**Worksheet 02: Classification Methods & Performance Metrics**

1. **Naïve Bayes Classifier**

Naïve Bayes is a simple yet powerful **probabilistic algorithm** used for **classification** tasks. It is based on **Bayes' Theorem**, which describes the probability of an event based on prior knowledge.

It is called **"naïve"** because it **assumes** that all features in the data are **independent**, which is not always true in real life but works well in practice.

**Best used for:**

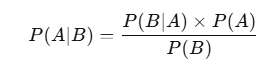
* **Text classification** (spam detection, sentiment analysis, topic categorization)
* **Medical diagnosis**
* **Face recognition**

**Types of Naïve Bayes:**

1. **Gaussian Naïve Bayes** – Used for continuous data.
2. **Multinomial Naïve Bayes** – Used for text classification.
3. **Bernoulli Naïve Bayes** – Used for binary features

## ****Understanding Bayes’ Theorem****

Bayes’ Theorem is the foundation of Naïve Bayes. It helps us update our beliefs based on new evidence.



### ****Breaking it Down:****

* **P(A∣B)→ Posterior Probability** (Probability of A happening, given B)
* **P(B∣A) → Likelihood** (Probability of B happening, given A)
* **P(A) → Prior Probability** (How much we already believe in A)
* **P(B)→ Evidence** (Probability of B occurring)

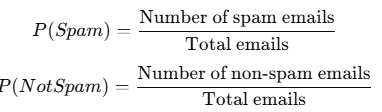
## ****How Does Naïve Bayes Work?****

### ****Example 1: Spam Email Detection 📧****

Imagine you receive an email, and you want to classify it as **"Spam"** or **"Not Spam"** based on words in the email.

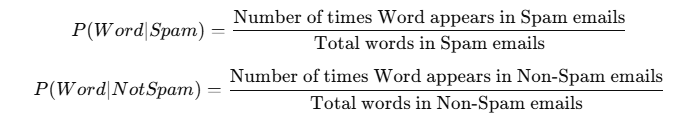
#### **Step 1: Calculate Prior Probabilities**

We check how many emails in our dataset are **Spam** and how many are **Not Spam**.



#### **Step 2: Calculate Likelihood for Each Word**

For each word in the email, we calculate how often it appears in spam vs. non-spam emails.



#### **Step 3: Apply Bayes' Theorem**

We multiply these probabilities to find out if the email is more likely to be spam or not.



If **P(Spam∣Words)** is higher than **P(NotSpam∣Words)**, the email is classified as spam.





## ****2. Linear Discriminant Analysis (LDA)****

Linear Discriminant Analysis (LDA) is a machine learning algorithm used for **classification**. It finds a **straight line (or a plane in higher dimensions)** that best separates different categories in the data.

Think of LDA as drawing the **best boundary** between different groups so that when a new data point appears, we can **easily classify it**.

### ****Concept:****

* A classification technique that **reduces dimensionality** while preserving class separability.
* Maximizes the ratio of **between-class variance to within-class variance**.

### ****Why Do We Need LDA?****

* If you have **multiple classes** (e.g., Spam vs. Not Spam, Disease vs. No Disease), LDA helps you **find patterns** in your data.
* LDA **reduces the dimensions** of the dataset while keeping the most important information.
* It is **faster than other classifiers** like k-NN or deep learning for small datasets.

### ****How Does LDA Work?****

LDA works in **three main steps**:

### ****Step 1: Compute the Mean of Each Class****

For every category in the dataset, we calculate the **average value** (centroid).

**Example:**  
If we are classifying students into "Pass" and "Fail" based on study hours and test scores:

* Compute the **average study hours and scores** for both Pass and Fail groups.

### ****Step 2: Compute Scatter (Variance) Within and Between Classes****

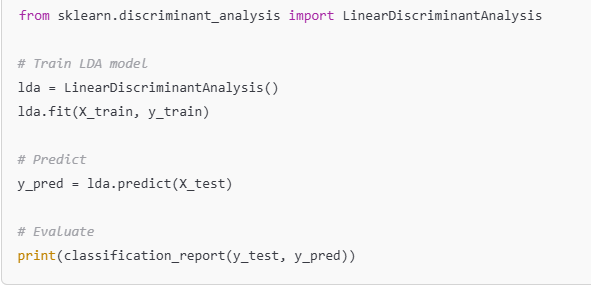
LDA calculates:

1. **Within-class scatter** → How much the data spreads **inside** each category.
   * If students who "Pass" have similar study hours but "Fail" students have a big variation, this affects the boundary.
2. **Between-class scatter** → How different the classes are from each other.
   * If "Pass" students have high scores and "Fail" students have low scores, the difference is clear.

### ****Step 3: Find the Best Line to Separate Classes****

LDA finds the **optimal line** (or plane) that: **Minimizes the overlap between categories** **Maximizes the distance between class centers**

The new data points are projected onto this line, making classification easier.



## ****3. Logistic Regression****

Logistic Regression is a **supervised learning algorithm** used for **classification** tasks. It predicts whether something belongs to **one of two categories** (binary classification).

**Example:**

* **Spam Detection** → Is an email spam (1) or not spam (0)?
* **Disease Prediction** → Does a patient have a disease (1) or not (0)?
* **Loan Approval** → Will a person get a loan (1) or not (0)?

### ****Concept:****

* Used for **binary & multiclass classification**.
* Uses **sigmoid function** to predict probabilities.

### ****How Does Logistic Regression Work?****

Instead of predicting values directly, Logistic Regression calculates **probabilities** using the **Sigmoid Function**.

#### **Step 1: Calculate the Weighted Sum**

Just like linear regression, we compute:

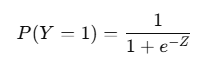


where:

* x1, x2​ are input features (e.g., study hours, past grades).
* w1, w2​ are weights (learned by the model).
* B is the bias term.

#### **Step 2: Apply the Sigmoid Function**

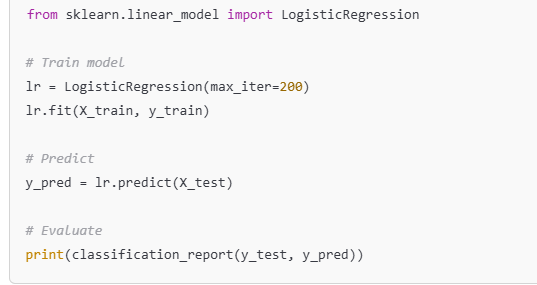
The sigmoid function converts Z into a probability between **0 and 1**:



* If P(Y=1)>0.5 → Predict **1 (Yes/Positive)**
* If P(Y=1)<0.5 → Predict **0 (No/Negative)**

**Think of it like a switch:**

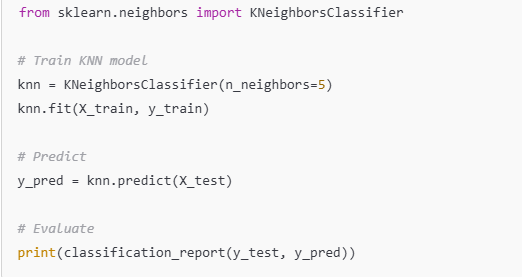
* If probability is **high (>50%)**, we classify it as **1**.
* If probability is **low (<50%)**, we classify it as **0**.



## ****4. K-Nearest Neighbors (KNN) Classifier****

### ****Concept:****

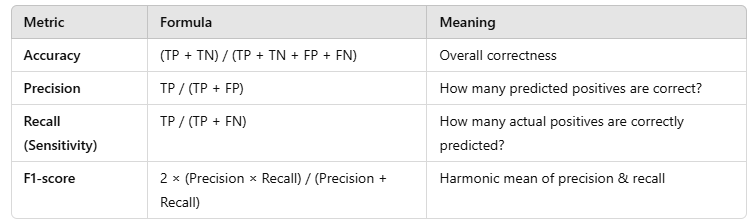
* A **non-parametric** classifier that classifies based on the majority class of **K nearest neighbors**.
* Sensitive to the choice of **K**.

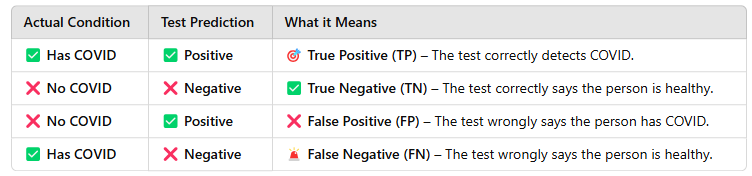


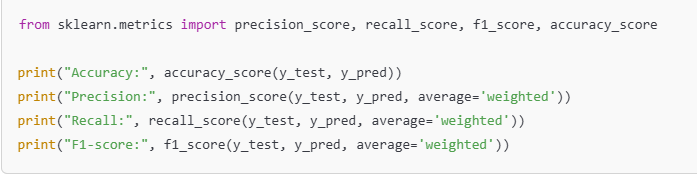
## ****5. Performance Metrics for Classification****

## When evaluating a classification model, we use different metrics to measure how well it's performing. The four most common metrics are ****Precision, Recall, Accuracy, and F1-Score****.

**Precision, Recall, Accuracy, F1-score**

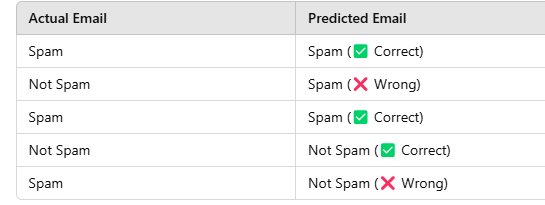
****

****

****

Example Scenario: Spam Email Classification

Imagine we have an AI model that predicts whether an email is **Spam (1)** or **Not Spam (0)**.

****

Now, we define key terms:

* **True Positives (TP)**: Correctly classified spam emails.
* **False Positives (FP)**: Normal emails wrongly classified as spam.
* **False Negatives (FN)**: Spam emails wrongly classified as normal.
* **True Negatives (TN)**: Correctly classified normal emails.

## ****Precision (How many predicted spam emails are actually spam?)****

**Formula:**

Precision=TP/(TP+FP)​

* **High precision** means fewer false positives (fewer normal emails marked as spam).
* **Low precision** means the model makes a lot of mistakes by marking normal emails as spam.

**Use Case:** When **False Positives** are costly, like detecting fraud in banking.

## ****Recall (How many actual spam emails were detected?)****

**Formula:**

Recall={TP}/{TP + FN}​

* **High recall** means the model finds most spam emails.
* **Low recall** means many spam emails go undetected.

**Use Case:** When **False Negatives** are costly, like detecting cancer in medical tests.

## ****Accuracy (Overall correctness of the model)****

**Formula:**

Accuracy= {TP + TN}/{TP + TN + FP + FN}

* **High accuracy** means the model is making very few mistakes.
* **Low accuracy** means the model is predicting a lot of emails incorrectly.

**Problem with Accuracy:** If spam emails are only **5% of all emails**, and the model always predicts "Not Spam," it will have **95% accuracy but be completely useless for detecting spam!**

**Use Case:** Best when the dataset is balanced (equal spam and normal emails).

## ****F1-Score (Balances Precision and Recall)****

**Formula:**

F1= {2×Precision×Recal}/{Precision+Recall}​

* **High F1-Score** means both **Precision and Recall** are high.
* **Low F1-Score** means one of them is very low.

**Use Case:** When we need a balance between **Precision and Recall**, like spam detection.

**Note:**

✔ **If you want fewer false alarms (FP), focus on Precision.**  
✔ **If you don’t want to miss important cases (FN), focus on Recall.**  
✔ **If data is balanced, Accuracy is good.**  
✔ **If you need both Precision and Recall, use F1-Score.**

### ****Confusion Matrix****

* A table showing **true vs. predicted labels**.
* **TP (True Positives), TN (True Negatives), FP (False Positives), FN (False Negatives)**.

