Internship Report

Task 4: Sentiment Analysis on YouTube Comments

The objective was to build a model that could read YouTube comments and automatically detect the **sentiment** behind them — whether they were **positive**, **neutral**, or **negative**. This task falls under **Natural Language Processing (NLP)**, a field that focuses on helping machines understand and interpret human language.

Tools & Technologies Used:

- **Google Colab** – for Python-based development
- **Python libraries** – Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
- **Text Vectorization** – TF-IDF (Term Frequency-Inverse Document Frequency)
- **ML Model** – Logistic Regression

Dataset Overview:

- The dataset consisted of **18,408 real YouTube comments**, each labeled with a sentiment.
- There were **three sentiment classes**:
    - positive
    - neutral
    - negative

Step-by-Step Project Process:

1. **Data Loading & Cleaning**
    - Loaded the dataset using Pandas in Google Colab.
    - Found 44 missing comment rows and removed them to ensure clean data.
2. **Exploratory Analysis**
    - Checked total rows and column types.
    - Plotted a **bar chart** showing the number of comments per sentiment using Seaborn.

3. **Text Preprocessing**
    - Converted comments to lowercase.
    - Removed punctuation, numbers, URLs, hashtags, and extra whitespace.
    - Created a new column Cleaned Comment to store the processed text.
4. **Vectorization (TF-IDF)**
    - Transformed the cleaned text into numerical vectors using **TF-IDF**.
    - Limited to **top 5000 words** to reduce noise and improve model speed.
5. **Label Encoding**
    - Encoded sentiment labels:
      negative = 0, neutral = 1, positive = 2
6. **Train-Test Split**
    - Split the data into 80% training and 20% testing using train_test_split.
7. **Model Training (Logistic Regression)**
    - Trained a **Logistic Regression** model on the TF-IDF features.
    - Model learned to classify text into sentiment classes.
8. **Model Evaluation**
    - Achieved **75.82% accuracy** on the test data.
    - Generated a **classification report** with precision, recall, and F1-score.
    - Visualized results with a **confusion matrix heatmap**.

Final Results:

- ✅ **Model Accuracy:** 75.82%
- ✅ **Best performing class:** Positive sentiment (high precision and recall)
- ✅ **Visualization:** Bar chart for sentiment count + confusion matrix
- ✅ **Libraries used:** Sklearn, Matplotlib, Seaborn, Pandas

Conclusion:

This project gave me **hands-on experience with real-world text data** and helped me understand how machine learning can be applied to natural language. I learned how to:

- Clean and preprocess messy text
- Convert text into numerical data using TF-IDF
- Build and evaluate a machine learning model
- Analyse model performance through accuracy and visuals

It also improved my confidence in using tools like Google Colab and Python libraries for practical projects. This task made me realize the power of sentiment analysis in understanding user opinions and emotions — which can be applied in real-life business and media use cases.