**Movie Review Sentiment Analysis – Project Report**

**1) Introduction**

Understanding public sentiment from movie reviews provides valuable insights into audience preferences. In this project, I built a sentiment analysis system using machine learning techniques to classify movie reviews as positive or negative. I worked with a dataset of 50,000 reviews, applying classical ML models — Naive Bayes (NB), Logistic Regression (LR), and Support Vector Machine (SVM) — to evaluate performance and interpret results through visualizations.

As a beginner, I used resources like the YouTube channel **CampusX**, which provided step-by-step guidance for sentiment analysis, and they helped me through the process of building the project. Below are the links to the videos from CampusX that I followed:

1. [Video 1](https://www.youtube.com/watch?v=f73AnhKaKTE)
2. [Video 2](https://www.youtube.com/watch?v=thgIn3OcKnQ)
3. [Video 3](https://www.youtube.com/watch?v=2Zim6B3hBw4)

Additionally, I referred to various articles on the **Towards Data Science** website for further insights into model evaluation and preprocessing techniques.

**2) Approach**

**Data Preprocessing**  
Handling a large dataset of 50,000 reviews was a challenge, especially with limited memory resources. To address this, I loaded the data in chunks of 10,000 rows and processed them sequentially.  
Text cleaning involved:

* Converting text to lowercase
* Removing special characters and punctuation
* Eliminating stopwords
* Applying lemmatization to reduce words to their base form  
  This helped prepare consistent and meaningful input for the models.

**Feature Extraction**  
I used **TF-IDF vectorization** to transform the text into numerical vectors. TF-IDF helped emphasize important words in the reviews while reducing the impact of common words. This representation worked well with all three classical models.

**3) Model Training**

* **Naive Bayes (NB):** Used as a baseline. It’s fast and simple and performed reasonably well on this NLP task.
* **Logistic Regression (LR):** A more flexible linear model, LR improved accuracy and generalized better on unseen data.
* **Support Vector Machine (SVM):** SVM provided the best performance, effectively handling the high-dimensional feature space generated by TF-IDF.

**4) Visualization**

To better understand and evaluate the models and data, I created several visualizations:

* **Word Clouds** for positive and negative reviews, showing the most frequent terms in each category.
* **Confusion Matrices** for each model, providing a clear picture of true vs. false predictions.
* **Bar Graphs** comparing Accuracy and F1 Score across the three models, highlighting their performance differences.

**5) Challenges Faced**

* **Memory Management:** Loading and processing a 50,000-row dataset was resource-intensive. Handling data in chunks was essential.
* **Preprocessing Text:** Finding the right balance between cleaning and retaining meaningful data required trial and error.
* **Vectorization Overhead:** TF-IDF resulted in a high number of features, increasing computational cost during model training.
* **Model Evaluation:** Choosing the right metrics (like F1 score) helped assess performance beyond just accuracy.

**6) Model Performance**

| **Model** | **Accuracy** | **F1 Score** |
| --- | --- | --- |
| Naive Bayes | ~85% | ~84% |
| Logistic Regression | ~88% | ~87% |
| SVM | ~90% | ~89% |

SVM outperformed the other models in both accuracy and F1 score. In the future, I plan to explore deep learning models such as LSTM or pre-trained embeddings like GloVe and BERT for deeper contextual understanding.

**7) Conclusion**

This project gave me a solid understanding of text preprocessing, feature extraction with TF-IDF, and applying classical ML models for sentiment analysis. Although there was no deployment in this version, the focus on data handling, model evaluation, and visual representation made the project highly educational and insightful. It laid a strong foundation for future work involving advanced NLP and model deployment.