0 indicates that the input email is Ham.

Without any modification of email, our model predicted it as spam.

Graphical user interface, text

Description automatically generated with medium confidence

**Step 4:**

We now modify the data and feed it to the model: And examine the result.

**Input:**

Text

Description automatically generated

**Output:**

So we were able to manipulate the model to predict that spam email ham

**Graphical user interface, text, application

Description automatically generated**

**Step 5:**

We ultimately implement this on the whole dataset:

In order to compare the result with the initial result.

We successfully dropped the result accuracy from 96% to 65%.

Graphical user interface, text, application, Word

Description automatically generated

And overall accuracy to 65%

Graphical user interface, text, application

Description automatically generated

For comparison of the model that we have built we are using accuracy as a parameter to show that a change in accuracy makes the model misclassify the Spam emails.

**Before Manipulation of the emails:**

**Email before modification:**

**A picture containing text

Description automatically generated**

Graphical user interface, text, application

Description automatically generated

**After manipulation of the emails:**

**Email After modification but the model predicts it as an unspam msg.**

**Text

Description automatically generated with medium confidence**

# Graphical user interface, text, application Description automatically generated

o **Graph model with explanation:**

Initial data representation:

**Graphical user interface, text, application, email

Description automatically generated**

**After various preprocessing methods like:**

**The preprocessing output is:**

**Text

Description automatically generated with medium confidence**

**Output:**

# Graphical user interface, application, Word Description automatically generated

This is the output of the first stage of pre-processing.

Now we preprocess the spam message so that we are able to misclassify spam msg as ham/unspam.

**Email before modification:**

**A picture containing text

Description automatically generated**

**Email After modification but the model predicts it as an unspam msg.**

**Text

Description automatically generated with medium confidence**

In the previous phase, we didn’t do well with finding the best words that help in the classification of the model.

**The frequency plot of the words used in spam.**

**A picture containing text, measuring stick

Description automatically generated**

**The frequency plot of the words used in ham.**

**A picture containing text, measuring stick

Description automatically generated**

**• Project Management**

Implementation status report ▪

**Work 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& Contributions:**

Deeksha& sai kumar reddy : Still need some minor corrections in the pre-processed text , benchmark the current results and save them for future use.

Bhanu pranav & purushotham: Try to retrieve the weights from the ML model such that we can calculate which feature helped us to get the current classification.

Bhanu Pranav:If the above method fails we will build a simple feed-forward neural network and extract the weightage of each feature such as heatmaps in CNN.

Sai kumar Reddy & Purushotham: And After retrieving try to use the NLP process to reverse engineer the message and convert it to HAM if it is SPAM.

Deeksha Chinthapatla: 24%

Sai Kumar Reddy Ramasahayam:26%

Bhhanu Pranav Pandirla: 28%

Purushotham Kankanala:22%

**• Issues/Concerns**

Firstly we observed that even after the first phase of the preprocessing the data was not well polished so we used further preprocessing by removing the words that are not useful for prediction.

Secondly, We were unable to retrieve the individual weightage of each word as we have to construct dummies for each word in the processed text. As this led us to enormous storage space. So we avoided that method.

# 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)

[Let Sentiment Classification Model speak for itself using Grad CAM | by gogamza (Heewon Jeon) | Apache MXNet | Medium](https://medium.com/apache-mxnet/let-sentiment-classification-model-speak-for-itself-using-grad-cam-88292b8e4186)

[A Review of Different Interpretation Methods (Part 1: Saliency Map, CAM, Grad-CAM) | by Mohammadreza (Reza) Salehi | Medium](https://mrsalehi.medium.com/a-review-of-different-interpretation-methods-in-deep-learning-part-1-saliency-map-cam-grad-cam-3a34476bc24d)