**Introduction**

**Spam Mail Detection Classifier using Naïve Bayes**

Classify emails as Spam or Not Spam (Ham) using a machine learning model.

**Objective**

The objective of this project is to build a machine learning model that classifies emails as **Spam** or **Not Spam (Ham)**. This helps in filtering unwanted emails and improving user productivity and security.

**How It Works**

1. **Problem Definition:** The problem is to classify emails into spam or ham categories using machine learning**.**

2. **Dataset Collection:**

* Dataset loaded from platforms like Kaggle.
* Contains two main columns: email text and spam/ham labels.

3. **Data Preprocessing:**

* Convert text to lowercase.
* Remove punctuation, numbers, and stopwords.
* Tokenize and apply lemmatization or stemming to standardize words.

4. **Feature Engineering:**

* Convert text data into numerical format using CountVectorizer or TF-IDF Vectorizer.

5. **Train-Test Split:**

* Data split into training (80%) and testing (20%) sets for evaluation.

6. **Model Building:**

* Used Multinomial Naive Bayes (MultinomialNB) model — ideal for text classification tasks.

7. **Training:**

* The model is trained on the vectorized training data.

8. **Model Evaluation:**

* Metrics used: Accuracy, Precision, Recall, F1-score, and Confusion Matrix to measure performance.

9. **Frontend & Backend Integration (Next Step):**

* Build a frontend to take user input (email text).
* Connect with the backend Python model to predict spam/ham results in real-time

**Tools & Technologies Used**

**Programming Language**

* **Python** – The core implementation language.

**Libraries**

* **scikit-learn (sklearn)** — For machine learning algorithms and evaluation metrics.
* **pandas** — For data manipulation and analysis.
* **seaborn** — For statistical data visualization.
* **matplotlib** — For plotting graphs and visualizing data distributions.
* **re (Regular Expressions)** — For text cleaning and preprocessing..

**Development Tools**

* **Jupyter Notebook** – Used as the primary code editor for development and debugging.

**Use Cases**

1.Email service providers (Gmail, Yahoo, Outlook) for filtering spam messages.

2. Security applications to detect phishing or fraudulent emails.

3. Automating spam detection in CRM or business tools.

4. SMS and notification filtering systems.

**Conclusion**

By following this step-by-step process, we successfully trained a Spam Detection Model using Python and machine learning. The model shows good accuracy and can effectively classify emails into spam or ham categories. The next phase involves creating a user-friendly frontend and connecting it with the backend model for real-time prediction and deployment.