Classification of spam and non-spam emails using Naïve Bayes Algorithm

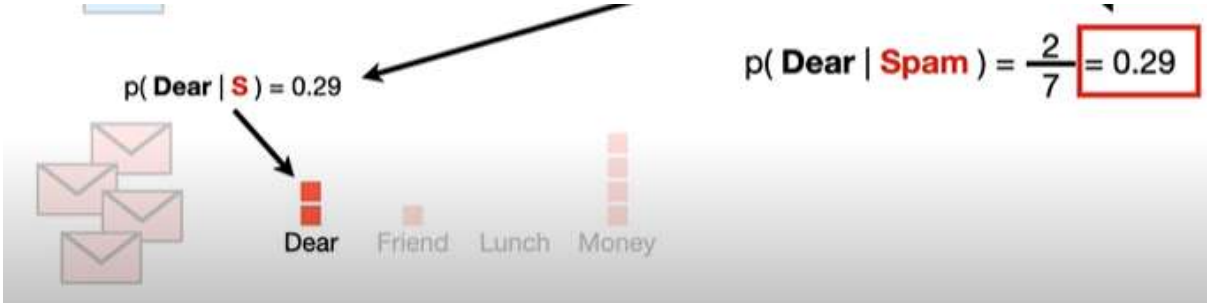
The first thing we will do considering we have data of emails that are classified as spam or non-spam, we will execute some preprocessing algorithms to convert the information into much simpler terms which will help the machine learning algorithm to accurately and efficiently calculate the probabilities.

Preprocessing:

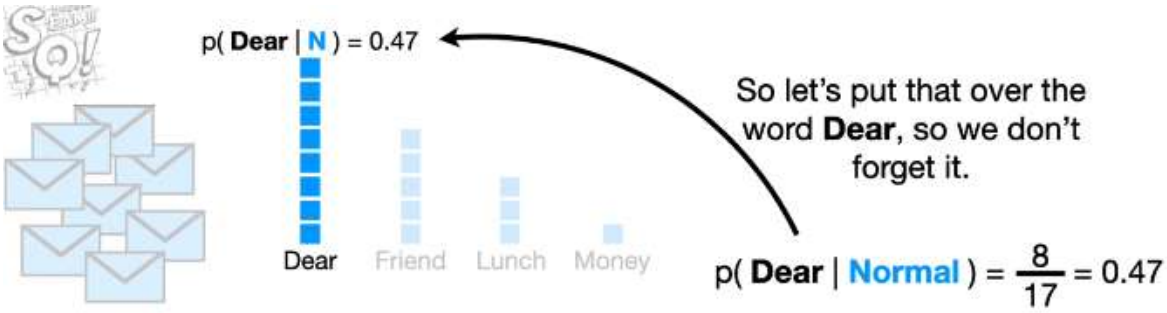
We prepare the data by cleaning it. This involves removing unnecessary information like special characters, commonly used words like “like”,” a”,” are”, numbers, and HTML tags. We convert all text to lowercase to ensure consistency.

1. **Tokenization:** Break down the text of each email into individual words or tokens. For example, the sentence "Sale! Buy t-shirts in 299" would be tokenized into ["sale!", "buy", "t-shirts", “in”].
2. **Create a Vocabulary:** Build a list of all unique words (tokens) in your dataset. This list is your vocabulary. For example, your vocabulary might include words like "buy," "sale!" "t-shirts,".

3. **Count Word Frequencies:** For each email in your dataset, count how many times each word from your vocabulary appears in that email. This creates a "word count" or "term frequency" representation of each email. For example, we have the word “dear” in the data which has a word count of 2500 in non-spam emails and a word count of 300 in spam emails
4. **Calculate Class Probabilities:** Compute two sets of probabilities:
    - **P(word|spam):** Calculate the probability that a specific word appears in a spam email. This is done by counting how many times each word appears in all the spam emails in your dataset. For example, as the word “dear” has a word count of 300 in the spam email data, the probability of dear being a part of spam email would be considerably low.



- **P(word|non-spam):** Calculate the probability that a specific word appears in a non-spam email. This is done by counting how many times each word appears in all the non-spam emails in your dataset. For example, as the word “dear” has a word count of 2500 in the non-spam email data, the probability of dear being a part of spam email would be considerably high.
-



-

5. **Calculate Prior Probabilities:** Determine the overall probability of an email being spam or non-spam, irrespective of its content. You calculate these probabilities based on the proportion of spam and non-spam emails in your dataset. These are denoted as:
    - **P(spam):** The probability that an email is spam.
    - **P(non-spam):** The probability that an email is non-spam.

We will first identify the probability of the email being spam or non-spam. Suppose we have 10 emails in our data and 7 of them are non-spam emails and 3 of them are spam emails then p(spam) will be 0.3 and p(non-spam) will be 0.7.

6. **Calculate Posterior Probabilities:** For a new, unseen email, calculate the probability that it belongs to each class (spam or non-spam) using Bayes' theorem. This involves multiplying the prior probability (P(spam) or P(non-spam)) by the product of the probabilities of the individual words in the email.

   Suppose we have email#1 containing the words “dear ” and “friend” and we have prior data of probability of the email being spam is 0.3 and the probability of the email being non-spam is 0.7

   probability of dear and friend being spam is 0.2, 0.1 resp. and the probability of dear and friend being non-spam is 0.5, 0.3

   So for calculating the probability of email#1 being non spam,

   We will use naïve bayes algorithm which states

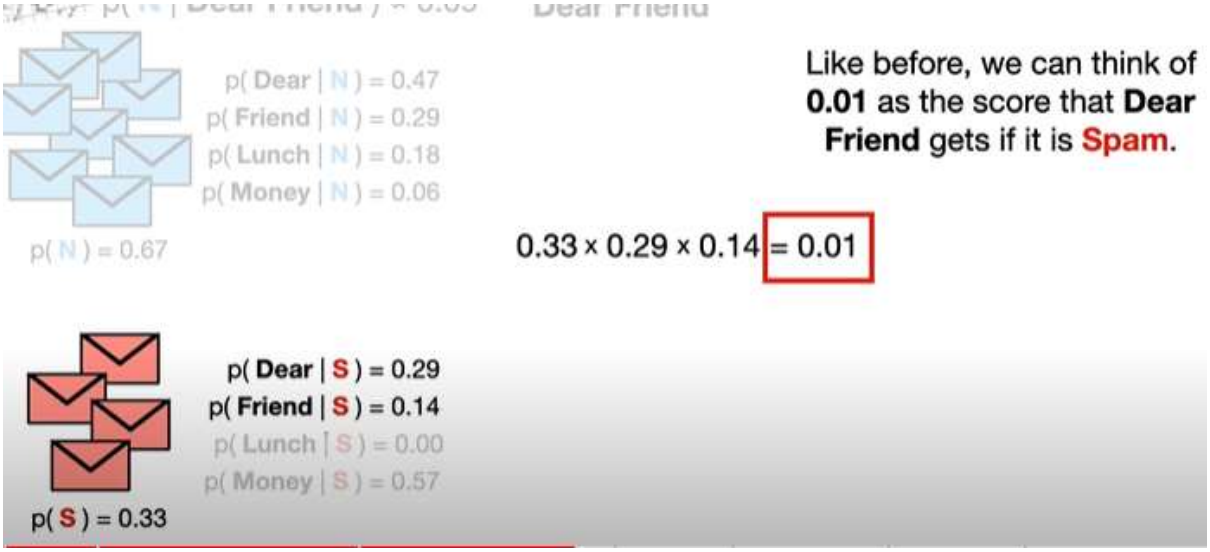
   P(email#1) being non spam = p(non-spam) x p(dear|non spam) x p(friend|non spam)

   Which would be **0.7x0.5x0.3 = 0.105**

And for calculating the probability of email#1 being spam,

P(email#1) being spam = p(spam) x p(dear|spam) x p(friend|spam)

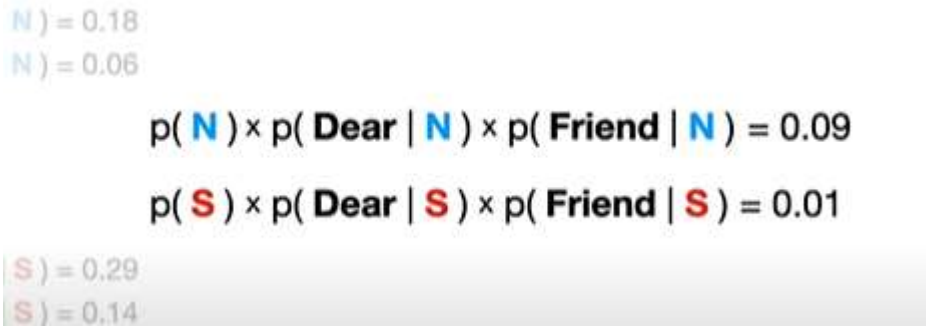
Which would be **0.3x0.2x0.1 = 0.006**



7. **Make a Prediction:** Compare the calculated probabilities for spam and non-spam. The class with the higher probability is the prediction for the email. If P(spam) > P(non-spam), you predict it's spam; otherwise, you predict it's non-spam.

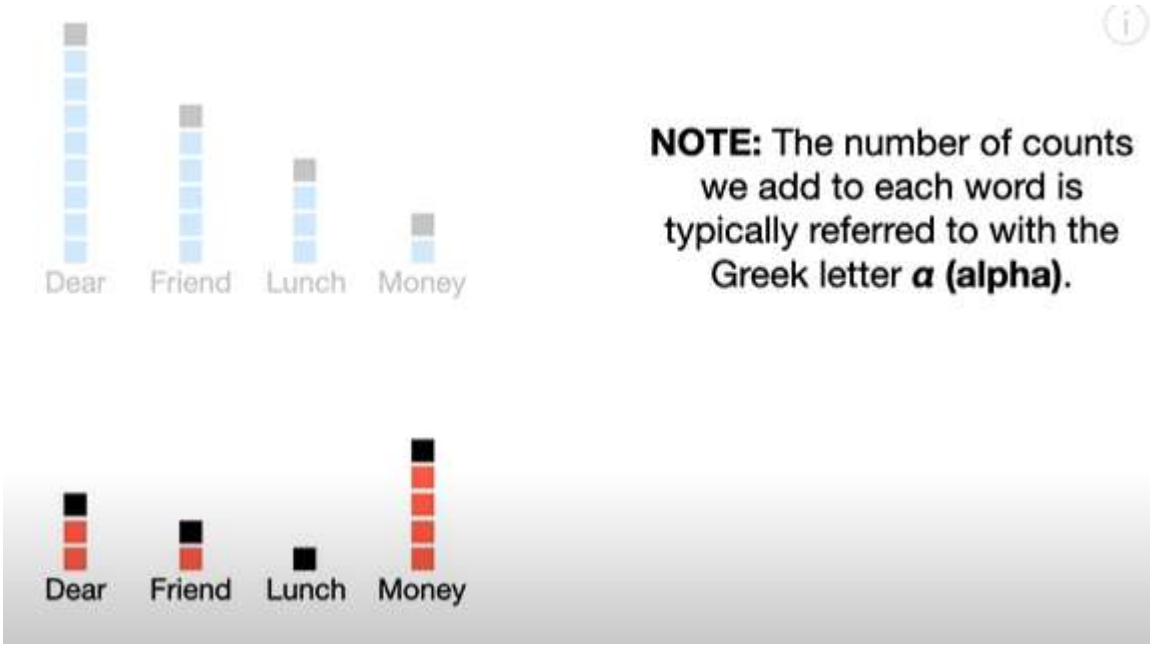
   So in the case of email#1 the p(non-spam)>p(spam)

   Hence, we can predict that email#1 is indeed a non-spam email.



8. **Evaluation:** To assess the model's performance, use a separate set of emails that it hasn't seen before (testing data). Compare the model's predictions to the actual labels (spam or non-spam) to calculate metrics like accuracy, precision, recall, and F1-score.

   We also need to set an alpha value for the data set for example if the word “luck” is not present in the dataset of non-spam emails, it will make the overall probability zero hence we need at least one extra block under every word so that probability doesn’t be zero.



9. **Fine-Tuning:** Depending on the model's performance, you may need to fine-tune it by adjusting parameters or trying different variations of the Naive Bayes algorithm to improve its accuracy.

That's the detailed process of building a spam email prediction model using the Naive Bayes algorithm. This approach leverages probability theory to classify emails based on the words they contain, making it a common and effective method for spam detection.