**NLP Sentiment Analysis**

**Problem:**

Analyze the Sentiment dataset using NLP to:

1. View the observations,
2. Verify the length of the messages and add it as a new column,
3. Apply a transformer and fit the data in the bag of words,
4. Print the shape for the transformer, and
5. Check the model for predicted and expected values.

Solution:

Steps Involved are:

1. Read the imdb comments data.
2. Find the length of the messages.
3. Remove punctuations and stopwords. Split the sentence into words. This process is called tokenization.
4. Apply CountVectorizer and Transform. This converts the words to an integer or float. This process is called as Feature Extraction.
5. Apply TF/IDF transform - Term Frequency and Inverse Document Frequency Transform
6. Split data into train and test
7. Using Naive Bayes Classification, first train the model with train data.
8. Test the model and get the prediction.
9. Compare prediction vs Actual and get the confusion matrix.
10. Get the accuracy score for the model.
11. Read the imdb comments data.

Graphical user interface

Description automatically generated with low confidence

1. Find the length of the messages.

Graphical user interface, text

Description automatically generated

3. Remove punctuations and stopwords. Split the sentence into words. This process is called tokenization. Stopwords are the most frequently occurring words like **“a”, “the”, “to”, “for”,** etc. that do not really add value while doing various NLP operations.

Text

Description automatically generated

Text, letter

Description automatically generated

Above screenshots shows that the stop words and punctuations are removed.

1. Apply CountVectorizer and Transform. This converts the words to an integer or float. This process is called as Feature Extraction.

Graphical user interface, text, application

Description automatically generated

Each word is assigned a number based on alphabetical order.

After applying transform, the output is a matrix of token counts as shown below:

Graphical user interface, text, application

Description automatically generated with medium confidence

1. Apply TF/IDF transform - Term Frequency and Inverse Document Frequency Transform.

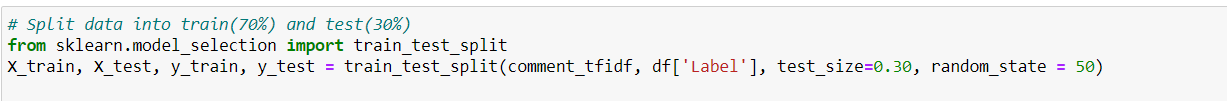
Example: Consider a document containing 100 words wherein the word cat appears 3 times.

The term frequency (i.e., tf) for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 \* 4 = 0.12.

Graphical user interface, text, application, email

Description automatically generated

1. Split the data into train and test.



1. Using Naive Bayes Classification, first train the model with train data.

Text

Description automatically generated

1. Test the model using test data.

Text

Description automatically generated

1. Compare predicted and actual data using Confusion matrix.

Text

Description automatically generated

1. Calculate accuracy score.

User can see that the model has predicted 74+91=165 records correctly . The accuracy score is correct predictions / total number which is 165 / 225 = 0.7333. This means the model is 73.3% accurate.

Graphical user interface, text, application

Description automatically generated