# Spam Detection

**Team Members**: Tejas Harish Borkar (2K18/CO/373) ,Tushar Ahuja (2K18/CO/374)

**Aim:** To develop spam detection filter using **Naive Bayes**.

## Objective

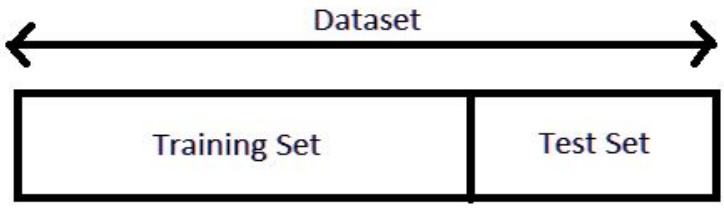
- Our goal is to code a spam filter from scratch that classifies messages as spam and non-spam with an accuracy greater than 90%.
- We will be using multinomial Naive Bayes and conditional probability to achieve it.

## Idea

- Spam is referred to as any irrelevant or unsolicited messages sent over the Internet, typically to a large number of users, for the purposes of advertising, phishing, spreading malware, etc.
- Therefore, it is important to classify messages as spam and non-spam for the user's convenience.
- A dataset of 5,572 SMS messages will be used to develop the spam filter.

The steps involved are:

## Training and Test Set




- The dataset is split into a training set and a test set.
- 80% of the data is used for training and the remaining 20% for testing.
- The entire dataset is randomized before splitting to ensure that spam and ham messages are spread properly throughout the dataset.
- The percentage of spam and ham messages in the training and testing sets are analyzed..
- The percentages are expected to be close to what we have in the full dataset.

## Data Preprocessing

The Preprocessing of the training data and testing is separately performed to simplify the further process.The phases being taken in the preprocessing includes :

- **Case Folding and removal of characters:** All text is converted to lowercase aiming to homogenize the data.This is followed by deleting the characters other than letters, numbers and punctuation.
- **Tokenization:**The process of tokenization is carried out to break the string into tokens or individual words.This helps us to build the vocabulary in the next step.

## Creating the Vocabulary

- The vocabulary is created, which means a list with all the unique words in our training set.
- We will now use the vocabulary we created to make a new DataFrame.
- Therefore, data access is simple and this DataFrame is used to make the required data transformation

| | Label | SMS |
|---|---|---|
| **0** | spam | SECRET PRIZE! CLAIM SECRET PRIZE NOW!! |
| **1** | ham | Coming to my secret party? |
| **2** | spam | Winner! Claim secret prize now! |

| | Label | secret | prize | claim | now | coming | to | my | party | winner |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | spam | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **1** | ham | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| **2** | spam | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

.

## Calculating Constants and parameters(Naive Bayes Algorithm)

When a new message comes in, our **multinomial Naive Bayes algorithm** will make the classification based on the results it gets to these two equations below, where "$w_1$" is the first word, and $w_1$,$w_2$, ..., $w_n$ is the entire message:

$$\mathbf{P(Spam|w_1, w_2, ..., w_n) \propto P(Spam) \cdot \prod_{i=1}^{n} P(w_i|Spam)}$$

$$\mathbf{P(Ham|w_1, w_2, ..., w_n) \propto P(Ham) \cdot \prod_{i=1}^{n} P(w_i|Ham)}$$

If P(Spam | $w_1$,$w_2$, ..., $w_n$) is greater than P(Ham | $w_1$,$w_2$, ..., $w_n$), then the message is spam.

To calculate P($w_i$|Spam) and P($w_i$|Ham), we need to use separate equations:

$$\mathbf{P(w_i|Spam) = \frac{N_{w_i|Spam} + \alpha}{N_{Spam} + \alpha \cdot N_{Vocabulary}}}$$

$$\mathbf{P(w_i|Ham) = \frac{N_{w_i|Ham} + \alpha}{N_{Ham} + \alpha \cdot N_{Vocabulary}}}$$

$\mathbf{N_{w_i|Spam}}$ = the number of times the word $\mathbf{w_i}$ occurs in spam messages

$\mathbf{N_{w_i|Ham}}$ = the number of times the word $\mathbf{w_i}$ occurs in ham messages

$\mathbf{N_{Spam}}$ = total number of words in spam messages

$\mathbf{N_{Ham}}$ = total number of words in ham messages

$\mathbf{N_{Vocabulary}}$ = total number of words in the vocabulary

$\boldsymbol{\alpha} = 1$ ($\boldsymbol{\alpha}$ is a smoothing parameter)

Now the constants are calculated

- P(Spam) and P(Ham)
- $N_{Spam}$, $N_{Ham}$, $N_{Vocabulary}$

We'll also use Laplace smoothing and set $\boldsymbol{\alpha} = 1$.

## Classifying A New Message

Now that we have all our parameters calculated, we can start creating the spam filter. The spam filter is understood as a function that:

- ❖ Takes in as input a new message ($w_1$, $w_2$, ..., $w_n$).
- ❖ Calculates P(Spam|$w_1$, $w_2$, ..., $w_n$) and P(Ham|$w_1$, $w_2$, ..., $w_n$).
- ❖ Compares the values of P(Spam|$w_1$, $w_2$, ..., $w_n$) and P(Ham|$w_1$, $w_2$, ..., $w_n$), and:
- ➢ If P(Ham|$w_1$, $w_2$, ..., $w_n$) > P(Spam|$w_1$, $w_2$, ..., $w_n$), then the message is classified as ham.
- ➢ If P(Ham|$w_1$, $w_2$, ..., $w_n$) < P(Spam|$w_1$, $w_2$, ..., $w_n$), then the message is classified as spam.
- ➢ If P(Ham|$w_1$, $w_2$, ..., $w_n$) = P(Spam|$w_1$, $w_2$, ..., $w_n$), then the algorithm may request human help.

Some new messages will contain words that are not part of the vocabulary. We will simply ignore these words when we're calculating the probabilities.

## Measuring the Spam Filter's Accuracy

We can compare the predicted values with the actual values to measure how good our spam filter is with classifying new messages. To make the measurement, we'll use **accuracy** as a metric:

$$\text{Accuracy} = \frac{\text{number of correctly classified messages}}{\text{total number of classified messages}}$$

## Technologies and language used

- We will be using Python language and Jupyter Notebook for the implementation.
- We will also use the python libraries:
  - Pandas
  - Numpy
  - Matplotlib (for visualisation)
  - Scikit

## Closest Paper

Arifin, D. D., & Bijaksana, M. A. (2016, September). Enhancing spam detection on mobile phone Short Message Service (SMS) performance using FP-growth and Naive Bayes Classifier. In 2016 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob) (pp. 80-84). IEEE.