**End-to-End Project on SMS Spam Classification**

***Question:*** Build an End-to-End application to check whether the given SMS is SPAM or Not using **NAÏVE BASED CLASSIFICATION** - SMS Collection data set / refer another data set of your wish.

Solution: The data set has been obtained from

UCI Machine Learning Repository: The UCI SMS Spam Collection Dataset is available at <https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>.

Nature of data is as follows:

1. Data is in a .csv file with 2 attributes, [label], [message]
2. "label": The classification label for each message is shown in this column. It tells if a communication is non spam or legitimate (referred to as "ham" or "spam").
3. "message": The textual content of each communication is contained in this column.

Added by me for further analysis, by using various functions:

1. "message\_length": This column displays the character count for each message's length.
2. “count”: This column displays the number of non-numeric non alphabetical characters in the message.

Description on few functions and logics used in this process:

1. CountVectorizer(): from the scikit-learn class, transforms text messages into numerical feature vectors. It lowercases the text, tokenizes it, and builds a vocabulary of words. The vectorizer object's fit\_transform() method is used to convert the text messages into a sparse matrix representation.
2. The Multinomial Naive Bayes algorithm is implemented by the Scikit-Learn class MultinomialNB(). It computes the probability of each class based on the feature vectors and assumes that the features (word counts) are multinomially distributed.
3. The Multinomial Naive Bayes algorithm, which works well for text classification applications with discrete count-based characteristics, is implemented as the class MultinomialNB(). Due to its ease of use, effectiveness, and efficiency, particularly when working with text data, it is a popular option for spam detection and other text classification issues.
4. Fit(): To train the model, this method is used on the classifier object. It accepts as inputs the feature vectors (X\_train) and the labels that go with them (y\_train).
5. To produce predictions using fresh data, the classifier object's predict() method is used. The projected labels are returned when the feature vectors of the new data have been supplied.
6. Violin Plot: which show data distribution and give a visual depiction of the shape, spread, and density of values, are used to visualise distribution. When comparing distributions across many categories or groupings, they are especially helpful.

Box plot and kernel density plot combined: Violin graphs incorporate the advantages of both types of charts. They display the median, quartiles, and outliers (much like a box plot), and by varying the width of the violin plot, they can approximate the distribution of the underlying data.

Violin plots can display many categories or groups side by side, making it simple to visually compare their distributions.

Compact description: Violin plots are well suited for displaying large datasets or several variables because they offer a clear and straightforward description of the data.

1. The main idea: CONFUSION MATRIX:

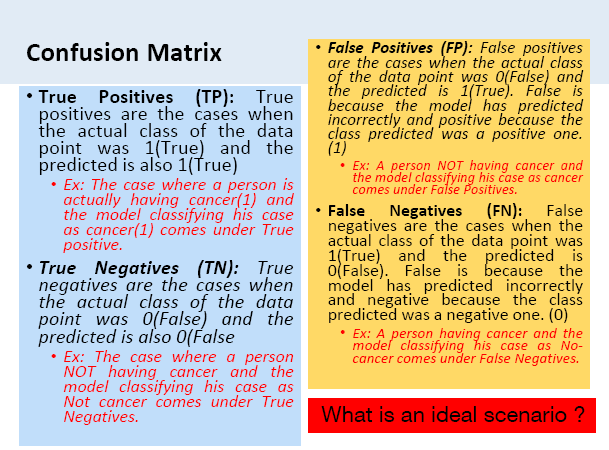
The confusion matrix consists of four key metrics:

True Positives (TP): The number of spam messages that were correctly classified as spam.

True Negatives (TN): The number of non-spam messages that were correctly classified as non-spam.

False Positives (FP): The number of non-spam messages that were incorrectly classified as spam (Type I error).

False Negatives (FN): The number of spam messages that were incorrectly classified as non-spam (Type II error).



Team Details:

S Raj Abhinay

21BAI1778