# AI -Spam Classifier

**NAIVE BAYES**

1.Begin by gathering a labeled dataset. The dataset should include examples of both spam and non-spam messages. Each example should be represented as a feature vector

2.Next, we will need to calculate the probability distributions for each label (spam and non-spam) given the data in your training set.

a. Start by calculating the number of messages for each label.

b. Next, calculate the probabilities of each feature occurring for each label. We can do this by counting the number of times each feature appears for each label and then dividing by the total number of messages for that label.

3.After we have calculated the probability distributions, we can use the Naive Bayes model to make predictions about new, unseen data.

a. To predict whether a new message is spam or not, calculate the probabilities of each feature occurring for the spam label and for the non-spam label.

b. Next, use the Bayes theorem to calculate the probability of the spam label given the features in the new message, and the probability of the non-spam label given the features in the new message.

c. Finally, classify the new message as spam if the probability of the spam label is greater than the probability of the non-spam label, and as non-spam otherwise

**TRAINING OF A MODEL**

Naive Bayes is a simple but powerful algorithm for classification tasks. For spam classification, we can follow these general steps to train a Naive Bayes model:

Data Preprocessing: Clean and preprocess your data, which involves tasks such as tokenization, removing stopwords, and stemming.

Feature Extraction: Create a bag-of-words or TF-IDF representation of your text data. This step involves converting text data into numerical feature vectors that can be used as input for the model.

Model Training: Use the preprocessed data to train the Naive Bayes classifier, considering the conditional probabilities of each word or feature given the class (spam or non-spam) using the Bayes theorem.

Model Evaluation: Evaluate the trained model using a test set to assess its performance, considering metrics such as accuracy, precision, recall, and F1-score.

**Python Code:**

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score, classification\_report

# Assume you have 'X' as the input text data and 'y' as the corresponding labels.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature extraction

vectorizer = CountVectorizer()

X\_train\_counts = vectorizer.fit\_transform(X\_train)

X\_test\_counts = vectorizer.transform(X\_test)

# Model training

clf = MultinomialNB()

clf.fit(X\_train\_counts, y\_train)

# Model evaluation

y\_pred = clf.predict(X\_test\_counts)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**PERMANANCE EVALUATION:**

To properly evaluate the model, we would need to provide the training data (X\_train and y\_train) and the test data (X\_test and y\_test) that were used for the evaluation. Additionally, we should supply the actual performance metrics generated by the code, such as accuracy, precision, recall, and the F1-score.

Without the specific data and metrics, we can't directly evaluate the model's performance. However, we can check the printed results from the print statements in the code to obtain the accuracy score and the classification report, which includes precision, recall, and F1-score for each class.

Inspecting these metrics will give you a clear understanding of how well the model performs on the given dataset. If the accuracy is high and the precision, recall, and F1-scores are balanced, it indicates that the model is performing well. If the metrics are not satisfactory, you might need to explore other techniques, such as hyperparameter tuning, different feature extraction methods, or more advanced preprocessing techniques, to improve the model's performance."