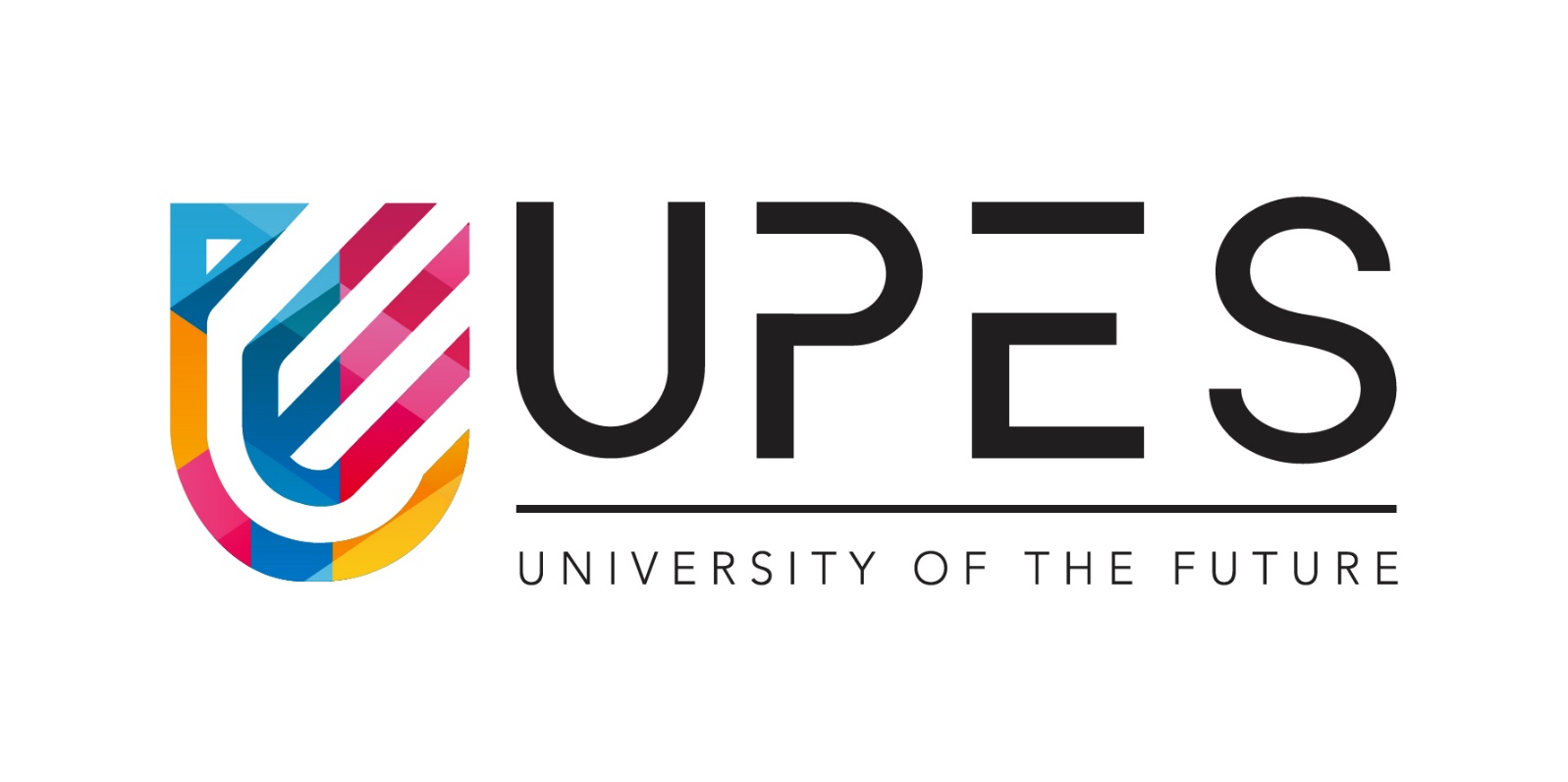
****

**UNIVERSITY OF PETROLEUM & ENERGY STUDIES**

**CUSTOMER SENTIMENT ANALYSIS USING SOCIAL MEDIA DATA**

**WEEKLY REPORT**

**Submitted By:** Shaurya Parashar

**SAP ID**: 500093985

**B.Tech. Computer Science & Engineering**

**BIG DATA**

**GitHub –**

**Week1**

Data Collection

Task: Gather Social Media data and prepare for analysis

In this week, the process of collecting YouTube video data and comments using the YouTube Data API, followed by storing the data in a CSV file. The data collected can be used for various purposes, including sentiment analysis, comment analysis, and video content analysis.

## Requirements

To run the script and collect data, the following Python libraries are required:

* google-api-python-client: To interact with the YouTube Data API.
* pandas: For data manipulation and storage.
* csv: For writing data to a CSV file.

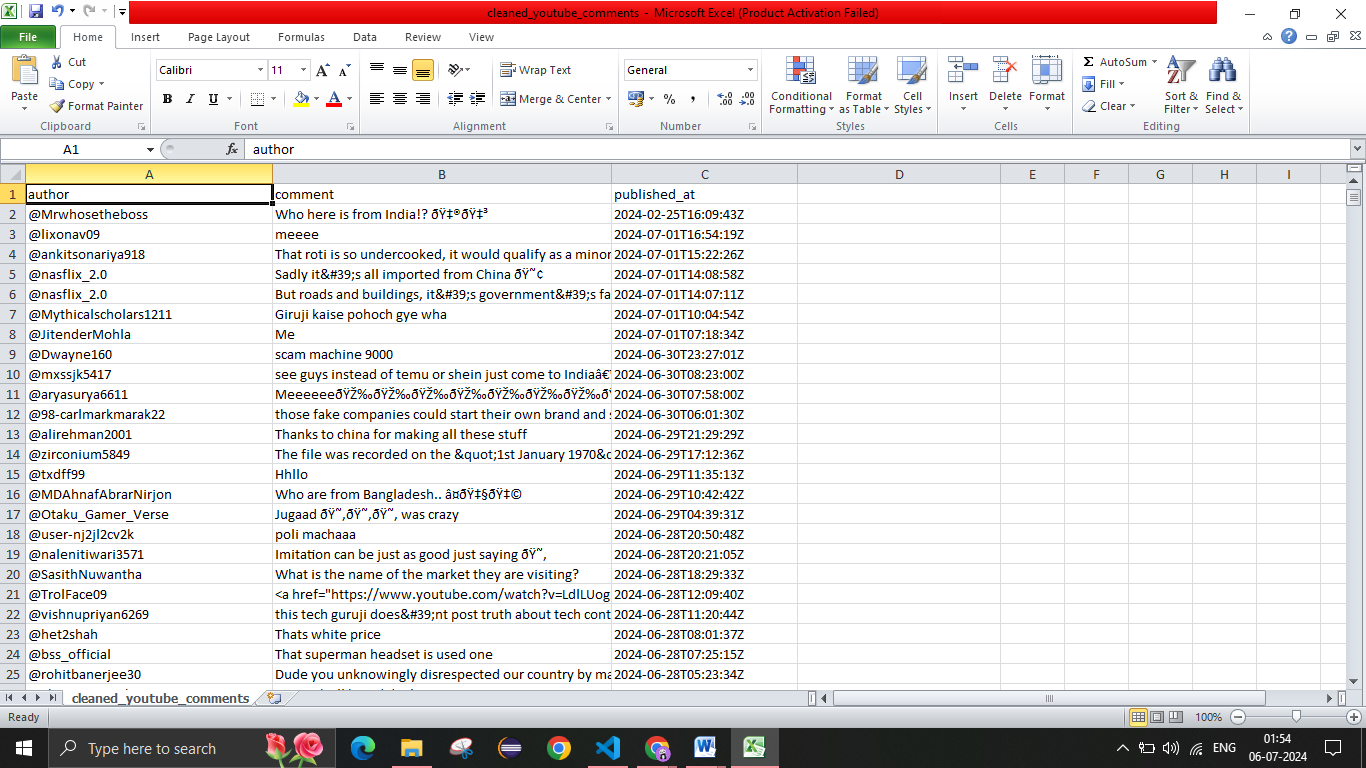
To retrieve comments from a specific YouTube video, the YouTube Data API (google-api-python-client) was used in the first week.

`get\_comments` is defined to retrieve comments repeatedly, handling pagination with `next\_page\_token`, after importing required libraries and initialising the YouTube API client with an API key. The video\_title, author, and comment of each comment are taken out and put into a list. After that, a pandas DataFrame is created from the collected comments to facilitate simple manipulation and analysis. The DataFrame is then produced and stored as `youtube\_dataa.csv}, a CSV file. This allows comments on YouTube videos to be reviewed and stored efficiently.

**Challenges and Solutions**

1. **API Rate Limits:**
   * **Challenge:** Encountered rate limits while making API calls.
   * **Solution:** Implemented pagination and handled nextPageToken to ensure continuous data retrieval.
2. **Data Cleaning:**
   * **Challenge:** Ensuring comments were free from HTML tags and special characters.
   * **Solution:** Used Python's libraries to clean the data.
3. **HDFS Storage:**
   * **Challenge:** Ensuring the data was correctly formatted and transferred to HDFS.
   * **Solution:** Verified data integrity post-transfer using HDFS command-line tools.

THE CSV FILE:



**Week2**

Data Processing and Transformation

Task: To process and transform collected data to make it suitable for analysis

This week focuses on the steps taken to clean YouTube comment data, process it using PySpark and pandas, and perform sentiment analysis using TextBlob. The goal was to clean the data, perform basic transformations and aggregations, and analyze the sentiment of the comments.

**Data Cleaning and Preparation**

The initial steps involved cleaning the data using pandas. The tasks performed were:

1. **Remove Duplicate Rows:** Ensured the dataset had unique entries by removing any duplicate rows.
2. **Fill Missing Values:** Filled any missing values with an empty string to avoid issues during further processing.
3. **Drop Rows with Missing Values:** Removed any rows that still had missing values after filling.
4. **Save Cleaned Data:** Saved the cleaned data to a new CSV file named cleaned\_youtube\_data.csv.

**Data Transformation using PySpark**

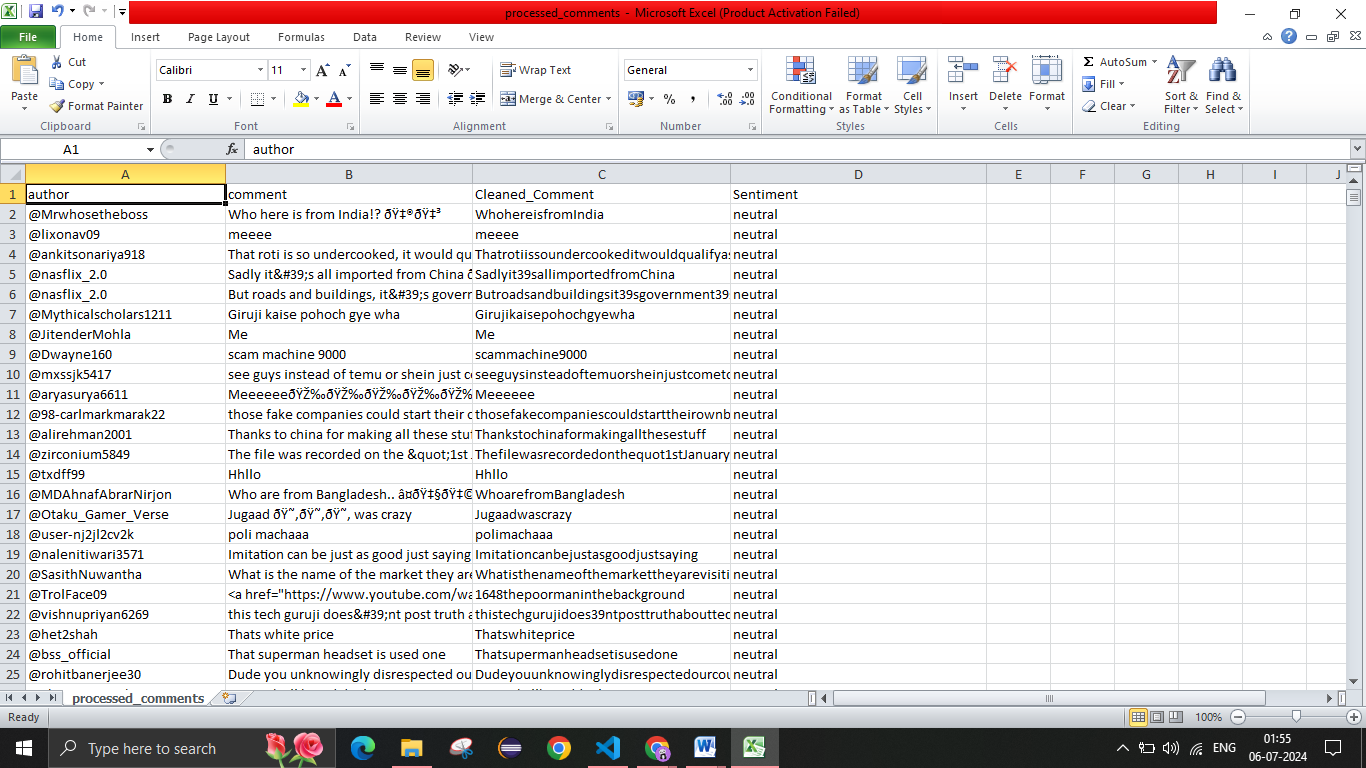
Using PySpark, the cleaned data was loaded and further transformations were performed:

1. **Load Data into Spark DataFrame:** Loaded the cleaned CSV data into a Spark DataFrame.
2. **Filtering:** Removed rows where the comment text was empty.
3. **Aggregation:** Grouped the data by video title and counted the number of comments for each video. Further processing was performed to clean the comment text and analyze the sentiment.

**Preprocessing and analysis**

1. **Load Cleaned Data:** Loaded the cleaned CSV data into a pandas DataFrame.
2. **Clean Comment Text:** Defined a function to clean HTML tags, special characters, emojis, and excessive whitespace from the comment text.
3. **Analyze Sentiment:** Defined a function to analyze the sentiment of the cleaned comment text and classify it as positive, negative, or neutral.
4. **Apply Cleaning and Sentiment Analysis:** Applied the cleaning and sentiment analysis functions to the comment text.
5. **Save Processed Data:** Selected relevant columns and saved the processed data to a new CSV file named processed\_comments.csv.

THE PROCESSED CSV FILE:



**Week-3**

Data Analysis

Task: To analyze the processed data to derive meaningful insight

This week focuses on the data analysis, visualization, sentiment classification, and model saving processes performed on a dataset of YouTube comments. The steps include loading the data, checking its quality, visualizing key insights, performing sentiment classification using a Naive Bayes classifier, and saving the model for future use.

**Data Loading and Initial Analysis**

The first step involved loading the processed data from a CSV file into a pandas DataFrame and performing initial checks:

1. **Data Types and Missing Values:** Checked the data types of each column and counted missing values.
2. **Summary Statistics:** Calculated summary statistics for numerical columns.
3. **Frequency Counts:** Calculated the frequency of each unique value in the categorical columns.

**Data Visualization**

To gain insights from the data, various visualizations were created using Matplotlib and Seaborn:

1. **Distribution of Sentiment Values:** Plotted the distribution of sentiment values.
2. **Top 10 Most Frequent Video Titles:** Plotted the top 10 most frequent video titles.
3. **Top 10 Most Frequent Authors:** Plotted the top 10 most frequent authors.
4. **Word Cloud:** Created a word cloud of the most common words in the comments.

**Sentiment Classification**

A Multinomial Naive Bayes classifier was used to predict the sentiment of the comments:

1. **Data Preparation:** Split the data into features (comment text) and target (sentiment).
2. **Vectorization:** Converted text data into numerical features using CountVectorizer.
3. **Model Training and Prediction:** Trained the Naive Bayes classifier on the training data and made predictions on the test data.
4. **Model Evaluation:** Evaluated the classifier's performance using accuracy, classification report, and confusion matrix.

## Comparison of True vs Predicted Sentiments

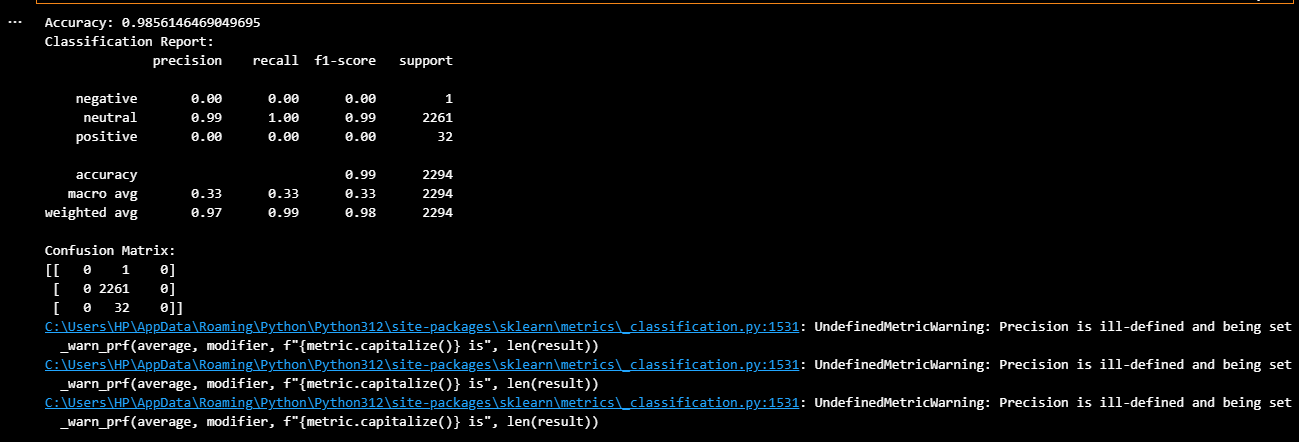
The true and predicted sentiments were compared visually.

The trained Naive Bayes classifier and the CountVectorizer object were saved for future use.

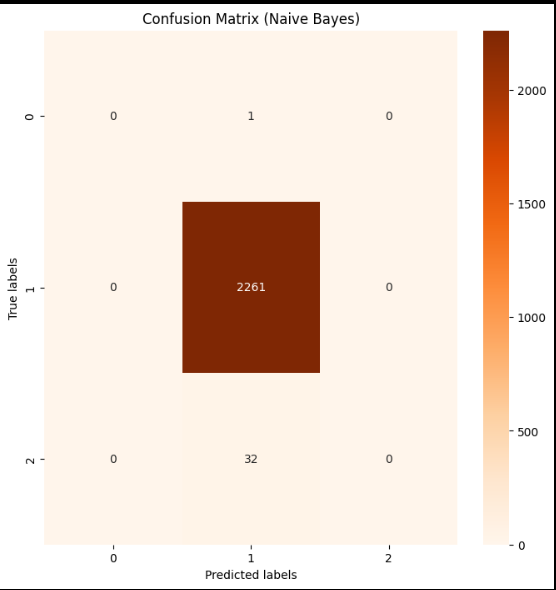
**Challenges and Solutions**

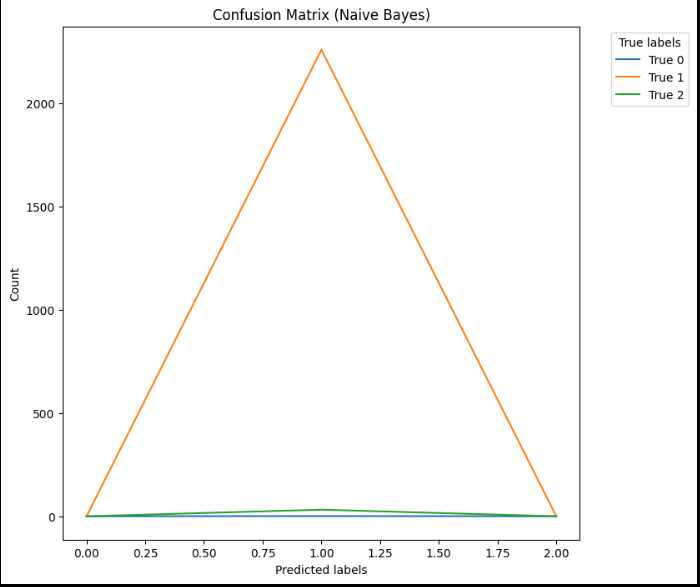
1. **Model Accuracy:**
   * **Challenge:** Achieving high accuracy in sentiment prediction.
   * **Solution:** Fine-tuned the model parameters and scaled the input data for better performance.
2. **Visualization Complexity:**
   * **Challenge:** Effectively visualizing the comparison between true and predicted sentiments.
   * **Solution:** Used seaborn and matplotlib to create clear and informative visualizations.

**OUTPUT:**



**VISULAIZATION:**





**Week-4**

Reporting and Deployment

**Task Summary**

This week was focused on the final deployment of the sentiment analysis project using a Streamlit web application. The app allows users to input comments and receive sentiment predictions (Positive, Negative, Neutral).

**Tasks Completed**

* **Streamlit Application Setup**
* **Description:** Set up the Streamlit environment to develop a web application for sentiment analysis.
* **Details:**
* Created a Streamlit application with navigation options for Home, Predict, and About pages.
* **Pre-processing and Vectorization**
* **Description:** Implemented text pre-processing and vectorization functions.
* **Details:**
* Used spaCy for lemmatization and removal of stop words and punctuation.
* Vectorized text using spaCy's language model to prepare it for sentiment prediction.
* **Model Loading and Prediction**
* **Description:** Loaded the pre-trained sentiment analysis model and integrated it into the Streamlit app for predictions.
* **Details:**
* Loaded the model from a serialized file using joblib.
* Created a function to pre-process, vectorize, and predict the sentiment of user input.
* **User Interface Design**
* **Description:** Designed the user interface for the Streamlit app.
* **Details:**
* Home Page: Introduced the app and its functionality.
* Predict Page: Allowed users to input comments and receive sentiment predictions.
* About Page: Provided information about the project and the developer.
* **Final Testing and Deployment**
* **Description:** Tested the Streamlit application for functionality and performance.
* **Details:**
* Verified the accuracy of sentiment predictions.
* Ensured smooth navigation and user experience across all pages.

**Challenges and Solutions**

* **Model Integration:**
* **Challenge:** Integrating the pre-trained model with the Streamlit application.
* **Solution:** Used joblib to load the model and ensured it worked seamlessly with the app's prediction logic.
* **User Experience:**
* **Challenge:** Designing an intuitive and user-friendly interface.
* **Solution:** Created clear and concise navigation options and ensured the app responded promptly to user inputs.

