Spam or Ham? Identifying and Filtering Unwanted Emails using Bayesian analysis

A Project Work for

Postgraduate Diploma in Statistical Methods Analytics

in

Statistics

by

Adarsh Kumar
DSTC-22/23-002

and

Sourav Ghosh
DSTC-22/23-016

to

INDIAN STATISTICAL INSTITUTE
CHENNAI,TAMILNDU,INDIA

May 2023

DECLARATION

We, **Adarsh Kumar (DSTC-22/23-002)** and **Sourav Ghosh (DSTC-22/23-016)**, hereby declare that, this report entitled **"Spam or Ham? Identifying and Filtering Unwanted Emails using Bayesian analysis"** submitted to **Indian Statistical Institute Chennai** requirement of **Postgraduate Diploma in Statistical Methods Analytics in Statistics** in **Department of statistics** , is an original work carried out by us under the supervision of **Dr. Sampangi Raman** . We have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.

Chennai
May 2023

Adarsh Kumar, Sourav Ghosh

CERTIFICATE

This is to certify that the work contained in this project report entitled **"Spam or Ham? Identifying and Filtering Unwanted Emails using Bayesian analysis"** submitted by **Adarsh Kumar** and **Sourav Ghaosh** to Indian Statistical Institute Chennai towards the **Postgraduate Diploma in Statistical Methods and Analytics** has been carried out by them under my supervision and it has not been submitted elsewhere for the award of any degree.

ISI Chennai
May 2023

Dr. Sampangi Raman
Project Supervisor

ACKNOWLEDGEMENT

We, Adarsh Kumar and Sourav Ghosh, would like to express our heartfelt gratitude to everyone who has supported us in completing our project on ”Spam or Ham? Identifying and Filtering Unwanted Emails using Bayesian Analysis.”

First and foremost, we extend our deepest appreciation to Professor Sampangi Raman for his exceptional guidance and mentorship throughout this project. His expertise in Bayesian analysis and his unwavering support have been instrumental in our success. We are grateful for his patience, availability, and dedication to our learning.

We would also like to acknowledge the invaluable contributions of our esteemed professors and mentors who have played a significant role in shaping our academic journey. Their knowledge, encouragement, and insightful feedback have pushed us to excel and have been crucial to the completion of our project.

Furthermore, we are grateful to our friends and classmates, including Joshua, Wahed, Srimanta and others, who have provided us with valuable insights and engaging discussions. Their collaboration and support have greatly enhanced the quality of our work.

Chennai **Adarsh Kumar, Sourav Ghosh**

ABSTRACT

Email Spam has become a major problem nowadays, with the Rapid growth of internet users, Email spam is also increasing. People are using them for illegal and un- ethical conduct, phishing and fraud. Sending malicious links through spam emails can harm our system and can also seek into your system. Creating a fake profile and email account is much easy for the spammers, they pretend like genuine people in their spam emails, these spammers target those peoples who are not aware of these frauds. So, it is needed to Identify those spam mails which are fraud, this project will identify those spam by using techniques of machine learning, this paper will discuss the machine learning algorithms and apply all these algorithms to our data sets and the best algorithm is selected for the email spam detection having best precision and accuracy.

Keywords: *Bayes' theorem, Naive Bayesian analysis*

Contents

Chapter 1

Introduction

1.1 Why Spam filtering ?

Emails are an integral part of our daily lives, and it is hard to imagine a day without checking our inboxes. With the rise in digital communication, email has become the primary mode of communication, and it is crucial that our inbox is free from Spam messages. A Spam email is an unsolicited message that is sent in bulk to a large number of users, and it is mostly intended for commercial or malicious purposes.

Spam emails have become a severe problem in recent years, and it is estimated that over 50 % of emails sent globally are Spam. These emails can be annoying, time-consuming, and sometimes even dangerous, as they may contain phishing links, malware, or other malicious content. It is therefore essential to have a reliable Spam filter that can identify and segregate Spam emails from legitimate ones.

One of the most effective methods for Spam filtering is Bayesian classification, which is a statistical approach to Spam filtering. Bayesian classification uses probabilistic reasoning to calculate the probability of an email being Spam or legitimate based on the occurrence of specific keywords, phrases, or patterns in the email content.

In this article, we will explore how to implement Spam filtering in Python using Bayesian classification. We will start by discussing the basics of Bayesian classification and how it works in the context of email filtering. We will then dive into the implementation details, including data preprocessing, model training, and testing, and finally, we will evaluate the performance of our Spam filter.

1.2 Outline to the Spam Filtering project

Here is a project outline for building a Spam filter using Bayesian techniques:

- Data Collection: Gather a large dataset of emails, including both Spam and non-Spam messages. There are several public datasets available online that we can use for this purpose.
- Data Preprocessing: Clean and preprocess the data by removing stop words, stemming words, and converting all text to lowercase. We may also need to remove any special characters or formatting in the emails.
- Feature Extraction: Extract relevant features from the emails, such as word frequencies, the presence of specific keywords, and other characteristics that may help distinguish between Spam and non-Spam emails.
- Train the Model: Split the dataset into training and testing sets, and use the training set to train a Naive Bayes classifier on the extracted features. The Naive Bayes algorithm is particularly effective for Spam filtering, as it can handle large datasets and is relatively easy to implement.
- Evaluate the Model: Test the trained model on the testing set to evaluate its performance in classifying Spam and non-Spam emails. We will use metrics such as accuracy, precision, recall to evaluate the model's performance.

Chapter 2

Mathematical tools

2.1 Bayes' theorem

Bayes' theorem is a fundamental concept used in Bayesian Spam filtering to calculate the probability that an email is Spam or not_Spam. It states that the probability of a hypothesis (in this case, an email being Spam) given some observed evidence (such as the frequency of certain words in the email) is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis, divided by the probability of the evidence overall. Mathematically, Bayes' theorem can be written as:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where:

- $P(A \mid B)$ is the probability of hypothesis (A) given the evidence (B)
- $P(B \mid A)$ is the probability of the evidence (B) given the hypothesis (A)
- P(A) is the prior probability of hypothesis (A)
- P(B) is the probability of the evidence (B) overall

In Bayesian Spam filtering, the hypothesis is whether an email is a Spam or not, and the evidence is the frequency of certain words or phrases in the email. The prior probability of an email being Spam can be based on the prevalence of Spam in the dataset or on external information about the likelihood of an email being Spam.

Using Bayes' theorem, we can calculate the probability that an email is Spam given the observed evidence, and then use a threshold to classify the email as Spam or not. By updating the model with new training data and adjusting the priors, the Bayesian Spam filter can continually adapt and improve its accuracy over time.

Let's take an example:

Suppose we have a set of training data consisting of 1000 emails, out of which 200 are Spam and 800 are not_Spam. We want to classify a new email as Spam or not_Spam based on its content.

To apply Bayes' algorithm, we first need to calculate the probabilities of certain words appearing in Spam and non-Spam emails. For example, the word "Loan" is more likely to appear in a Spam email than a non-Spam email.

Let's assume that we have calculated the following probabilities based on our training data:

- P(S) = probability of a message being Spam = 0.2 (200/1000)
- P(H) = probability of a message being not_Spam = 0.8 (800/1000)
- $P(Loan \mid S)$ = probability of the word "Loan" appearing in a Spam email = 0.8
- $P(Loan \mid H)$ = probability of the word "Loan" appearing in a non-Spam email = 0.01

Now suppose we have a new email that contains the word "Loan". We can calculate the probability of this email being Spam using Bayes' theorem:

$$P(Spam \mid Loan) = \frac{P(Loan \mid Spam)P(Spam)}{P(Loan)}$$

where P(Loan) is the probability of the word "Loan" appearing in any email, whether Spam or not. We can calculate this probability as:

$$P(Loan) = P(Loan \mid Spam) * P(Spam) + P(Loan \mid H) * P(H)$$

Plugging in the values, we get:

P(Loan) = 0.8 * 0.2 + 0.01 * 0.8 = 0.168

Now we can calculate the probability of the email being Spam:

$$P(Spam \mid Loan) = \frac{P(Loan \mid Spam)P(Spam)}{P(Loan)} = 0.8 * 0.2/0.168 = 0.95$$

This means that the probability of the email being Spam when it contains the word "Loan" is 95 %.

2.2 Naive Bayesian Analysis

Naive Bayes is a variant of Bayes' theorem that is commonly used in Spam filtering. It assumes that the presence of each word in an email is independent of the presence of all other words in the email, which simplifies the calculations and allows for the efficient processing of large datasets.

The basic idea behind Naive Bayes for Spam filtering is to calculate the probability that an email is a Spam or not_Spam based on the frequency of each word in the email. To do this, we first need to build a training dataset consisting of a set of known Spam and non-Spam emails. For each word in the training dataset, we calculate the probability of the word occurring in a Spam email and in a non-Spam email. This is known as the conditional probability of the word given the class (Spam or non-Spam).

When a new email arrives, we calculate the probability of the email being Spam or non-Spam by multiplying the conditional probabilities of each word in the email given the class (Spam or non-Spam). We then compare the probabilities for the two classes and classify the email as Spam or non-Spam based on which class has the higher probability.

Mathematically, the Naive Bayes algorithm for Spam filtering can be written as:

$$P(Spam \mid w_1, w_2, ..., w_n) = \frac{P(Spam) * P(w_1 \mid Spam) * P(w_2 \mid Spam) * ... * P(w_n \mid Spam)}{P(w_1, w_2, ..., w_n)}$$

where:

- $P(Spam \mid w_1, w_2, ..., w_n)$ is the probability that the email is Spam given the words $w_1, w_2, ..., w_n$.

- P(Spam) is the prior probability that an email is Spam.
- $P(w_i \mid Spam)$ is the conditional probability of the ith word given that the email is Spam.
- $P(w_1, w_2, ..., w_n)$ is the overall probability of the words in the email.

The Naive Bayes algorithm assumes that the probability of each word occurring in the email is independent of the probability of all other words occurring in the email. This simplifying assumption is not strictly true, but it works well in practice and allows for the efficient processing of large datasets.

Overall, Naive Bayes is a powerful and effective algorithm for Spam filtering that is widely used in industry and research. It is computationally efficient and can be easily implemented on modern computer hardware. Naive Bayes is a statistical algorithm that uses Naive Bayes' theorem to calculate the probability of a given event, given some evidence or conditions. In Spam filtering, we want to calculate the probability that an email is Spam or not_Spam, given the words in the email.

Mathematical formulation

- P(Spam) is the prior probability of Spam (i.e., the probability that any given email is Spam) P(not_Spam) is the prior probability of not_Spam
- $P(word \mid Spam)$ is the conditional probability of a word given the email is Spam (i.e., the probability that a word appears in a Spam email)
- $P(word \mid \text{Ham})$ is the conditional probability of a word given the email is Ham (i.e., the probability that a word appears in a non-Spam email)
- V $P(email \mid Spam)$ is the probability of the email given that it is Spam $P(email \mid \text{Ham})$ is the probability of the email given that it is Ham

Using Bayes' theorem, we can calculate the probability of an email being Spam or not_Spam given the words in the email:

$$P(Spam \mid email) = \frac{P(email \mid Spam) * P(Spam)}{P(email)}$$

$$P(\text{Ham} \mid email) = \frac{P(email \mid \text{Ham}) * P(\text{Ham})}{P(email)}$$

where P(email) is a normalization constant that ensures the probabilities sum to 1 (one).

To calculate $P(email \mid Spam)$ and $P(email \mid \text{Ham})$, we assume that the probability of each word appearing in the email is independent of the other words (hence the " Naive " in Naive Bayes). Then we can use the following formula:

$$P(email \mid Spam) = P(word1 \mid Spam) * ... * P(wordn \mid Spam) P(email \mid \text{Ham})$$

$$= P(word1 \mid \text{Ham}) * P(word2 \mid \text{Ham}) * ... * P(wordn \mid \text{Ham})$$

where $word_1, word_2, ..., word_n$ are the words in the email.
To calculate $P(word \mid Spam)$ and $P(word \mid \text{Ham})$, we count the number of times each word appears in the Spam and Ham emails in our training set and then divide by the total number of words in each class:

$$P(word \mid Spam)$$

= count(word in Spam) divided by total words in Spam

$$P(word \mid \text{Ham})$$

= count(word in Ham) divided by total words in not Spam

Then we can substitute these probabilities into the above equations to calculate the probability of an email being Spam or Ham given the words in the email.

Once we have calculated these probabilities for a given email, we can compare the probabilities for Spam and Ham, and classify the email as Spam or Ham based on which probability is higher.

Laplace Smoothing in Naive Bayes

Laplace smoothing, also known as additive smoothing or alpha smoothing, is a technique used in the Naive Bayes algorithm to handle zero probabilities. It involves adding a small constant (α) to the numerator and denominator when calculating the conditional probabilities.

The conditional probability calculation without smoothing is given by:

$$P(x|y) = \frac{\text{count}(x, y)}{\text{count}(y)}$$

Where:

- $\text{count}(x, y)$ is the number of occurrences of feature x in class y
- $\text{count}(y)$ is the total count of class y

With Laplace smoothing, the modified conditional probability calculation becomes:

$$P(x|y) = \frac{\text{count}(x, y) + \alpha}{\text{count}(y) + \alpha \cdot V}$$

Where:

- α is the smoothing parameter
- V is the vocabulary size (total number of unique features)

By adding α to the numerator and $\alpha \cdot V$ to the denominator, Laplace smoothing ensures that no probability becomes zero and allows for non-zero probabilities for unseen features.

Choosing the appropriate value of α is important. A larger α value results in stronger smoothing, giving more weight to the uniform distribution. Conversely, a smaller α value gives more weight to the observed frequencies in the training data.

In practice, the value of α is typically determined through cross-validation or other evaluation techniques to find the optimal smoothing parameter for the specific problem at hand.

We will use $\alpha = 1$ in our project work.

Naive Bayesian Example
Lets take example of a spam and ham in tabular form.

Table 2.1: Spam Analysis Examples

Spam/Ham	Sentence
Spam	50% off on all products! Limited time offer!
Spam	Special promotion: Claim your reward.
Ham	The weather is perfect for a picnic.
Ham	Urgent: Update your password immediately.
Ham	Dentist appointment scheduled for Friday at 2 PM.
Ham	Thank you for your purchase. Order shipped.
Ham	Found an interesting article. Check it out!

To perform Naive Bayes classification analysis for spam and ham on the given sentences, we need to calculate the probabilities of each class (spam and ham) for each sentence.

Let's denote:

- $P(\text{Spam})$ as the prior probability of spam class
- $P(\text{Ham})$ as the prior probability of ham class
- $P(\text{Word}|\text{Spam})$ as the conditional probability of a word given spam class
- $P(\text{Word}|\text{Ham})$ as the conditional probability of a word given ham class We also need to calculate the likelihood of each sentence belonging to the spam and ham classes using the Naive Bayes formula:

To perform Naive Bayes spam analysis on the sentence "Earn thousands of dollars from home!" using the given training data, we need to calculate the probabilities of it being classified as spam or ham. Assuming we use an alpha value of 1 for smoothing, the calculation is as follows:

Calculate the prior probabilities:

$$P(\text{Spam}) = \frac{\text{Number of spam instances}}{\text{Total number of instances}}$$

$$P(\text{Ham}) = \frac{\text{Number of ham instances}}{\text{Total number of instances}}$$

Calculate the conditional probabilities: We need to calculate the conditional probabilities of each word appearing in spam or ham sentences. For the word "Earn":

$$P(\text{"Earn"}|\text{Spam}) = \frac{\text{Number of instances with "Earn" in spam} + \alpha}{\text{Total number of spam instances} + \alpha \times \text{Vocabulary size}}$$
$$P(\text{"Earn"}|\text{Ham}) = \frac{\text{Number of instances with "Earn" in ham} + \alpha}{\text{Total number of ham instances} + \alpha \times \text{Vocabulary size}}$$

Assuming "Earn" does not appear in any spam or ham instances, we have:

$$P(\text{"Earn"}|\text{Spam}) = \frac{0+1}{2+1\times 24} = \frac{1}{26}$$
$$P(\text{"Earn"}|\text{Ham}) = \frac{0+1}{5+1\times 24} = \frac{1}{29}$$

$$P(\text{"thousands"}|\text{Spam}) = \frac{0+1}{2+1\times 24} = \frac{1}{26}$$
$$P(\text{"thousands"}|\text{Ham}) = \frac{0+1}{5+1\times 24} = \frac{1}{29}$$

$$P(\text{"of"}|\text{Spam}) = \frac{0+1}{2+1\times 24} = \frac{1}{26}$$
$$P(\text{"of"}|\text{Ham}) = \frac{0+1}{5+1\times 24} = \frac{1}{29}$$

$$P(\text{"dollars"}|\text{Spam}) = \frac{0+1}{2+1\times 24} = \frac{1}{26}$$
$$P(\text{"dollars"}|\text{Ham}) = \frac{0+1}{5+1\times 24} = \frac{1}{29}$$

$$P(\text{"from"}|\text{Spam}) = \frac{0+1}{2+1\times 24} = \frac{1}{26}$$
$$P(\text{"from"}|\text{Ham}) = \frac{0+1}{5+1\times 24} = \frac{1}{29}$$

$$P(\text{"home"}|\text{Spam}) = \frac{0+1}{2+1\times 24} = \frac{1}{26}$$
$$P(\text{"home"}|\text{Ham}) = \frac{0+1}{5+1\times 24} = \frac{1}{29}$$

Calculate the posterior probabilities: Using Bayes' theorem, we can calculate the posterior probabilities of the sentence being classified as spam or ham:

$$P(\text{Spam}|\text{"Earn thousands of dollars from home!"}) =$$
$$P(\text{Spam}) \times P(\text{"Earn"}|\text{Spam}) \times P(\text{"thousands"}|\text{Spam}) \times P(\text{"of"}|\text{Spam}) \times$$
$$P(\text{"dollars"}|\text{Spam}) \times P(\text{"from"}|\text{Spam}) \times P(\text{"home"}|\text{Spam})$$

$$P(\text{Spam}|\text{"Earn thousands of dollars from home"}) =$$
$$P(\text{Spam}) \times P(\text{"Earn"}|\text{Spam}) \times P(\text{"thousands"}|\text{Spam}) \times P(\text{"of"}|\text{Spam}) \times$$

P("dollars"

$$|\text{Spam}) \times P(\text{"from"}|\text{Spam}) \times P(\text{"home"}|\text{Spam})$$

Given the probabilities:

$$P(\text{Spam}) = \frac{2}{7}$$
$$P(\text{"Earn"}|\text{Spam}) = \frac{1}{26}$$
$$P(\text{"thousands"}|\text{Spam}) = \frac{1}{26}$$
$$P(\text{"of"}|\text{Spam}) = \frac{1}{26}$$
$$P(\text{"dollars"}|\text{Spam}) = \frac{1}{26}$$
$$P(\text{"from"}|\text{Spam}) = \frac{1}{26}$$
$$P(\text{"home"}|\text{Spam}) = \frac{1}{26}$$

Let's calculate the posterior probability:

P(Spam—"Earn thousands of dollars from home") = $\left(\frac{2}{7}\right) \times \left(\frac{1}{26}\right) \times \left(\frac{1}{26}\right) \times \left(\frac{1}{26}\right) \times \left(\frac{1}{26}\right) \times \left(\frac{1}{26}\right) \times \left(\frac{1}{26}\right) \approx 9 \times 10^{-10}$

Therefore, the probability of the sentence "Earn thousands of dollars from home" being classified as spam is approximately 9×10^{-10}.

$$P(\text{Ham}|\text{"Earn thousands of dollars from home!"}) =$$
$$P(\text{Ham}) \times P(\text{"Earn"}|\text{Ham}) \times P(\text{"thousands"}|\text{Ham}) \times P(\text{"of"}|\text{Ham}) \times$$
$$P(\text{"dollars"}|\text{Ham}) \times P(\text{"from"}|\text{Ham}) \times P(\text{"home!"}|\text{Ham})$$

$$P(\text{Ham}) = \frac{5}{7}$$

$$\begin{aligned}
P(\text{"Earn"}|\text{Ham}) &= (0+1)/(5+1*24) = \frac{1}{29} \\
P(\text{"thousands"}|\text{Ham}) &= (0+1)/(5+1*24) = \frac{1}{29} \\
P(\text{"of"}|\text{Ham}) &= (0+1)/(5+1*24) = \frac{1}{29} \\
P(\text{"dollars"}|\text{Ham}) &= (0+1)/(5+1*24) = \frac{1}{29} \\
P(\text{"from"}|\text{Ham}) &= (0+1)/(5+1*24) = \frac{1}{29} \\
P(\text{"home"}|\text{Ham}) &= (1+1)/(5+1*24) = \frac{2}{29}
\end{aligned}$$

Now let's calculate the posterior probability:

P(Ham—"Earn thousands of dollars from home") = P(Ham) $\times P(\text{"Earn"}|\text{Ham}) \times$ $P(\text{"thousands"}|\text{Ham}) \quad \times P(\text{"of"}|\text{Ham}) \times P(\text{"dollars"}|\text{Ham}) \times P(\text{"from"}|\text{Ham}) \times$ $P(\text{"home"}|\text{Ham})$

Substituting the values:

$$\begin{aligned}
P(\text{Ham}|\text{"Earn thousands of dollars from home"}) \approx & \left(\frac{5}{7}\right) \times \left(\frac{1}{29}\right) \times \left(\frac{1}{29}\right) \\
& \times \left(\frac{1}{29}\right) \times \left(\frac{1}{29}\right) \times \left(\frac{1}{29}\right) \times \left(\frac{1}{29}\right)
\end{aligned}$$

Simplifying the calculation:

$$P(\text{Ham}|\text{"Earn thousands of dollars from home"}) \approx 12 \times 10^{-10}$$

Therefore, the probability of the sentence "Earn thousands of dollars from home" being classified as ham with Laplace smoothing (alpha = 1) is approximately 12×10^{-10}.

The probabilities for the other words can be calculated based on the training data.

Chapter 3

Python code for Spam filtering

We are Reading data from Kaggle so loading data on Google colab

```
! pip install -q kaggle
from google.colab import files
files.upLoand()
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets downLoand -d venky73/Spam-mails-dataset
```

Unziping the zipped file

```
# Unzipping code
import zipfile
zip_ref = zipfile.ZipFile('/content/Spam-mails-dataset.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

Loading all essential packages

```
import random
import re
import math
from collections import defaultdict
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

Reading dataset using pandas

```
df=pd.read_csv("/content/spam_ham_dataset.csv", header=None,
   names=['Label', 'Text','tag'])
df = df.drop('tag', axis=1)
df = df.drop(0)
df
```

```
        Label                                                Text
NaN     label                                                text
605.0     ham  Subject: enron methanol ; meter # : 988291\r\n...
2349.0    ham  Subject: hpl nom for january 9 , 2001\r\n( see...
3624.0    ham  Subject: neon retreat\r\nho ho ho , we ' re ar...
4685.0   spam  Subject: photoshop , windows , office . cheap ...
...       ...                                                 ...
1518.0    ham  Subject: put the 10 on the ft\r\nthe transport...
404.0     ham  Subject: 3 / 4 / 2000 and following noms\r\nhp...
2933.0    ham  Subject: calpine daily gas nomination\r\n>\r\n...
1409.0    ham  Subject: industrial worksheets for august 2000...
4807.0   spam  Subject: important online banking alert\r\ndea...

[5171 rows x 2 columns]
```

Reindexing the dataset

```
df = df.iloc[1:]
df = df.reset_index(drop=True)
df
```

```
     Label                                               Text
0      ham  Subject: enron methanol ; meter # : 988291\r\n...
1      ham  Subject: hpl nom for january 9 , 2001\r\n( see...
2      ham  Subject: neon retreat\r\nho ho ho , we ’ re ar...
3     spam  Subject: photoshop , windows , office . cheap ...
4      ham  Subject: re : indian springs\r\nthis deal is t...
...    ...                                                ...
5165   ham  Subject: put the 10 on the ft\r\nthe transport...
5166   ham  Subject: 3 / 4 / 2000 and following noms\r\nhp...
5167   ham  Subject: calpine daily gas nomination\r\n>\r\n...
5168   ham  Subject: industrial worksheets for august 2000...
5169  spam  Subject: important online banking alert\r\ndea...

[5170 rows x 2 columns]
```

Randomly choosing an email from dataset

```
df.iloc[random.randint(0,len(df)+1),1]
```

```
’ Subject: enron / hpl noms for november 16 , 2000 ( see
 attached file: hplnl 116 . xls ) - hplnl 116 . xls ’
```

Code for cleaning dataset

```
def remove_digits(sentence):
    pattern = r'\d+'
    return re.sub(pattern, '', sentence)

def remove_chars(string):
    remove = ['\r', '\n', ':', ',','.',';']
    return string.translate(str.maketrans('', '', ''.join(remove)))

def remove_special_chars(sentence):
    pattern = r'[^a-zA-Z0-9\s]'
    return re.sub(pattern, '', sentence)

def remove_long_spaces(sentence):
    words = sentence.split()                      # split the
        sentence into words
    clean_words = [word for word in words if word] # remove empty
        words
    return " ".join(clean_words)

def convert_labels(labels):
    new_labels = []
    for label in labels:
        if label == 'spam':
            new_labels.append(2)
        else:
            new_labels.append(0)
    return new_labels

def remove_single_letters(sentence):
    words = sentence.split()                # split the sentence
        into words
    clean_words = [word for word in words if len(word) > 1] #
        remove single-letter words
    return " ".join(clean_words)           # join the remaining
        words into a sentence
```

Applying the data cleaning function in Dataset

```
df["Label"] = df["Label"].apply(lambda x: 1 if "spam" in x.lower()
    else 0)
df["Text"]=df["Text"].apply(remove_digits)
df["Text"]=df["Text"].apply(remove_long_spaces)
df["Text"]=df["Text"].apply(remove_chars)
df["Text"]=df["Text"].apply(remove_special_chars)
df['Text']=df['Text'].str.lower()
df['Text']=df['Text'].apply(remove_single_letters)
df.sample(5)
```

Random massage after data cleaning

```
df.iloc[random.randint(0,len(df)+1),1]
```

Subject daren the above meter began flow on volume of decatherms versus the start date of the th as indicated on sitara deal ticket can you please notify me as to whether the deal ticket will reflect this change thanks jackie

Counting spam and ham in dataset

```
df.groupby('Label').describe()
```

```
         Text
        count unique                                           top freq
Label
0        3671   3175  subject hpl nom for march see attached file hp...   31
1        1499   1460                                        subject   18
```

Probablity of being spam of a massage

```
df['Label'].value_counts(normalize=True)
```

```
0 0.710058
1 0.289942
Name: Label, dtype: float64
```

Making a Naive Bayes Classification code

```
class NaiveBayesClassifier:
    def __init__(self):
        self.word_counts = defaultdict(lambda: [0, 0]) #
            [spam_count, ham_count]
        self.total_spam = 0
        self.total_ham = 0
        self.spam_prior = 0.0
        self.ham_prior = 0.0

    def train(self, training_data):
        for _, row in training_data.iterrows():
            text, label = row['Text'], row['Label']
            words = self.preprocess_text(text)
            for word in words:
                self.word_counts[word][label] += 1
                if label == 1:
                    self.total_spam += 1
                else:
                    self.total_ham += 1

        total_messages = self.total_spam + self.total_ham
        self.spam_prior = self.total_spam / total_messages
        self.ham_prior = self.total_ham / total_messages
```

Reading data using pandas

```
    def classify(self, text):
        words = self.preprocess_text(text)
        spam_prob = math.log(self.spam_prior)
        ham_prob = math.log(self.ham_prior)

        for word in words:
            word_data = self.word_counts[word]
            spam_prob += math.log((word_data[1] + 1) /
                (self.total_spam + len(self.word_counts)))
            ham_prob += math.log((word_data[0] + 1) /
                (self.total_ham + len(self.word_counts)))

        return spam_prob > ham_prob


    def preprocess_text(text):
        # Preprocess the text by converting to lowercase and
           extracting words
        words = re.findall(r'\w+', text.lower())
        return words
```

Naming the Model and running on train set

```
# Create a NaiveBayesClassifier instance
classifier = NaiveBayesClassifier()
# Train the classifier
classifier.train(X_train)
```

Applying the model

```
# Test the classifier
email = "Shop now and save 50% off your purchase."
is_spam = classifier.classify(email)

if is_spam:
    print(f"The email '{email}' is classified as spam.")
else:
    print(f"The email '{email}' is classified as ham.")
```

```
' The email 'Shop now and save 50% off your purchase.' is classified as spam.'
```

Train dataset

```
X_train
```

```
   Label Text
1487 0 subject procedure for adding new capacity tick...
1780 1 subject doctor aipproved cia lls lev itra will...
422 0 subject re cornhusker tenaska iv has been oper...
1779 0 subject hpl noms for nov see attached file hpl...
1957 0 subject re meter dec deal one year deal out of...
... ... ...
4426 0 subject re deal daren thu has asked if can ext...
466 0 subject tenaska iv bob understand from sandi t...
3092 1 subject fwd here all meds ana vlagr fiori term...
3772 0 subject fw candle lighting original message fr...
860 0 subject re and yes it is gtc spot daren farmer...
```

Test dataset

```
X_test
```

```
    Label Text
1566 0 subject enron actuals for april estimated actu...
1988 0 subject natural gas nomination for enron metha...
1235 1 subject learn to save on medications at discou...
2868 0 subject re correction to nominations for eastr...
3435 1 subject imageplus ink toner and ribbon cartrid...
1471 0 subject meter roos common point trade zone am ...
1129 1 subject
3747 1 subject hi paliourg get all pills everything f...
3047 0 subject enron hpl noms for february february t...
530 0 subject tenaska iv transport took look at the ...
4980 1 subject would help your fa mily hello sent you...
3148 0 subject fw customer list this one includes mor...
1662 0 subject skydive spaceland specials skydive spa...
3819 1 subject awesome movies of the super sexy jaime...
4700 0 subject fw waha hubco fyi original message fro...
3975 0 subject revisions march wellhead estimate dare...
5111 0 subject revision hpl nom for august see attach...
1835 1 subject if you want to get me into bed have to...
4679 0 subject bammel children christmas program pict...
842 1 subject stock market standouts infotex holding...
2462 0 subject re hpl system training daren would lik...
683 0 subject hpl nom for august see attached file h...
2664 1 subject
5144 0 subject re epgt gloria the difference between ...
3123 0 subject enron actuals for july thru july teco ...
3378 0 subject deal this is the deal we made for el p...
```

Naive bayes model code

```
 # Writting a new codes for Naive Bayes Classifier as above
    mathematics in chapter 2

class NaiveBayesClassifier:
    def __init__(self):
        self.word_counts = defaultdict(lambda: [0, 0]) #
            [spam_count, ham_count]
        self.total_spam = 0
        self.total_ham = 0
        self.spam_prior = 0.0
        self.ham_prior = 0.0

    def train(self, training_data):
        for _, row in training_data.iterrows():
            text, label = row['Text'], row['Label']
            words = self.preprocess_text(text)
            for word in words:
                self.word_counts[word][label] += 1
                if label == 1:
                    self.total_spam += 1
                else:
                    self.total_ham += 1

        total_messages = self.total_spam + self.total_ham
        self.spam_prior = self.total_spam / total_messages
        self.ham_prior = self.total_ham / total_messages
```

Continued

```
    def classify(self, text):
        words = self.preprocess_text(text)
        spam_prob = math.log(self.spam_prior)
        ham_prob = math.log(self.ham_prior)

        for word in words:
            word_data = self.word_counts[word]
            spam_prob += math.log((word_data[1] + 1) /
                (self.total_spam + len(self.word_counts)))
            ham_prob += math.log((word_data[0] + 1) /
                (self.total_ham + len(self.word_counts)))

        return spam_prob > ham_prob

    @staticmethod
    def preprocess_text(text):
        # Preprocess the text by converting to lowercase and
            extracting words
        words = re.findall(r'\w+', text.lower())
        return words

# Create a NaiveBayesClassifier instance
classifier = NaiveBayesClassifier()
```

Code for finding model accuracy

```python
def calculate_accuracy(model, test_data):
    total_examples = len(test_data)
    correct_predictions = 0

    for _, row in test_data.iterrows():
        text, true_label = row['Text'], row['Label']
        predicted_label = model.classify(text)

        if predicted_label == true_label:
            correct_predictions += 1

    accuracy = correct_predictions / total_examples
    return accuracy
```

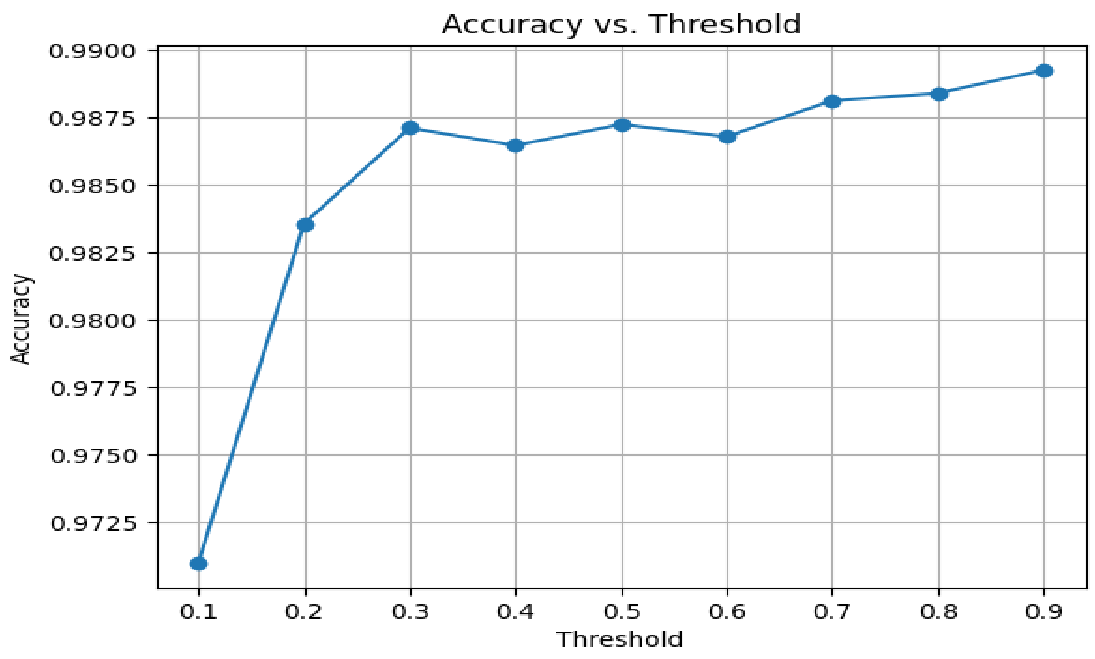
Checking model accuracy using several spliting dataset

```
#Code for checking accuracy at several threshold points.

thresholds = [0.1, 0.2, 0.3, 0.4, 0.5,0.6,0.7,0.8,0.9]
accuracies = []

for threshold in thresholds:
    test_size = threshold # Calculate test size based on the
        threshold
    X_train, X_test = train_test_split(df, test_size=test_size,
        random_state=42)
    classifier.train(X_train)
    accuracy = calculate_accuracy(classifier, X_test)
    accuracies.append(accuracy)

# Plot the accuracy graph
plt.plot(thresholds, accuracies, marker='o')
plt.xlabel('Threshold')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. Threshold')
plt.grid(True)
plt.show()
```



Here, we have chosen a threshold value of 0.4, which implies a dataset split of 60 percent for the training set and 40 percent for the test set. This splitting configuration aligns with the principle of allocating a significant portion of the data for training purposes while still reserving a substantial portion for testing and evaluating the model's performance. By adhering to this approach, we can ensure that the data sets benefit from a substantial amount of training data, which is crucial for effective model training and generalization.

Finding the accuracy of the model

```
#Selecting optiomal threshold point
# Splitting the data into training and testing sets
X_train, X_test = train_test_split(df , test_size=0.4,
    random_state=42)

# Assuming you have trained the model and have a test dataset
    named 'test_data'
accuracy = calculate_accuracy(classifier, X_test)
print("Accuracy:", accuracy*100,"%")
```

```
Accuracy: 98.64603481624758 %
```

Applying classifier model with new sentence

```
# Applying classifier model
email = "bring me back"
is_spam = classifier.classify(email)

if is_spam:
    print(f"The email '{email}' is classified as spam.")
else:
    print(f"The email '{email}' is classified as ham.")
```

```
The email 'bring me back' is classified as ham.
```

Chapter 4

Conclusion

In this study, we applied the Naive Bayes classification algorithm to the "Spam Mails Dataset" obtained from Kaggle. The dataset consists of labeled email messages, where the task is to classify emails as either spam or ham. After training and evaluating the Naive Bayes model, we achieved an accuracy of 0.9865.

The Naive Bayes algorithm is a popular choice for spam classification tasks due to its simplicity and effectiveness in handling text data. It leverages the probabilistic framework of Bayes' theorem and assumes independence between features to calculate the likelihood of an email being spam or ham.

Our obtained accuracy of 0.9865 indicates that our Naive Bayes spam classifier correctly predicted the labels for approximately 98.65% of the emails in the dataset. This demonstrates a strong performance in distinguishing between spam and non-spam emails.

The high accuracy suggests that our Naive Bayes classifier can be relied upon for spam classification tasks. It successfully captured important patterns and features indicative of spam emails, leading to accurate predictions. The model's performance indicates its ability to effectively differentiate between the characteristics of spam and ham emails.

In conclusion, our study demonstrates the effectiveness of the Naive Bayes algorithm for spam classification. With an accuracy of 0.9865, our model successfully identified a significant majority of the emails as spam or ham. These findings contribute to the growing body of research in spam detection and provide practical insights for developing robust spam filters in email systems.

Bibliography

[1] Metsis, V., Androutsopoulos, I., Paliouras, G. (2006). Spam filtering with naive Bayes—Which naive Bayes? CEAS, 158, 27-28.

[2] Rennie, J. D., Shih, L., Teevan, J., Karger, D. R. (2003). Tackling the poor assumptions of naive Bayes text classifiers. In Proceedings of the Twentieth International Conference on Machine Learning (ICML-03) (pp. 616-623).

[3] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G. (2000). An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 160-167).

[4] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. (1998). A Bayesian approach to filtering junk e-mail. In AAAI Workshop on Learning for Text Categorization.

[5] Yang, Y., Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. ICML, 97, 412-420.