**Project Report:**

**Members: Mohsin Ali**

**Abdul Ahad**

**Hassan Abbas**

**Project: Sentiment Analysis of YouTube Comments**

**Scrapping Step**

In this part, we collected YouTube comments using **Selenium**. Since YouTube loads comments dynamically as you scroll, we used code to automatically scroll down and load more comments.

**✅ What the Code Does?**

* Opens Chrome using Selenium.
* Goes to 3 different YouTube video links.
* Scrolls down the page to load more comments.
* Collect all visible comments on the page.
* Stops when it collects 5,000 comments total.
* Saves the comments to a CSV file for later use.

**🛠️ Tools Used:**

* Selenium WebDriver
* ChromeDriver (via webdriver\_manager)

**📂 Output:**

* A file named scraped\_youtube\_comments.csv with 5,000 comments.
* The file has one column: comment.

We successfully scraped 5,000 comments from YouTube videos using Selenium.

**Preprocessing step:**

This preprocessing step cleaned and standardized the YouTube comments. It removed noise, reduced word variations, and saved the final result in a new file. These cleaned comments will now be used for sentiment classification.

**✅ What the Code Does:**

* Loads the scraped comments from the CSV file.
* Removes duplicate and empty comments.
* Converts all text to lowercase.
* Removes links, special characters, emojis, and non-English characters.
* Removes common stopwords like "the", "is", "and", etc.
* Also removes custom words like "youtube", "video", "subscribe", etc.
* Lemmatizes the words (e.g., changing "running" to "run").
* Saves the cleaned comments to a new CSV file.

**🛠️ Tools Used:**

* NLTK (Natural Language Toolkit)
  + Stopwords
  + Tokenizer
  + WordNet Lemmatizer
* Regular expressions (re)

**📂 Output:**

* A file named cleaned\_youtube\_comments.csv with cleaned comments.
* Each comment is processed and ready for sentiment analysis.

**Feature Engineering Step:**

Feature Engineering Section focuses on **analyzing the sentiment** of each comment and **converting the text into numerical features** using TF-IDF. These features will be useful for building ML models to predict sentiment (positive or negative).

**✅ What the Code Does:**

1. **Loads the cleaned comments** from the CSV file.
2. **Calculates sentiment score**:
   * Uses **TextBlob** to get a polarity score for each comment.
   * Score ranges from **-1 (negative)** to **+1 (positive)**.
3. **Applies TF-IDF (Term Frequency-Inverse Document Frequency)**:
   * Converts the text comments into numbers that represent word importance.
   * Uses up to 2,000 most important words (unigrams and bigrams).
4. **Creates a final DataFrame**:
   * Each row contains TF-IDF features of a comment.
   * Adds the sentiment score as a column.
5. **Saves the final data** to a new CSV file for further analysis or modeling.

**🛠️ Tools Used:**

* TextBlob (for sentiment analysis)
* Scikit-learn’s TfidfVectorizer (for feature extraction)

**📂 Output:**

* A file named featured\_youtube\_comments.csv.
* This file contains:
  + TF-IDF features (words turned into numbers)
  + A sentiment score column for each comment

**Algorithm Applying Step:**

We successfully trained a logistic regression model using TF-IDF features and sentiment labels. This model can now be used to automatically classify the sentiment of new YouTube comments.

**✅ What the Code Does:**

1. **Loads the processed data** from the CSV file that contains TF-IDF features and sentiment scores.
2. **Creates sentiment labels**:
   * Comments with a score greater than 0 are labeled **"positive"**.
   * Comments with a score of 0 or less are labeled **"negative"**.
3. **Splits the data**:
   * 80% is used for training the model.
   * 20% is used for testing the model.
4. **Trains a Logistic Regression model**:
   * Logistic Regression is a simple and effective algorithm for binary classification.
   * We used class\_weight='balanced' to handle any imbalance between positive and negative labels.
5. **Fits the model** to the training data so it can learn patterns from the comments.

**🛠️ Tools Used:**

* Scikit-learn (train\_test\_split, LogisticRegression)

**📂 Output:**

* A trained machine learning model that can predict whether a YouTube comment is positive or negative.

**Evaluation Step:**

After training the sentiment classification model, we evaluated how well it performs using the test data.The model was tested on unseen data, and it achieved a good accuracy and balanced performance between positive and negative predictions.

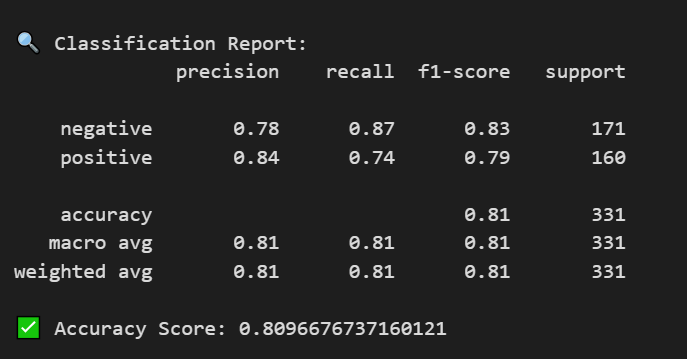
**✅ What the Code Does:**

1. **Makes predictions**:
   * The trained model is used to predict the sentiment of the test comments.
2. **Evaluates the predictions**:
   * classification\_report() shows precision, recall, and F1-score for both **positive** and **negative** labels.
   * accuracy\_score() shows the overall percentage of correctly predicted comments.

**🛠️ Tools Used:**

* Scikit-learn (classification\_report, accuracy\_score)

**📊 Output Example:**



✅ Accuracy Score: 0.8096676737160121 *(Note: Numbers above are just an example.)*