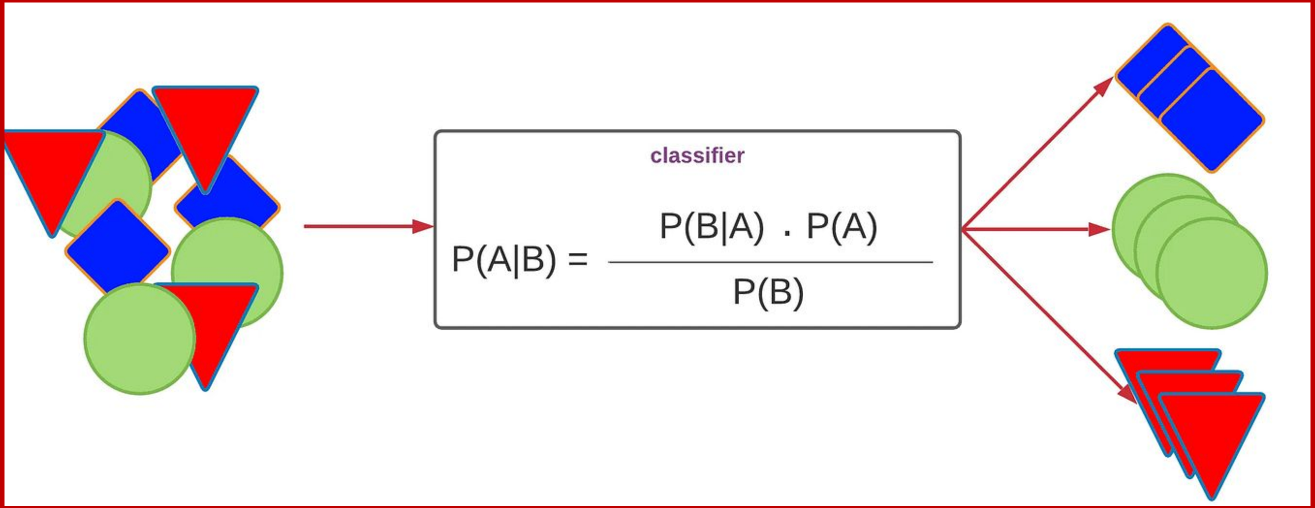
Session 4

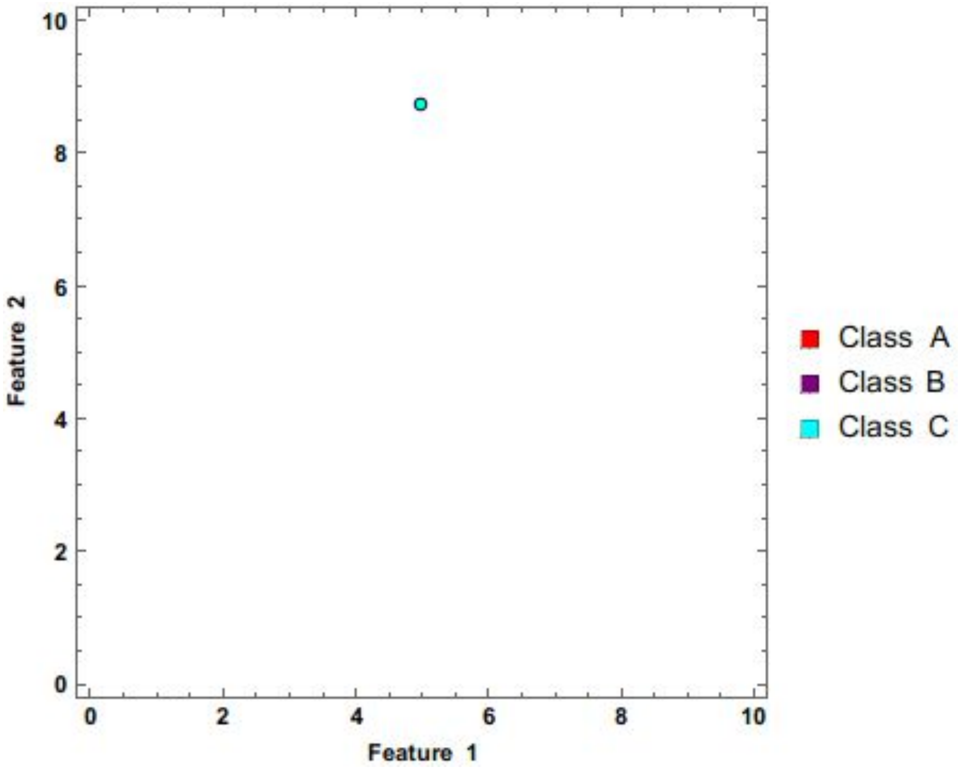
Naïve Bayes' Classifier



Naïve Bayes Classifier

- Naïve Bayes Classifier can be trained easily and fast and can be used as benchmark model.
- When variable selection is carried out properly, Naïve Bayes can perform as well as or even better than other statistical models such as logistic regression.
- Naive Bayes requires a strong assumption of independent predictors, so when the model has a bad performance, the reason leading to that may be the dependence between predictors.
- Doesn’t require feature scaling

Naïve Bayes' Classifier



Naïve Bayes' Classifier:

- **Naive Bayes is the simplest algorithm that you can apply to your data.**
- **This algorithm assumes all the variables in the dataset "Naive" i.e., not correlated to each other.**
- It requires categorical values.
- It uses Bayes' theorem as its base.
- Mostly used to get the base accuracy of the dataset.
- Mostly used with:
  - Real time Prediction.
  - Multi class Prediction.
  - Text classification/ Spam Filtering/ Sentiment Analysis, and Recommendation Systems.

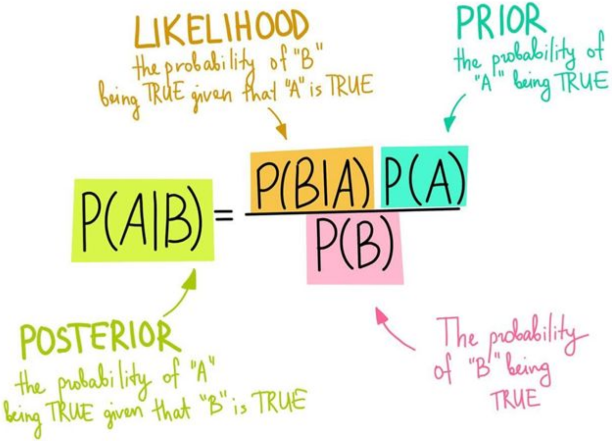
Types

- Multinomial Naïve Bayes → for text (word counts).
- Bernoulli Naïve Bayes → for binary data (0/1 features).
- Gaussian Naïve Bayes → for numerical (continuous) data.

Example:

Priors: احتمال كل كلاس حسب عدد الأمثلة.

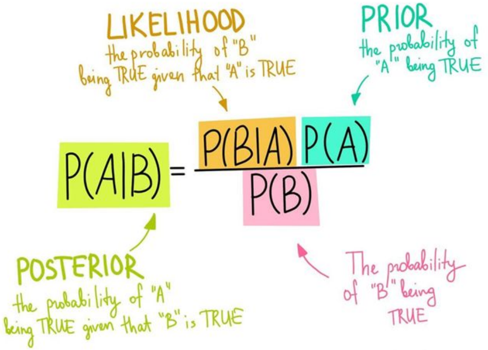
Likelihoods: لكل Feature، عد كم مرة ظهر مع كل كلاس.



- **Problem:** Classify whether an email is "spam" or "not spam"
- **Features:** Presence (1) or absence (0) of words "buy", "cheap", "offer".
- **Training Data:**

Email	buy	cheap	offer	Class
1	1	0	1	spam
2	1	1	0	not spam
3	0	1	1	spam
4	1	0	0	not spam
5	0	1	1	spam

Calculate Priors



Prior Probability of Spam ($P(\text{spam})$):

$$P(\text{spam}) = \frac{\text{Number of spam emails}}{\text{Total number of emails}} = \frac{3}{5} = 0.6$$

Prior Probability of Not Spam ($P(\text{not spam})$):

$$P(\text{not spam}) = \frac{\text{Number of not spam emails}}{\text{Total number of emails}} = \frac{2}{5} = 0.4$$

Calculate Likelihoods

- **Likelihoods for "spam" class:(1,3,5)**
  - buy=1 → 1/3 , buy=0 → 2/3
  - cheap=1 → 2/3 , cheap=0 →1/3
  - offer=1 → 3/3=1 , offer=0 → 0

- **Likelihoods for "not spam" class: (2,4)**
  - buy=1 → 2/2=1 , buy=0 → 0
  - cheap=1 → 1/2 , cheap=0 → 1/2
  - offer=1 → 0 , offer=0 → 2/2=1

New Email to Classify

- New Email: $"buy"=1,"cheap"=1,"offer"=0$
- Objective: Determine whether the new email is "spam" or "not spam".

Calculate Posterior for Spam

1

Posterior Probability of Spam

$$(P(spam \mid buy = 1, cheap = 1, offer = 0)$$

$$\propto P(buy = 1 \mid spam) \cdot P(cheap = 1 \mid spam) \cdot P(offer = 0 \mid spam) \cdot P(spam)$$

$= (\frac{1}{3}) * (\frac{2}{3}) * (0) * 0.6$ **$= 0$**

New Email to Classify

- New Email: $"buy"=1,"cheap"=1,"offer"=0$
- Objective: Determine whether the new email is "spam" or "not spam".

Calculate Posterior for Not Spam

2

Posterior Probability for Not Spam

$$P(notspam \mid buy = 1, cheap = 1, offer = 0)$$

$$\propto P(buy = 1 \mid notspam) \cdot P(cheap = 1 \mid notspam) \cdot P(offer = 0 \mid notspam) \cdot P(notspam)$$

= (1) * (½) * (1) * 0.4 **= 0.2**

New Email: $"buy"=1,"cheap"=1,"offer"=0$

Step 5: Final Decision

- Spam = 0
- Not Spam = 0.2

👉 **Email classify as : Not Spam** ✅

Why Normalize Probabilities?

1. **To Compare Probabilities Directly:**
    a. When calculating conditional probabilities $P(Y|X)$ using Bayes' Theorem, we need to normalize the probabilities so they can be compared directly.
    b. Without normalization, the probabilities are not on the same scale and cannot be directly compared.

> Example: Spam = 0.2, Not Spam = 0.05
> → Without normalization, you could say "Spam is bigger."
> → **But to say** *Spam = 80% chance* and *Not Spam = 20% chance*, **you need normalization.**

2. **To Ensure the Probabilities Sum to 1:**
    a. After normalization, the sum of the probabilities for all classes (e.g., spam and not spam) should equal 1.
    b. This ensures we have a valid probability distribution that can be used for classification.

- Without smoothing → rare/missing features kill probabilities.
    - If a **feature is very rare or never appears** in the training data for a class, then its likelihood becomes **0**.
- With Laplace smoothing → every feature has at least a small chance, making the model more robust.

Laplace Smoothing (Add-1 Smoothing)

- To fix this, we use **Laplace smoothing** (also called **add-1 smoothing**).
- Instead of using raw counts, we add **+1 to every count**:

$$P(feature \mid class) = \frac{\text{count(feature, class)} + 1}{\text{count(class)} + k}$$

where:

- k = number of possible feature values (for binary features, $k = 2$).

Example

Suppose in "Spam" emails we never saw the word "**offer=0**".

- Without smoothing:

$$P(offer = 0 \mid Spam) = 0$$

- With Laplace smoothing:

$$P(offer = 0 \mid Spam) = \frac{0 + 1}{3 + 2} = 0.2$$

Now it's a small probability instead of zero.

Supervised machine learning

Algorithm	Needs Feature Scaling?	Reason
KNN (K-Nearest Neighbors)	✅ Yes	Relies on distance between points
SVM (Support Vector Machine)	✅ Yes	Depends on dot product and distances
Linear Regression	✅ (Recommended)	Regularization + Gradient Descent convergence
Logistic Regression	✅ (Recommended)	Regularization + coefficient stability
Decision Tree	❌ No	Works with threshold splits, not distances
Random Forest	❌ No	Based on Decision Trees
Naïve Bayes	⚠️ Sometimes	- **Gaussian NB** assumes normally distributed data → scaling helps - **Multinomial / Bernoulli NB**: handle counts/binary values → scaling not needed