we give text to a machine learning model, it cannot directly “understand” words like *cat*, *happy*, or *India*. Instead, we need to **convert words into numbers (vectors)** that capture their meaning or at least their presence

**1. Bag of Words (BoW)**

* BoW is the simplest way to turn text into numbers.
* We first make a list of all the unique words in our dataset (this is called the vocabulary).
* Then, each sentence or document is represented as a vector showing how many times each word appears.
* It does not care about grammar or word order, only word counts.

**Example:**  
Vocabulary = [“cat”, “dog”, “happy”]

* Sentence: “cat is happy” → [1, 0, 1]
* Sentence: “dog is happy” → [0, 1, 1]

Very simple, but the problem is: it ignores meaning and makes very large vectors if the vocabulary is big.

**2. TF-IDF (Term Frequency – Inverse Document Frequency)**

* TF-IDF improves BoW by also checking how important a word is.
* **TF (Term Frequency):** How often the word appears in the document.
* **IDF (Inverse Document Frequency):** Gives less importance to common words (like “the”, “is”) and more importance to rare but meaningful words.
* Final score = TF × IDF.

**Example:**  
If we have many documents:

* Word “India” appears in only 2 documents → High importance.
* Word “the” appears in almost all documents → Very low importance.

TF-IDF is better than BoW for tasks like search engines or spam detection, but it still ignores word order and meaning.