**How does the Naive Bayes Classifier work? What is Posterior Probability?**

The Naive Bayes Classifier is a machine learning method that is based on probability theory's Bayes theorem. It is a probabilistic model used to categorize data into various groups or classes.

The Naive Bayes Classifier works on the basis of the likelihood of each characteristic in the data to predict the class of the data. It is dubbed "Naive" because it implies that all features are independent of one another. Because of this assumption, the algorithm is computationally efficient and simple to implement.

The following are the steps in the Naive Bayes Classifier:

* Get training data
* Determine the probabilities
* Determine the prior probability
* Use the Bayes theorem
* Create predictions

Posterior probability is the updated likelihood of an occurrence or hypothesis after new data or information is considered. It is determined using Bayes' theorem, which connects the conditional probability of an event given some evidence to the event's prior probability and the likelihood of the evidence given the occurrence. The following is the formula for determining the posterior probability using Bayes' theorem:

P(A | B) = P(B | A) / P(A) (B)

P(A | B) is the probability of event A given evidence B.

P(B | A) represents the probability of the evidence B given event. A

P(A) represents the prior probability of event A.

P(B) denotes the likelihood of witnessing evidence B.

**What is the difference between stemming and lemmatization in NLP?**

NLP uses stemming and lemmatization to reduce words to their roots. Both methods accomplish comparable results, yet they differ.

Stemming removes suffixes and prefixes from a word to reveal its stem. "Running" stems to "run," and "Jumps" stems to "Jump." Stemming uses rules to simplify words.

Lemmatization, on the other hand, uses context and meaning to reduce a word to its lemma. It removes inflectional endings and returns the base form by analyzing vocabulary and morphology. Lemmatizing "running" yields "run," and "jumps" yields "jump." As "good" is the lemma of "better," lemmatizing it yields "good."

Stemming is simpler and faster than lemmatization, but it doesn't examine the word's context or meaning, therefore it may not yield accurate results. Lemmatization, which considers context and meaning, is slower and more complicated, but it yields more accurate results.

**What is Word2Vec and how does it work?**

Word2Vec is a prominent word embedding method in NLP (NLP). It uses neural networks to learn word vector representations (embeddings) from a corpus of text.

Word2Vec places words with comparable meanings near each other in a high-dimensional environment (usually hundreds or thousands of dimensions). This simplifies word semantic similarity computation and comparison.

Word2Vec uses CBOW and skip-gram architectures.

CBOW predicts the current word from its context words, while Skip-gram predicts the context words from the current word. The neural network predicts the probability of a word given its context or a context given a word.

Backpropagation weights each syllable during training. The Word2Vec model can map words to vector representations for downstream NLP applications like sentiment analysis, document categorization, and machine translation.

Word2Vec, a popular NLP technique, can capture semantic links between words, such as analogies (e.g., "king" - "man" + "woman" = "queen"), and can be trained on massive text corpora. Information retrieval, recommendation systems, and chatbots use it.

**When to use GRU over LSTM?**

Recurrent neural networks (RNNs) like Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks are employed in sequential data processing and natural language processing. GRUs and LSTMs both solve the RNN vanishing gradient problem, although they have different uses.

Use GRU over LSTM when:

* GRUs are simpler than LSTMs and train faster. GRUs are better for smaller datasets or fast, efficient applications.
* Simpler jobs: GRUs are superior for simpler tasks that do not require modeling long-term relationships since they have fewer gates and can capture short-term dependencies more efficiently. Language modeling and machine translation involve long-term dependencies, which LSTMs excel at.
* GRUs perform better with smaller datasets because they have fewer parameters and are less prone to overfitting. They're better for data-limited applications.
* Memory constraints: GRUs may be better than LSTMs for mobile devices and embedded systems due to their lesser complexity.

LSTMs are better for modeling long-term dependencies, while GRUs are better for simpler jobs that require fast training and lesser complexity. GRUs or LSTMs depend on the task and resources.