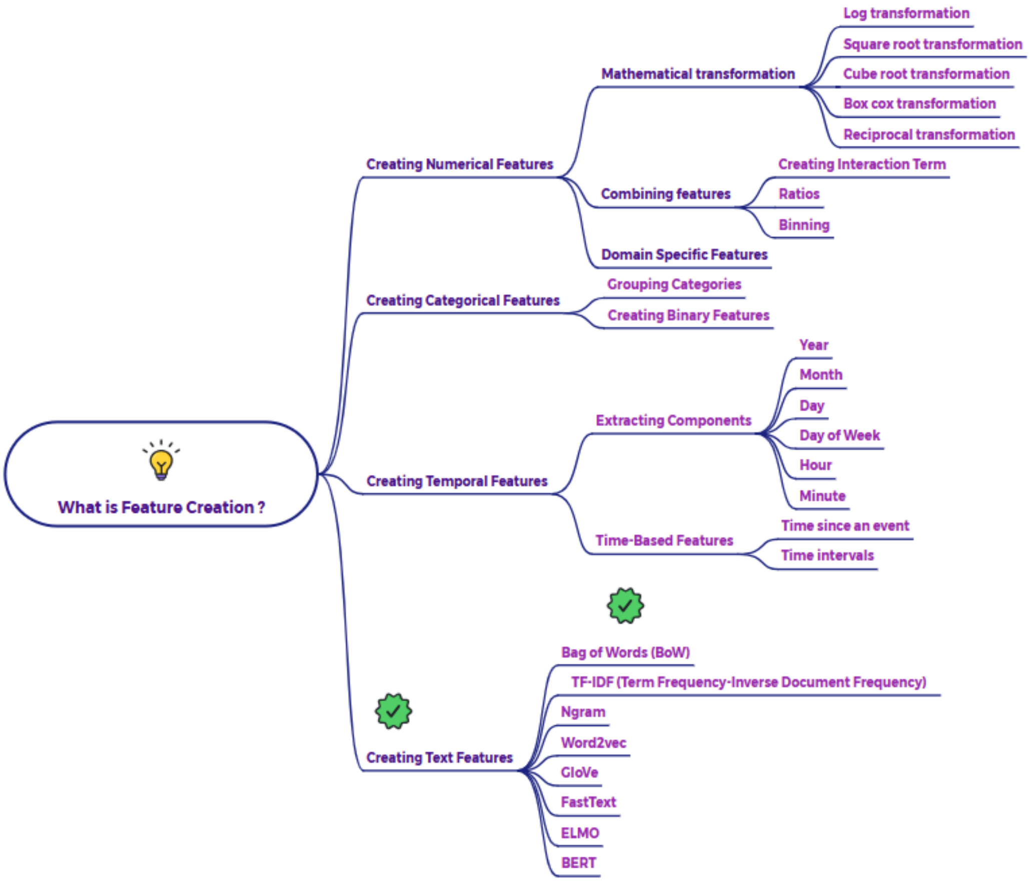
Explain BOW with an example



Bag of Words (BoW)

Bag of Words is a fundamental technique in natural language processing (NLP) for representing text data in a numerical format that machine learning models can understand. It's considered a "frequency-based" method because its core idea is to count the occurrences of words within a document. Essentially, BoW transforms a piece of text, whether it's a sentence, a paragraph, or an entire article, into a vector of word frequencies, disregarding the grammatical structure and word order.

Here's a more detailed breakdown of how it works:

1. **Corpus and Vocabulary Creation**:
   - We start with a collection of text documents, which we call a "corpus."

- The first step is to create a vocabulary. This involves examining all the documents in the corpus and identifying all the unique words.
- Typically, we perform some preprocessing steps such as:
    - **Tokenization**: Splitting the text into individual words or tokens.
    - **Lowercasing**: Converting all words to lowercase to treat "The" and "the" as the same word.
    - **Removing punctuation**: Eliminating characters like commas, periods, and question marks.
    - **Removing stop words**: Discarding common words like "the," "is," and "and" that don't carry much meaning.
    - **Stemming or lemmatization**: Reducing words to their root form (e.g., "running" to "run").

2. **Document Representation:**
    - Once we have the vocabulary, we can represent each document as a numerical vector.
    - The vector has a length equal to the number of unique words in the vocabulary.
    - Each position (index) in the vector corresponds to a specific word in the vocabulary.
    - For each document, we count how many times each word from the vocabulary appears in that document.
    - The resulting vector contains these counts.

Detailed Example:

Let's consider a small corpus with three short documents:

- Document 1: "The quick brown fox jumps over the lazy dog."
- Document 2: "The dog is happy."
- Document 3: "A quick brown fox."

1. **Preprocessing**: After preprocessing (tokenization, lowercasing, removing punctuation and stop words), our documents might look like this:
    - Document 1: "quick brown fox jumps lazy dog"
    - Document 2: "dog happy"
    - Document 3: "quick brown fox"
2. **Vocabulary**: Our vocabulary becomes:
    - {"quick", "brown", "fox", "jumps", "lazy", "dog", "happy"}
3. **Document Vectors**: Now, we represent each document as a vector:
    - Document 1: "quick brown fox jumps lazy dog"
        - Vector: [1, 1, 1, 1, 1, 1, 0]
        - Explanation: "quick" appears once, "brown" appears once, "fox" appears once, "jumps" appears once, "lazy" appears once, "dog" appears once, and "happy" appears zero times.
    - Document 2: "dog happy"
        - Vector: [0, 0, 0, 0, 0, 1, 1]
        - Explanation: "quick" appears zero times, "brown" appears zero times, "fox" appears zero times, "jumps" appears zero times, "lazy" appears zero times, "dog" appears once, and "happy" appears once.
    - Document 3: "quick brown fox"
        - Vector: [1, 1, 1, 0, 0, 0, 0]
        - Explanation: "quick" appears once, "brown" appears once, "fox" appears once, "jumps" appears zero times, "lazy" appears zero times, "dog" appears zero times, and "happy" appears zero times.

In essence, BoW converts text into a format suitable for machine learning by focusing on word frequencies, albeit at the cost of losing word order and context.