**Comparative Analysis of Classification Algorithms for Text Sentiment Analysis**

**Comparative Analysis of Classification Algorithms for Text Sentiment Analysis**

**Introduction:**

Natural language processing relies heavily on sentiment analysis to categorize text input into positive, negative, or neutral attitudes. The three well-liked classification methods for text sentiment analysis— Support Vector Machines, Naive Bayes and Logistic Regression-are compared in this research (Tauzin et al., 2021). The evaluation uses two feature extraction techniques—Frequency-Inverse Document Frequency (TF-IDF).

**Data Loading and Processing:**

The loading and preparation of the datasets start the analysis. The code reads the training and test datasets using the read\_csv function from the panda's package. The variable train\_data contains the test dataset, whereas test\_data contains the training dataset (Tauzin et al., 2021). The text information in both datasets has undergone preprocessing to guarantee uniformity in the study. The text is converted to lowercase using the lower() technique, which lessens the effect of case sensitivity on sentiment analysis. This preprocessing procedure aids in lowering the dimensionality and standardizing the text data. The train\_data and test\_data dataframes now have a new column called processed\_text that holds the processed text data.

**Feature Extraction:**

After the data is loaded and preprocessed, feature extraction techniques were applied to convert the text data into numerical representations that the classification algorithms can use:

The TF-IDF Vectors: The method uses the TfidfVectorizer class from the sklearn.feature\_extraction.text package to extract TF-IDF vectors. TF-IDF considers its relevance by determining a word's frequency both within the document and throughout the full dataset. The TF-IDF matrix is created by fitting the TfidfVectorizer to the training data using the fit\_transform method, and it is then saved in the X\_train\_tfidf variable. The transform technique converts the test data into TF-IDF vectors, and the resultant matrix is saved in X\_test\_tfidf.

**Results:**

The classification algorithms were trained and assessed using the retrieved features after feature extraction. The following are the outcomes for each algorithm and feature extraction technique:

Logistic Regression:

BoW Accuracy: 0.5961

TF-IDF Accuracy: 0.6208

Naive Bayes:

BoW Accuracy: 0.5753

TF-IDF Accuracy: 0.5826

Support Vector Machines (SVM):

TF-IDF Accuracy: 0.3981

With accuracies of 0.5961 and 0.6208, respectively, the Logistic Regression algorithm had the maximum accuracy for the TF-IDF feature extraction approaches (Tauzin et al., 2021). Accuracy values for the Naive Bayes model were 0.5826 for the TF-IDF. With 0.3981 for TF-IDF, the Support Vector Machines (SVM) method demonstrated the least accuracy.

**Conclusion:**

This section covered the preparation and data loading procedures written into the code. The text data was preprocessed and converted to lowercase before the training and test datasets were imported using the read\_csv function. Additionally, we investigated two feature extraction methods to translate the text data into numerical representations: Term Frequency-Inverse Document Frequency (TF-IDF). Following Logistic Regression regarding accuracy among the analyzed algorithms, Naive Bayes emerged from the comparative study of the classification algorithms with the best accuracy. However, the accuracy of Support Vector Machines (SVM) in both feature extraction techniques was lower.

**References**

Tauzin, G., Lupo, U., Tunstall, L., Pérez, J. B., Caorsi, M., Medina-Mardones, A. M., ... & Hess, K. (2021). giotto-tda: A topological data analysis toolkit for machine learning and data exploration. The Journal of Machine Learning Research, 22(1), 1834-1839.