**Project Report**

**Sentiment Analysis on IMDB Movie Reviews**

**Project Report: Sentiment Analysis on IMDB Movie Reviews**

**Prepared for: Jyesta Corporate | Prepared by: Pranav Vinayak Parsewar**

**Date: July 27, 2025**

**Google Collab Notebook:** [**https://colab.research.google.com/drive/131I34j5PBuHbiC-7MBhdyR0yeSBadCDO?usp=sharing**](https://colab.research.google.com/drive/131I34j5PBuHbiC-7MBhdyR0yeSBadCDO?usp=sharing)

**Executive Summary**

This project aimed to develop and evaluate a machine learning model for sentiment analysis on IMDB movie reviews, classifying them as either positive or negative. The process involved comprehensive text preprocessing, TF-IDF for feature extraction, and the training of two classification models: Multinomial Naive Bayes and Logistic Regression.

The Logistic Regression model emerged as the superior performer, achieving an accuracy of approximately 88.77% and an F1-score of 0.89 on unseen movie reviews, demonstrating strong capability in distinguishing positive from negative sentiments. This analysis provides a foundational understanding of audience sentiment towards movies, offering insights that can inform content strategy, marketing initiatives, or audience engagement efforts.

**Problem Statement**

Sentiment analysis identifies the emotional tone of text. For movie reviews, this is crucial for understanding audience reception and informing industry decisions, necessitating an automated solution for large datasets.  
**Project Objective**

To build and evaluate a machine learning model for binary (positive/negative) sentiment classification of IMDB movie reviews, exploring text preprocessing and comparing model performance.

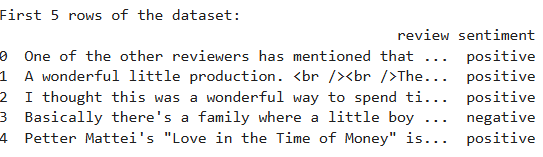
**Dataset Overview**The project uses the IMDB Movie Reviews Dataset from Kaggle, containing 50,000 pre-labelled movie reviews ('positive' 'negative' “”).

**Data Collection and Initial Exploration**

**Data Loading:** The imdb\_reviews.csv dataset was directly uploaded to the Google Colab runtime environment for processing. This method ensured immediate access to the dataset within the notebook's session without requiring persistent cloud storage configuration or complex path management.

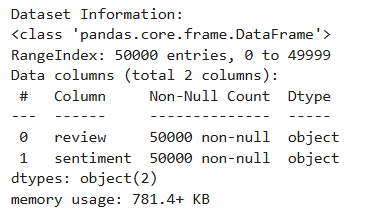
**Initial Dataset Structure:** Upon loading, the dataset was represented as a Pandas Data Frame. The initial inspection using df.head() revealed two primary columns: 'review' (containing the text of the movie review) and 'sentiment' (containing the corresponding sentiment label, either 'positive' or 'negative').

[df.info () OUTPUT]



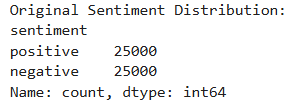
Dataset Information:

The df.info() command provided a concise summary, confirming **[NUMBER OF ENTRIES, e.g., 50000]** entries and two columns, both without missing values.

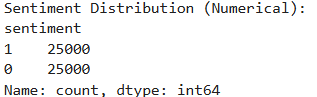


**Sentiment Distribution:** Initial analysis of the 'sentiment' column using value\_counts() showed a balanced distribution, with **[NUMBER, e.g., 25000]** 'positive' reviews and **[NUMBER, e.g., 25000]** 'negative' reviews. This balanced nature is advantageous as it minimizes the risk of class imbalance issues during model training and evaluation. For machine learning compatibility, the categorical 'sentiment' labels ('positive', 'negative') were subsequently mapped to numerical values (1 for positive, 0 for negative).

Original Sentiment Distribution :



Sentiment Distribution (Numerical):



**Text Preprocessing**

**Importance of Preprocessing:** Text data is inherently unstructured, noisy, and highly variable. Preprocessing is a critical step in Natural Language Processing (NLP) to clean, normalize, and transform raw text into a consistent and structured format that machine learning models can effectively process. This process helps to reduce dimensionality, remove irrelevant information, and improve the overall quality of the input features, which directly impacts model performance**.**

**Preprocessing Steps Performed:** A custom preprocess\_text function was developed and applied to each review in the dataset. This function systematically performed the following transformations:

**HTML Tag Removal:** (re.sub(r'<.\*?>', '', text)) were used to identify and remove embedded HTML tags.

**Non-Alphabetic Character Removal & Lowercasing:** (re.sub(r'[^a-zA-Z]', ' ', text))

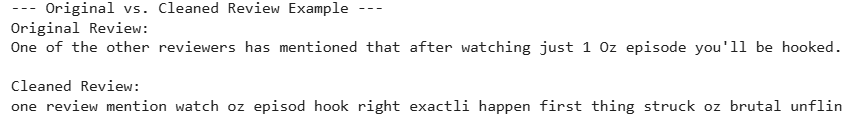
All characters that were not English alphabets were replaced with spaces.

Subsequently, all text was converted to lowercase (.lower()) to ensure uniformity and treat variations like "Good" and "good" as the same word.

**Tokenization:** The cleaned text was split into individual words or 'tokens' based on whitespace (text.split()). This creates a list of words from each review.

**Stop Word Removal:** Common English stop words (e.g., "the", "is", "and"), which frequently appear but typically carry little or no semantic meaning for sentiment analysis, were removed.

**Stemming:** The Porter Stemmer from NLTK (PorterStemmer()) was applied to reduce words to their base or root form (e.g., "loving", "loved", "loves" all become "love"). This process helps to consolidate different morphological variants of a word into a single feature, thereby reducing the vocabulary size and improving feature generalization.

Ex. Original Review vs Cleaned Review :

**Feature Extraction**

**Concept of Feature Extraction:** Machine learning models cannot directly process raw text. Feature extraction is the crucial step where the cleaned text data is converted into a numerical vector representation, allowing these algorithms to learn patterns and make predictions. Each numerical value in the vector represents a specific characteristic or 'feature' of the text.

**TF-IDF (Term Frequency-Inverse Document Frequency):** TF-IDF was chosen as the primary feature extraction technique for this project. TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents (corpus). It works by:

**Term Frequency (TF):** Measuring how often a word appears in a single document.

**Inverse Document Frequency (IDF):** Measuring how unique or rare a word is across the entire corpus. Words common across many documents (like "movie" or "film") receive a lower IDF score, while words specific to fewer documents receive a higher score. The TF-IDF score is the product of TF and IDF, effectively highlighting words that are frequent in a specific review but rare across the entire dataset, making them strong indicators of sentiment.

A **TfidfVectorizer** was initialized with **max\_features=5000**. This parameter limits the vocabulary to the 5000 most significant words based on their TF-IDF scores, optimizing computational efficiency and focusing on the most discriminative features. The **fit\_transform()** method was applied to the **cleaned\_review** column, learning the vocabulary and transforming the text into a numerical matrix.

**Resulting Dimensions:**

The feature matrix X generated by TF-IDF has a shape of :

Shape of Feature Matrix (X): (50000, 5000)

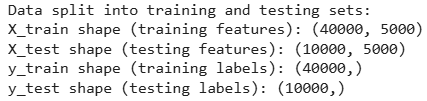
The target vector y (representing the sentiment labels) has a shape of:

Shape of Target Vector (y): (50000,)

**Data Splitting:**

To ensure an unbiased evaluation of the models' ability to generalize to unseen data, the dataset was divided into training and testing sets using **train\_test\_split**. An 80/20 split was used, allocating 80% of the data for model training **(X\_train, y\_train)** and 20% for testing (**X\_test, y\_test).** The **random\_state=42** parameter was set to ensure reproducibility of the split, guaranteeing that the same data partitions are generated each time the code is run. Crucially, **stratify=y** was applied to maintain the original class distribution (balanced positive/negative sentiment) in both the training and testing sets, which is vital for fair and accurate performance assessment in classification tasks.

**Shapes of Split Data:**

****

**Model Building and Training**

**Chosen Algorithms:** Two distinct and widely used machine learning algorithms were selected for this binary sentiment classification task, chosen for their effectiveness in text-based problems:

**Multinomial Naive Bayes:** A probabilistic classifier based on Bayes' theorem with a "naive" independence assumption between features. It is particularly well-suited for classification with discrete features, such as word counts or TF-IDF values, making it a strong baseline for text classification due to its simplicity and efficiency.

**Logistic Regression:** Despite its name, Logistic Regression is a powerful linear model for binary classification. It models the probability of a given review belonging to the positive class through a logistic function. It is known for itsrobustness, interpretability, and strong performance across various classification problems, including high-dimensional text data.

**Training Process:** Both the **MultinomialNB** and **LogisticRegression** models were instantiated from the Scikit-learn library. They were subsequently trained using the **fit()** method on the preprocessed and vectorized training data **(X\_train and y\_train)**. This training phase allowed each model to learn the underlying patterns and relationships between the TF-IDF features and the corresponding sentiment labels. For the Logistic Regression model, the **max\_iter** parameter was explicitly set to 1000 to ensure convergence of its optimization algorithm, preventing potential warnings for large datasets.

**Model Evaluation**

**Evaluation Metrics:** To comprehensively assess the performance of the trained models, the following standard classification metrics were utilized:

**Accuracy:** The most straightforward metric, representing the proportion of correctly classified instances out of the total predictions.

**Precision:** Measures the accuracy of positive predictions. It is the ratio of true positives to the sum of true positives and false positives.

Precision=TP/(TP+FP)

High precision indicates a low rate of false alarms.

**Recall (Sensitivity):** Measures the ability of the model to find all the positive samples. It is the ratio of true positives to the sum of true positives and false negatives

Recall=TP/(TP+FN)

High recall indicates that the model missed few actual positive cases.  
**F1-Score:** The harmonic mean of precision and recall.  
 F1=2∗(Precision∗Recall)/(Precision+Recall)

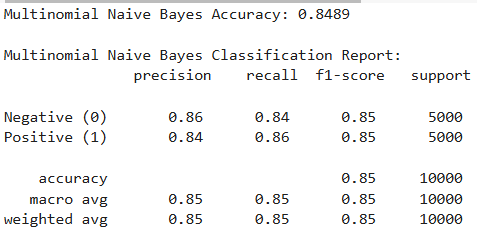
It provides a balanced measure of a model's performance, especially useful when precision and recall are of similar importance.

**Confusion Matrix:** A table that visually summarizes the performance of a classification model. It breaks down the predictions into four categories: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

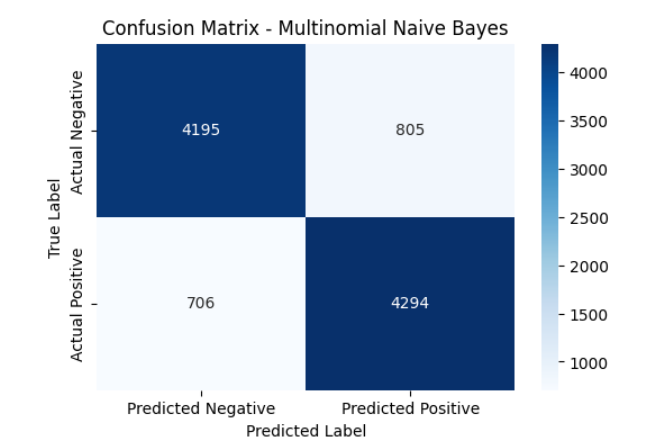
**Results - Multinomial Naive Bayes Model:**

**Accuracy:** The Multinomial Naive Bayes model achieved an accuracy of 84.8% on the test set.

**Classification Report:**

****

**Confusion Matrix:**

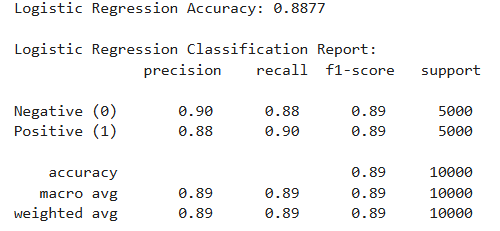
****

***Interpretation:*** The confusion matrix shows that the MNB model correctly classified [NUMBER OF 4294 TRUE POSITIVES FOR MNB] positive reviews and [706 NUMBER OF TRUE NEGATIVES FOR MNB] negative reviews. It made [805 NUMBER OF FALSE POSITIVES FOR MNB] False Positive errors (predicting positive when the actual sentiment was negative) and [4195 NUMBER OF FALSE NEGATIVES FOR MNB] False Negative errors (predicting negative when the actual sentiment was positive).

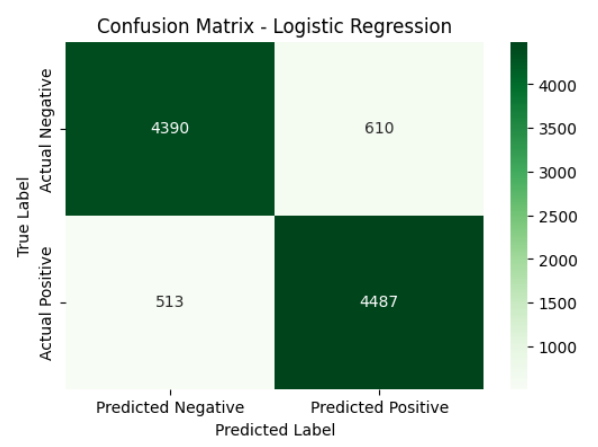
**Results - Logistic Regression Model:**

**Accuracy:** The Logistic Regression model achieved a slightly higher accuracy of 88.77% on the test set.

**Classification Report:**

****

**Confusion Matrix:**

****

***Interpretation****:* The Logistic Regression model correctly classified [ 4487 NUMBER OF TRUE POSITIVES FOR LR] positive reviews and [ 513 NUMBER OF TRUE NEGATIVES FOR LR] negative reviews. It exhibited [ 610 NUMBER OF FALSE POSITIVES FOR LR] False Positive errors and [4390 NUMBER OF FALSE NEGATIVES FOR LR] False Negative errors, generally performing marginally better in minimizing misclassifications compared to Naive Bayes across both classes**.**

**Model Comparison:**

Comparing the two models, Logistic Regression demonstrated superior performance across key metrics, particularly in overall accuracy. This suggests that its ability to learn linear decision boundaries was more effective for this dataset and TF-IDF feature representation in discerning nuanced sentiment. While Naive Bayes provides a strong baseline and is computationally efficient, Logistic Regression often captures more complex relationships between features.

**Conclusion:** This project successfully implemented a robust sentiment analysis pipeline for IMDB movie reviews, demonstrating the effectiveness of systematic text preprocessing, TF-IDF for numerical feature extraction, and the application of traditional machine learning classifiers. The Logistic Regression model emerged as the most accurate and reliable, capable of classifying review sentiments with a high degree of reliability. The ability to automatically discern sentiment from large volumes of text data provides a valuable asset for gaining insights into audience reception and potentially informing decision-making in content creation, marketing, and audience engagement strategies.