Report 3

Sentiment analysis is a process of analyzing text data to determine the sentiment polarity of the text, which is either positive, negative or neutral.

- can be useful in a variety of applications, such as social media monitoring, customer feedback analysis, and market research.
- Naive Bayes is a probabilistic algorithm commonly used for text classification, including sentiment analysis.

Naive Bayes is based on Bayes' theorem, which is a mathematical formula that calculates the probability of an event based on prior knowledge of conditions that might be related to the event. The algorithm works by assuming independence between features, which means that each feature contributes equally to the probability of the outcome. This assumption makes the algorithm simple and fast, but can also make it less accurate than more complex algorithms that take into account correlations between features.

To perform sentiment analysis using Naive Bayes in Python,

- one needs to collect a dataset of labeled tweets,
- preprocess the tweet text,
- extract features,
- train a Naive Bayes model,
- and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.

Collecting a labeled dataset of tweets involves obtaining a set of tweets that are labeled with their sentiment polarity. This can be done manually by having human annotators read and label each tweet, or by using an automated labeling tool that applies a pre-defined set of rules to classify each tweet.

Preprocessing the tweet text involves cleaning the text by removing unwanted characters, numbers, and punctuation, as well as normalizing the text by lowercasing all letters and removing stop words (common words that do not contribute to the sentiment of the text).

Extracting features involves representing the tweet text as a vector of numerical features that can be used as input to the Naive Bayes algorithm. Common feature extraction techniques include bag of words (counting the frequency of each word in the text), n-grams (counting the frequency of each sequence of n words in the text), and word embeddings (representing each word as a high-dimensional vector based on its context in a large corpus of text).

Training a Naive Bayes model involves fitting the algorithm to the dataset of labeled tweets by estimating the probabilities of each feature given each class (positive, negative, or neutral). This involves calculating the prior probability of each class (based on the number of tweets labeled with each class), as well as the likelihood of each feature given each class (based on the frequency of each feature in tweets labeled with each class).

Evaluating the performance of the Naive Bayes model involves testing the model on a separate dataset of labeled tweets and calculating metrics such as accuracy (the percentage of correctly classified tweets), precision (the percentage of tweets classified as positive that are actually positive), recall (the percentage of positive tweets that are correctly classified as positive), and F1-score (a weighted average of precision and recall that balances the trade-off between false positives and false negatives).

In conclusion, sentiment analysis using Naive Bayes in Python is a simple yet effective way to analyze text data and determine the sentiment polarity of the text. By collecting a labeled dataset of tweets, preprocessing the tweet text, extracting features, training a Naive Bayes model, and evaluating its performance, one can gain valuable insights into the sentiment of social media users, customers, and markets.