# Naive Bayes Classifier

## Theory

### Bayes Theorem

The Bayes' Theorem formula is expressed as:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Where:

- $P(A)$ and $P(B)$ are the probabilities of the events A and B.
- $P(A|B)$ is the probability of A occuring, given that B has occurred.
- $P(B|A)$ is the probability of B occuring, given that A has occurred.

### Naive Bayes Formula

Naive Bayes' takes advantage of the Bayes' Theorem to predict the probability that a given feature set belongs to particular label.

In the example demonstrated in the corresponding C++ code the feature set are words and the label is Spam or Not Spam.

The Naive Bayes' formula can be viewed as an extension of the Bayes Theorem. The formula for a text being spam given a set of words $w_1, w_2, ..., w_n$ can be expressed as:

$$P(Spam|w_1, w_2, ..., w_n) = \frac{P(w_1, w_2, ..., w_n|Spam) \times P(Spam)}{P(w_1, w_2, ..., w_n)}$$

### The "Naive" Assumption

The "naive" in Naive Bayes comes from the assumption that all the features (words in this case) are independent of each other given the class label. With this assumption, we can take advantage of the fact that:

$$P(w_1, w_2, ..., w_n|Spam) = P(w_1|Spam) \times P(w_2|Spam) \times ... \times P(w_n|Spam)$$

## Final steps

Finally we can use this to create our revised Naive Bayes formula:

$$P(Spam|w_1, w_2, ..., w_n) = \frac{P(Spam) \times P(w_1|Spam) \times P(w_2|Spam) \times ... \times P(w_n|Spam)}{P(w_1, w_2, ..., w_n)}$$

# C++ Implementation

## Class Definitions

### Public Methods

- train
- predict

### Private Members

- word_counts: A map for keeping track of the occurrence of each word.
- label_word_counts: A nested map for keeping the count of words given a specific label
- label_counts: A map to store the occurrence of each label.

## Training Method

The train() function takes two vectors as arguments, data and labels.

1. It iterates over all the data points and extracts words and labels.
2. The word occurrences are stored in word_counts.
3. The occurrences of words within each class are stored in label_word_counts.
4. The occurrence of each label is stored in label_counts.

## Prediction Method

The predict() function takes a string (composed of multiple words) as an argument and returns an integer label (either 0 or 1 in this example).

1. best_label and best_prob are initialized to keep track of the label with the highest probability.
2. For each label, it calculates the initial probability $P(label)$
3. For each word in the given string, it calculates (P(word|label)) and adds it to the probability.
4. If the calculated probability is greater than best_prob, it updates best_label and best_prob.
5. It returns best_label as the final prediction.

## Main Function

1. A NaiveBayes object is created.
2. Training data and labels are supplied to the train() function.
3. Several test data points are then classified using the predict() function.

## Key Notes

- Logarithm is used to prevent underflow and to speed up calculations.
- Add-one (Laplace) smoothing is used when calculating $P(label)$