**CODE EXPLANATION:**

1. **LIBRARIES:**

These are libraries that are used and imported**: pandas**: is used for data manipulation and analysis, **numpy**: used for numerical operations, **matplotlib.pyplot**: used for plotting graphs and visualisation, **seaborn**: For statistical data visualization, **re**: For regular expressions and pattern matching, **csv**: For reading and writing CSV files,**random** and **math**: For pseudo random number and mathematical functions, **sklearn.feature\_extraction.text**: For converting text into numerical features, **sklearn.naive\_bayes**: For Naive Bayes classification algorithms, **sklearn.preprocessing**: For preprocessing data (label encoding), **sklearn.metrics**: For evaluating model performance.

1. **READING AND CLEANING DATA:**

The data is first loaded then the training data is read from train\_split.csv and the shape and last 10 rows are printed. The training csv file is read as a string. f\_str defines a pattern to remove the html tags and characters from the csv string and is replaced with an empty string r\_str. f.write(clean\_csv) it writes the cleaned csv string to a new file. After that the cleaned csv file is read into dataframe and the shape and last 10 rows are printed. In data cleaning the datasets are cleaned by removing unwanted characters and symbols.

1. **LABEL ENCODING:**

The categorical labels are converted into numerical labels. The negative labels are encoded as ‘0’ and positive labels are encoded as ‘1’.

1. **READING AND CLEANING VALIDATION DATA:**

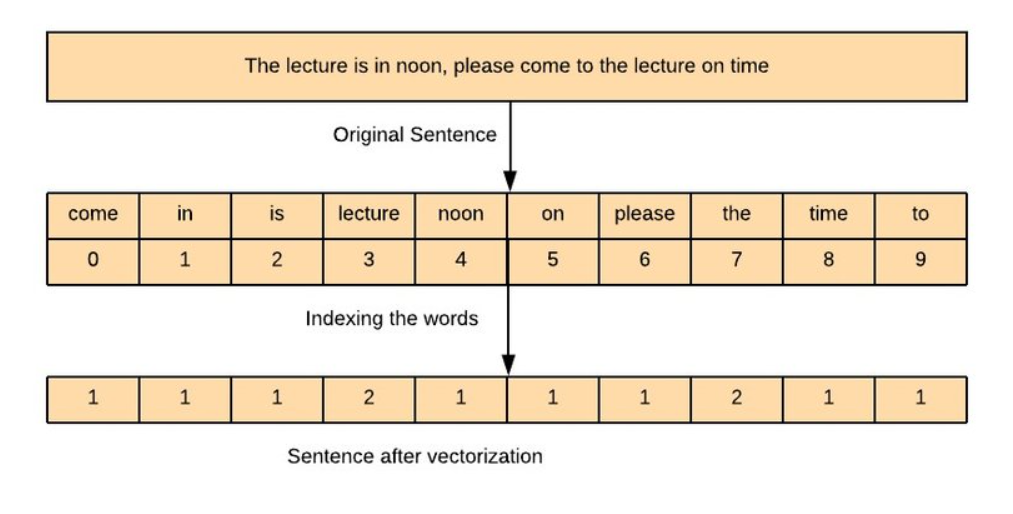
The validated data is read and shape and the last 10 rows are printed. The validated csv file is read as a string. A pattern is defined to remove html tags and characters from validated csv string and replaced with empty string. The cleaned validated string is written to a new file. Then it is read to dataframe and the shape and last 10 rows are printed. Label encoding is done negative label as ‘0’ and positive label as ‘1’.

1. **FEATURE EXTRACTION (VECTORIZATION):**

The text data is converted into numerical features using Count Vectorizer, TF-IDF Vectorizer, and Hashing Vectorizer. The CountVectorizer is used to convert text data into a matrix of token counts (bag-of-words model). The vectorizer fits on the training text data and transforms both training and validation text data into numerical feature vectors they both are vectorized. The shape, type, and array representation of the transformed data is printed.

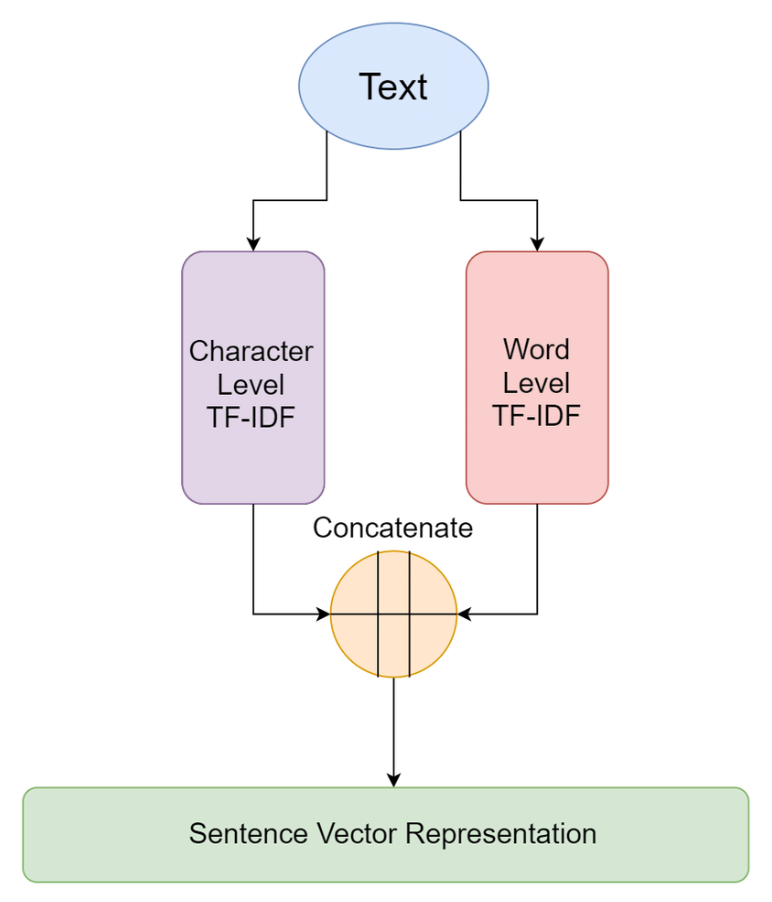
1. **BINARY OCCURRENCE:**

The Hashing Vectorizer is used with binary occurrence to convert text data into a matrix of token occurrences. Both training and validation text data are transformed into binary occurrence feature vectors. The shape and array representation of the transformed data are printed.



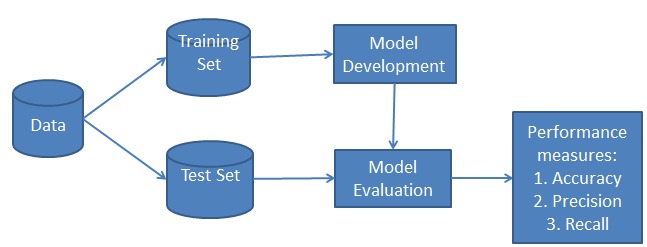
1. **TF/IDF VECTORIZATION:**

TF (Measures how frequently a term occurs in a document.)/IDF (Measures how important a term is in the entire corpus) Vectorizer is used to convert text data into a matrix of TF-IDF features that is it helps in transforming text data into a numerical format that can be used for machine learning models. The vectorizer fuses on the training text data and transforms both training and validation text data into TF-IDF feature vectors. This means it learns the vocabulary and IDF (inverse document frequency) from the training data. The shape and array representation of the transformed data are printed. This is useful for inspection but should be avoided for large datasets due to memory constraints.



1. **MODEL TRAINING AND EVALUATION:**

At first the data is prepared (for frequency count data). Trains Multinomial and Gaussian Naive Bayes models using both Count Vectorized and TF-IDF Vectorized data. The models are evaluated using accuracy, precision, recall, and F1 score. Labels (y\_train, y\_test) and feature vectors (X\_train, X\_test) for training and validation data are extracted. Feature vectors converted to arrays and labels to integers. The shapes of the training and validation feature vectors and labels are printed. Multinomial Naive Bayes model are trained on the TF-IDF vector training data. Labels are predicted for the validation data. The predicted labels, accuracy, precision, recall, and F1 scores are printed same goes for Gaussian Naïve Bayes.

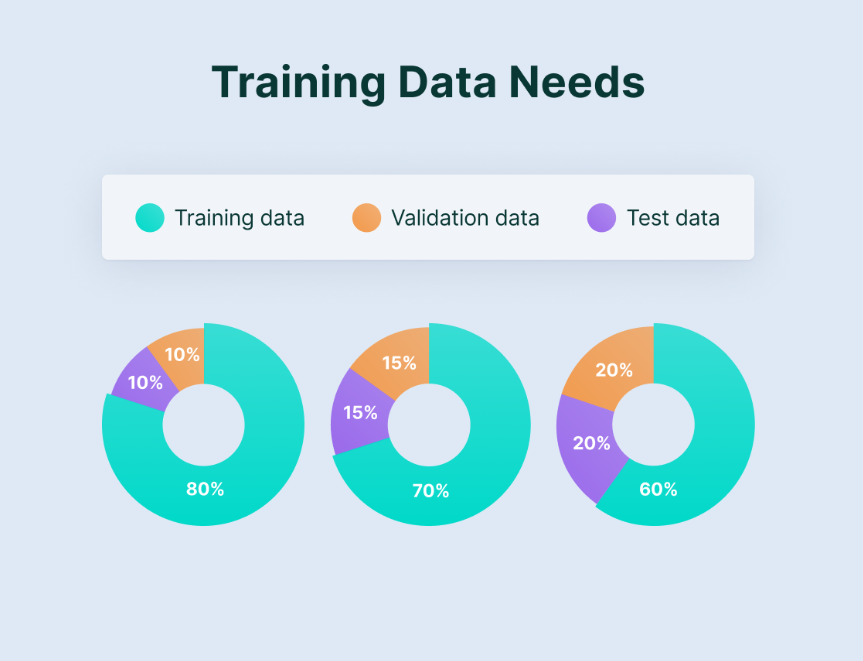


1. **READING AND CLEANING TEST DATA:**

The test data is read and the shape and last 10 rows are printed out. The test csv file is read as a string and a pattern is defined to remove the html tags and characters and is written as a new file. The cleaned test csv file is read into a dataframe and shape and the last 10 rows are printed. Feature extraction for test data is performed frequency count,TF/IDF.

1. **FINAL PREDICTION:**

The labels for the test data are predicted using the Multinomial Naive Bayes model trained on frequency vectors. The predicted labels and their shape are printed. The predictions are saved to a CSV file Freq\_Multinomial.csv. The labels are predicted for the test data using the Gaussian Naive Bayes model trained on frequency vectors. Then predicted labels are printed and saved to a CSV file Freq\_Gau.csv. Labels are predicted for the test data using the Multinomial Naive Bayes model trained on TF-IDF vectors. The predicted labels are printed and saved to a CSV file TFIDF\_Multi.csv. The labels are predicted for the test data using the Gaussian Naive Bayes model trained on TF-IDF vectors. The predicted labels are printed.



1. **CONCLUSION:**

The whole code report helps in understanding the purpose of each step and functionality in machine learning pipeline. The workflow for text classification using Naïve Bayes classifiers with different feature extraction techniques(count vectorizer,TF/IDF) is explained thoroughly. It includes data cleaning, preprocessing, model training, validation, and prediction, and it saves the predictions to CSV files for further analysis. Each step is important because if any of the step is missing it may lead to steps that cannot proceede without the previous step, likewise cleaning improves better model performances, label encoding is crucial as most algorithms require numerical input same goes for the other steps that helps in machine learning, understanding the model and in making real world predictions. This script demonstrates a comprehensive text classification pipeline using various vectorization techniques and Naive Bayes classifiers. The steps include data loading, cleaning, feature extraction, model training, evaluation, and making predictions on new data. The approach ensures that models are evaluated on data they haven't seen during training, providing a measure of their generalization performance before making predictions on the final test set.

