BMCS2013 Data Engineering

ASSIGNMENT 202401

Assignment Title : Sentiment Analysis of Shooting Game (Call of Duty Mobile)

Programme / Group : RDS G2

Student Name	Student ID	% Contribution	Signature
CHONG JING YUNG	21WMR04772	16.67	JingYung
ONG WENG KAI	22WMR03309	16.67	WengKai
RYAN KHO YUEN THIAN	22WMR04097	16.67	RyanK
SIM HONG LI	20WMR10587	16.67	HongLi
THONG CHENG HOW	22WMR03154	16.67	ThongCH
YONG ZEE LIN	22MWR03770	16.67	ZeeLin
		100%	

WORK ALLOCATION

Activity	Task	Team Member in Charge
Data Collection	• Web scraping	RYAN KHO YUEN THIAN
	• Data crawling	SIM HONG LI
	• Others (Kaggle)	THONG CHENG HOW
Data Store	• Hive	CHONG JING YUNG
	• MongoDB	ONG WENG KAI
	• Neo4J	THONG CHENG HOW
Data Streaming	• Kafka	YONG ZEE LIN
	• Structured Streaming	RYAN KHO YUEN THIAN
Visualization & Analysis		CHONG JING YUNG

Some functions are duplicated across notebook. Do consider extracting the functions and organising them into separate Python classes grouped according to the purpose of the functions.

Table of Contents

1. Introduction

Title: Sentiment Analysis of Shooting Game (Call of Duty: Mobile)
Domain: Mobile Game Domain involving Natural Language Processing (NLP)
Analytic Problem: Diagnostic Analytics

Call of Duty: Mobile (CODM), a mobile adaptation of the renowned Call of Duty franchise, was explicitly developed for smartphone and tablet users and is known for its engaging first-person shooter gameplay. It features a variety of multiplayer modes, such as Team Deathmatch, Domination, and Battle Royale, all designed with high-quality graphics and customizable controls to ensure a robust gaming experience on mobile devices. This free-to-play game also integrates social features, allowing players to connect and compete, making it a dynamic and accessible option for gamers on the go (Fandom, n.d.). As part of Activision's portfolio, the game not only captivates with immersive gameplay and intricate storylines but also faces the challenge of maintaining its popularity in a rapidly evolving digital landscape. In this context, sentiment analysis emerges as a critical tool for game developers and marketers.

Sentiment analysis, a powerful technique also known as opinion mining, plays a crucial role in the gaming industry. It is used to decipher the emotional tone behind a series of words, providing a deep understanding of the attitudes, opinions and emotions expressed within an online mention, social media post, or review (IBM, n.d.). In the gaming industry, sentiment analysis, exemplified by CODM, is transformative. It is essential for monitoring public opinion across digital platforms, where traditional metrics may be lacking. By analysing comments and reviews using natural language processing (NLP) and machine learning, Activision can gauge whether the sentiment is positive, negative or neutral. For CODM, this analysis is vital due to the industry's responsiveness to player sentiments. By tracking feedback in real-time, Activision can adapt game updates, releases and marketing campaigns to meet player expectations and enhance user experience in future updates. Sentiment analysis also helps in grasping market trends and preferences, crucial for sustaining growth. Moreover, it aids in strategic marketing, leveraging positive sentiments for promotion while addressing negatives to protect the game brand. Hence, it is a cornerstone for adapting and thriving in the competitive gaming landscape (Gameopedia, 2023).

In conclusion, given that there is a massive volume of user-generated data in the gaming world, sentiment analysis is beneficial and indispensable for CODM as it provides a sophisticated means to navigate and succeed in the complex ecosystem of the video gaming industry (Gameopedia, 2023). Through proactive engagement with player feedback across platforms, Activision can ensure that CODM not only remains relevant and enjoyable but also continues to lead as an innovator in the gaming world. This strategic approach not only enhances player satisfaction and loyalty but also solidifies the game's market position, driving ongoing success and development.

Citations from academic articles?

2. Data Sources

2.1. YouTube (Web Scraping)

PySpark
Actual_Web_Scraping_V3.ipynb
- avoid the use of "main" for functions other than the function that acts as the starting point of execution in the program.

YouTube is a platform for sharing videos, enabling users to watch, like, dislike, comment, share and publish their own videos (Saniya, 2023). This means that it is a great source of content for various topics like the game called Call of Duty Mobile (CODM). However, it is challenging to web scrape comments from Youtube as it is a dynamic site. This means that it uses JavaScript to dynamically generate the page content and that the majority of the data are situated within the script tags (Fadheli, 2022; ZenRows, 2023). Therefore, to successfully web scrape Youtube comments, 2 phases are required.

- **First phase of web scraping:**
  We first gathered the urls of 3 Youtube Channels that are focused on CODM. After that, we used **requests.get()** to retrieve the HTML contents of the 3 Youtube Channel pages, which also contain script tags. For each channel url, a **regular expression** is used to extract the urls of the videos that can be found on the channel's page.

- **Second phase of web scraping:**
  Beautiful Soup cannot directly access JavaScript-generated content because it only parses static HTML content, meaning that it cannot access data generated by JavaScript. Therefore, Beautiful Soup cannot be used to web scrape comments from Youtube. Therefore, we have to make use of several concepts like **Youtube Comments Ajax Url**, **user agent**, **requests library**, **regular expression**, etc.

  Youtube Comments Ajax Url refers to the URL for accessing YouTube's comment AJAX service. **User agent** refers to a user agent string used to mimic a web browser when making HTTP requests. The **requests library** is used to make HTTP requests to YouTube's servers and retrieve the HTML responses of the YouTube video pages (using the video urls done in the first phase). **Regular expressions** are used to extract data from HTML and JSON responses.
  Putting all these concepts together, after getting the HTML response from Youtube's servers, it is parsed to extract YouTube configuration data and initial video data. These 2 data are used to facilitate the web scraping process by providing essential parameters and context required for making subsequent requests and enriching the dataset with additional information about the video being scraped. After that, the comment continuations are iterated to retrieve all comments, handling

✓

pagination as needed. Lastly, comment data (such as text, author, votes, dislikes, replies, and ID) are extracted and yielded one by one. Note that during the process of web scraping comments, the comments are stored in **PySpark** dataframes.

2.2. Reddit (Data Crawling)

The primary data source utilised for this project on data crawling is Reddit, a popular social media platform. We extracted data from the "callofdutymobile" subreddit, focusing on discussions and comments related to the Call of Duty Mobile game. We leveraged the PRAW library to access Reddit's API and retrieve data from the specified subreddit. PRAW facilitated the extraction of various attributes from Reddit posts and comments, including titles, scores, IDs, URLs, and content. Utilising PRAW, we accessed the subreddit and retrieved the hottest posts, limiting the scrape to 100 posts per iteration. For each post, we collected relevant information such as title, score, ID, subreddit name, URL, and comment count. We then retrieved all comments for each post to capture additional insights and opinions from the community. Comments were processed to obtain attributes such as body content, score, ID, and creation timestamp. The extracted data was structured into Row objects and appended to a list for further processing.

✓

2.3. Kaggle (CODM dataset)

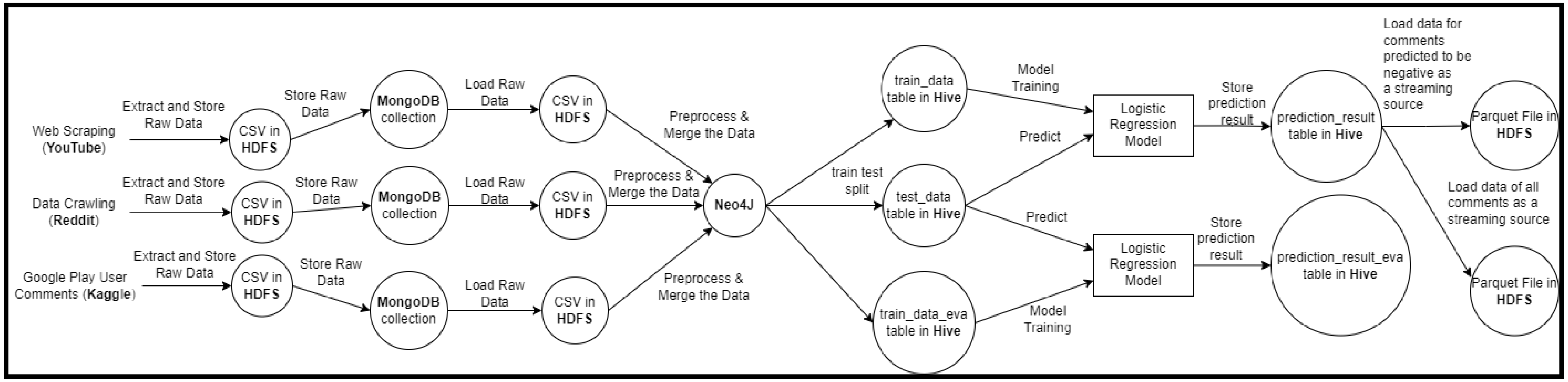
Link to dataset:
https://www.kaggle.com/datasets/munaee/call-of-duty-mobile-playstore-reviews-ratings

This dataset was obtained from Kaggle, which contains reviews and ratings about CODM on Google Play Store and is meant for use in a NLP/ML project. All information within this dataset is authentic and not generated randomly. The author created this dataset by scraping data from Google Play Store.

✓

3. Extract, Transform, Load, Transform (ETLT) Steps

Figure 1 ETLT Steps and Data Stores Used


Web Scraping (YouTube)
Data Crawling (Reddit)
Google Play User Comments (Kaggle)
Extract and Store Raw Data
CSV in HDFS
Store Raw Data
MongoDB collection
Load Raw Data
CSV in HDFS
Preprocess & Merge the Data
Neo4J
train test split
train_data table in Hive
test_data table in Hive
train_data_eva table in Hive
Model Training
Predict
Logistic Regression Model
Store prediction result
prediction_result table in Hive
prediction_result_eva table in Hive
Load data for comments predicted to be negative as a streaming source
Load data of all comments as a streaming source
Parquet File in HDFS

3.1. Data Collection

Task 1:	Getting the Data from **YouTube**
Extract	Gathered the urls of 3 Youtube Channels that are focused on CODM. Extract the urls of the videos that can be found on the pages of the Youtube Channels. For each video url, content like 'comment_text', 'comment_author', 'votes', 'dislikes', 'replies' and 'id' are extracted
Transform	Each record of data extracted from Youtube is initially in the form of a dictionary. Each dictionary of data is subsequently converted into a PySpark dataframe record and appended to a PySpark dataframe that contains all the data.
Load	After converting the PySpark dataframe containing all the data into a csv file, it is loaded into HDFS with the name ‘spark_hdfs_webScrapingv3.csv'

Task 2:	Getting the Data from **Reddit**
Extract	Extracted Reddit data on the subreddit of 'callofdutymobile' and crawled content like 'title', 'post_score', 'post_id', 'subreddit', 'url', 'comment', 'comment_score', 'comment_id', 'comment_created' and 'reply_number'
Transform	Converted the list into a Spark DataFrame named 'df', then verify if the data contains any null values
Load	Converted the data into csv format and loaded it into HDFS with the name ‘spark_hdfs_data_crawlingv3'

Task 3:	Getting the Data from **Kaggle**
Extract	Read the contents from the Kaggle file in Jupyter VM
Transform	Converted the data into a PySpark dataframe
Load	Loaded kaggle dataset ‘hdfs_CallOfDutyv2.csv’ into HDFS

Task 4:	Placing Youtube Data into **MongoDB Atlas** Database
Extract	Extract ‘spark_hdfs_webScrapingv3.csv’ file from HDFS
Transform	Converting the DataFrame rows into a list of dictionaries, where each dictionary represents a row of data.

Load	Load these dictionaries (documents) into a MongoDB collection named 'YoutubeRawData' in the 'AssignmentRawDat' database (in MongoDB Atlas Database)

Task 5:	Placing Reddit Data into **MongoDB Atlas** Database
Extract	Extract 'spark_hdfs_data_crawlingv3' file from HDFS
Transform	Converting the DataFrame rows into a list of dictionaries, where each dictionary represents a row of data.
Load	Load these dictionaries (documents) into a MongoDB collection named 'RedditRawData' in the 'AssignmentRawData' database (in MongoDB Atlas Database)

Task 6:	Placing Kaggle Data into **MongoDB Atlas** Database
Extract	Extract 'hdfs_CallOfDutyv2.csv' file from HDFS
Transform	Converting the DataFrame rows into a list of dictionaries, where each dictionary represents a row of data.
Load	Load these dictionaries (documents) into a MongoDB collection named 'KaggleRawData' in the 'AssignmentRawData' database (in MongoDB Atlas Database)

✓

3.2. Data Preprocessing

Task 1:	Data Cleaning & Preprocessing of all 3 Data Sources
Extract	Extracted data from the MongoDB collections 'KaggleRawData', 'RedditRawData' and 'YoutubeRawData' from database 'AssignmentRawData'
Transform	Performed data preprocessing, text cleaning, generated sentiment and score with NLTK and EDA for visualisation
Load	Loaded data as "WSCleaned.csv", "DCCleaned.csv" and "KGCleaned.csv" into HDFS

Task 2:	Combining and Loading the Data into NEO4J (AuraDB)
Extract	Extracted the files "WSCleaned.csv", "DCCleaned.csv" and "KGCleaned.csv" from HDFS

✓

Transform	Merge the columns “modified_comment”, “review” and “comments” as “modified_comment”, combined the “SentimentScore” and “Sentiment” from the three files.
Load	Joined the “modified_comment”, “SentimentScore” and “Sentiment” columns as combined_df PySpark dataframe. Renamed “modified_comment” to “comments”. Load the combined dataframe into NEO4J (AuraDB)

✓

3.3. Modelling and Visualisation

Task 1:	Storing the Training and Testing datasets into Hive (Round 1)
Extract	Extracted the “Combined” data from NEO4J
Transform	80% of the data was used for training. 20% of the data was used for testing
Load	Under the sentiment_data database in Hive, stored the datasets for train_data table & test_data table

Task 2:	Modelling (Round 1)
Extract	Extracted the training and test datasets from Hive
Transform	● Preprocessed each dataset using the techniques: Tokenization, hashing TF, IDF and Label Encoding in sequence. ● Choose Logistic Regression as the Machine Learning Model. Fit the training data into the model and fine-tuned it using Cross-Validation. ● Evaluated the model’s performance using the testing dataset and produced the model’s predicted sentiments
Load	Loaded the results for prediction into the **prediction_result** table in Hive under the **sentiment_data** database.

Task 3:	Storing the New Training dataset into Hive (Round 2)
Extract	Extracted data from the original training dataset
Transform	64% of the data was used for training. 20% of the data was used for testing (same data as before)
Load	Placed the reduced training dataset in a Hive table called

✓

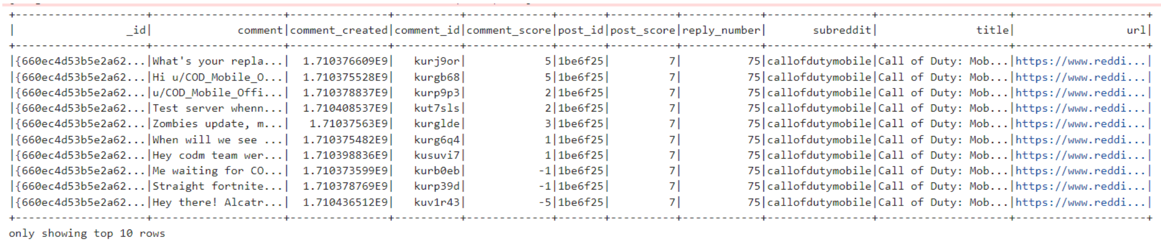
	train_data_eva

Task 4:	Modelling (Round 2)
Extract	Extracted the new training dataset from Hive
Transform	• Preprocessed each dataset using the techniques: Tokenization, Hashing TF, IDF and Label Encoding in sequence • Choose Logistic Regression as the Machine Learning Model. Fit the training data into the model and fine-tuned it using Cross-Validation. • Evaluated the model's performance using the testing dataset (same as before) and produced the model's predicted sentiments
Load	Loaded the results for prediction into the **prediction_result_eva** table in Hive under the **sentiment_data** database.

Task 5:	Writing Data into a Parquet File in HDFS for the purpose of Structured Streaming
Extract	Extracted data from "prediction_result" table in Hive
Transform	Kept the Negative Comments only OR Kept all the comments
Load	Saved the data as a Parquet File in HDFS

4. Data Conditioning

4.1 Data Preprocessing (Data Crawling)



```
+--------------------+--------------------+---------------+----------+-------------+-------+----------+------------+----------------+--------------------+--------------------+
|                 _id|             comment|comment_created|comment_id|comment_score|post_id|post_score|reply_number|       subreddit|               title|                 url|
+--------------------+--------------------+---------------+----------+-------------+-------+----------+------------+----------------+--------------------+--------------------+
|{660ec4d53b5e2a62...|What's your repla...| 1.710376609E9|   kurj9or|            5|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Hi u/COD_Mobile_O...| 1.710375528E9|   kurgb68|            5|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|u/COD_Mobile_Offi...| 1.710378837E9|   kurp9p3|            2|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Test server whenn...| 1.710408537E9|    kut7sls|            2|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Zombies update, m...|  1.71037563E9|   kurglde|            3|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|When will we see ...| 1.710375482E9|   kurg6q4|            1|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Hey codm team wer...| 1.710398836E9|   kusuvi7|            1|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Me waiting for CO...| 1.710373599E9|   kurb0eb|           -1|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Straight fortnite...| 1.710378769E9|   kurp39d|           -1|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
|{660ec4d53b5e2a62...|Hey there! Alcatr...| 1.710436512E9|   kuv1r43|           -5|1be6f25|         7|          75|callofdutymobile|Call of Duty: Mob...|https://www.reddi...|
+--------------------+--------------------+---------------+----------+-------------+-------+----------+------------+----------------+--------------------+--------------------+
only showing top 10 rows
```

Figure 4.1.1

Based on Figure 4.1.1, the dataset consists of the following features: _id, title, post_score, post_id, subreddit, url, comment, comment_score, comment_id, comment_created and reply_number. This dataset alone contains 1359 records.

4.1.1 Explanation of features

Feature name	Meaning
_id	The mongoDB id referencing to MongoDB object
comment	The text of a comment posted by a Reddit user in response to the original post
comment_created	The timestamp/date when the comment was created or posted on Reddit
comment_id	A unique identifier assigned to each comment on Reddit
comment_score	Similar to the post score, this column represents the score or rating of a comment, determined by the number of upvotes minus the number of downvotes it receives from Reddit users.
post_id	Unique identifier assigned to each post on Reddit
post_score	Represents the score or rating of the post in Reddit
reply_number	The number of replies or responses received by a particular comment.
subreddit	Specifies the subreddit to which the post belongs.
title	A brief description of the content or topic of the post.
url	URL or web address associated with the post

4.1.2 Preprocessing steps

- Dropped NAN value for comment field.
  Selected rows containing the phrase 'call of duty' and replace 'call of duty' with 'game'. Since 'call of duty' is a three-word phrase, it may affect our word count graph and sentiment classification. The 'modified_comment' field is the comment where 'call of duty' has been replaced.
- Removed punctuations, numbers, and special characters from the 'Comment' column. These are considered as irrelevant data that can reduce the performance of the classifier (Ahmed et al., 2021).
- Lowercased and broke down the text into individual tokens (Kılınç, 2019), enabling the model to understand the meaning of each word in the context of the sentence. This process facilitates the extraction of relevant features from the text, aiding in the accurate classification of sentiment. By tokenizing the text, the sentiment analysis model can better capture nuances and variations in language, ultimately improving the accuracy of sentiment predictions.
- Lemmatised (Prakash & Aloysius, 2019) and removed stopwords (Kılınç,

2019) from the tokenised text. The purpose is to help normalise words, remove common but less informative words, reduce noise, improve model performance, and enhance computational efficiency in natural language processing tasks.

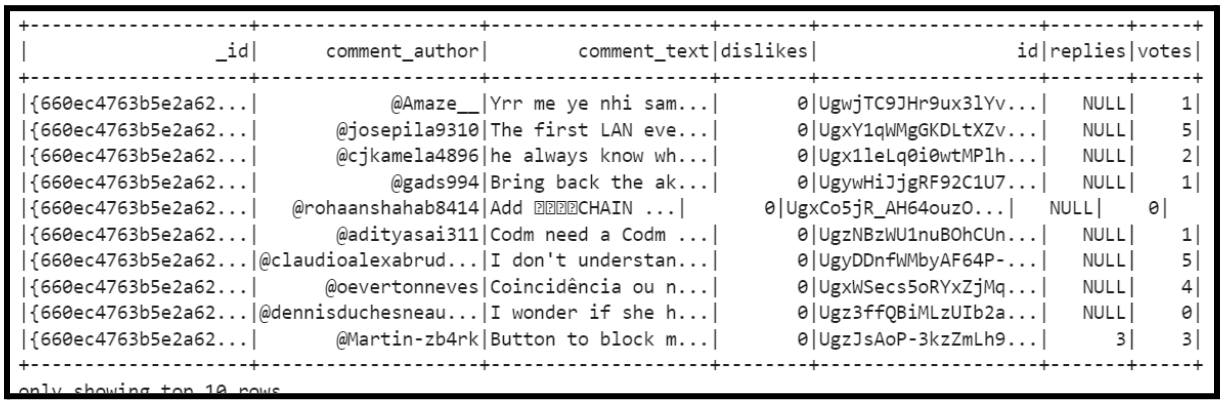
- Concatenated the Tokens for Downstream Analysis by using the concat_ws function. The JoinedComment column represents the tokenized text from the cleaned_tokens column joined into a single string. This concatenated string preserves the original structure of the comment while removing any extraneous characters or spaces. It's useful for downstream analysis or tasks where the entire comment needs to be treated as a single entity.
- After sentiment classification (explained in the next section), the columns for tokens_str and cleaned_tokens were dropped

4.1.3 Sentiment Classification

We used NLTK's VADER sentiment intensity analyzer to generate sentiment scores. The values for sentiment scores range from -1 (negative) to 1 (positive). Negative sentiment occurs for scores below 0, Neutral for a score of 0, and Positive for scores above 0 (Prakash & Aloysius, 2019). Based on the sentiment scores generated by the sentiment analyzer, the sentiments (Positive, Neutral, Negative) were determined and placed in a new column called Sentiment.

✓

4.2 Data Preprocessing (Web Scraping)



```
+--------------------+--------------------+--------------------+--------+--------------------+-------+-----+
|                 _id|      comment_author|        comment_text|dislikes|                  id|replies|votes|
+--------------------+--------------------+--------------------+--------+--------------------+-------+-----+
|{660ec4763b5e2a62...|            @Amaze__|Yrr me ye nhi sam...|       0|UgwjTC9JHr9ux3lYv...|   NULL|    1|
|{660ec4763b5e2a62...|        @josepila9310|The first LAN eve...|       0|UgxY1qWMgGKDLtXZv...|   NULL|    5|
|{660ec4763b5e2a62...|        @cjkamela4896|he always know wh...|       0|Ugx1leLq0i0wtMPlh...|   NULL|    2|
|{660ec4763b5e2a62...|            @gads994|Bring back the ak...|       0|UgywHiJjgRF92C1U7...|   NULL|    1|
|{660ec4763b5e2a62...|   @rohaanshahab8414|Add ����CHAIN ...|       0|UgxCo5jR_AH64ouzO...|   NULL|    0|
|{660ec4763b5e2a62...|       @adityasai311|Codm need a Codm ...|       0|UgzNBzWU1nuBOhCUn...|   NULL|    1|
|{660ec4763b5e2a62...|@claudioalexabrud...|I don't understan...|       0|UgyDDnfWMbyAF64P-...|   NULL|    5|
|{660ec4763b5e2a62...|      @oevertonneves|Coincidência ou n...|       0|UgxWSecs5oRYxZjMq...|   NULL|    4|
|{660ec4763b5e2a62...|@dennisduchesneau...|I wonder if she h...|       0|Ugz3ffQBiMLzUIb2a...|   NULL|    0|
|{660ec4763b5e2a62...|       @Martin-zb4rk|Button to block m...|       0|UgzJsAoP-3kzZmLh9...|      3|    3|
+--------------------+--------------------+--------------------+--------+--------------------+-------+-----+
only showing top 10 rows
```

Figure 4.2.1

Based on Figure 4.2.1, the dataset consists of 4 columns, which are _id, comment_author, comment_text, dislikes, id, replies and votes. This dataset alone contains 800 records.

✓

4.2.1 Explanation of features

Feature name	Meaning
_id	The mongoDB id referencing to MongoDB object

comment_author	The author that commented under YouTube's comment section
comment_text	Comment written by the author
dislikes	The number of dislikes given to the comment
id	YouTube ID given to the comment
replies	Number of replies given to the specific youtube comment
votes	Number of people that agree to that comment (likes)

4.2.2 Preprocessing step

- **Drop unnecessary columns**
  1. Since the **_id** column doesn't give any significant value for the dataset, we removed the _id column from the dataset.
  2. Having implemented code to count unique values for **dislikes**, we observed that the count solely comprises instances of zero dislikes for all the records, indicating an absence of meaningful data to contribute to the modelling stages. Consequently, we opted to remove the "dislikes" column from the dataset.
  3. Considering that the replies column predominantly comprises null values, it does not contribute significant information to the modelling process. Hence, we eliminated the **replies** column.
  4. Since the number of **replies** for half of the records is zero, we can't actually determine the votes based on each column because of complexity factors involved, example: type of english used, the sentence structure, and more. Therefore we decided to remove the **votes** column.

- **Lowercasing**
  To ensure readability and consistency throughout the dataset, we decided to lowercase data of string data types like comment_text.

- **Removing new lines in the column, leading and trailing blanks**
  In this phase of pre-processing string values, we eliminated newlines and both trailing and leading blanks. This helps to clean and standardise text data for analysis, eliminating irrelevant noise, ensuring consistency and improving computational efficiency.

- **Removing special characters and punctuations from comments (Ahmed et al., 2021)**
  This ensures focus on core content, standardised text for consistency, reduces

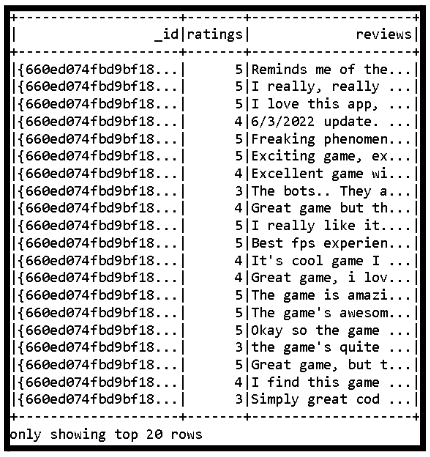
dimensionality for efficient computation and simplifies tokenization

- **Tokenization, lemmatization, count vectorization, feature engineering** (Kılınç, 2019; Prakash & Aloysius, 2019)
  We cleaned up the comments in the dataset by breaking them into words, simplifying them, converting the comments into a matrix of token counts and feature engineering to gain more insights for the comments in the dataset. This helped us understand the comments better before saving everything in the Neo4j database .

4.2.3 Sentiment Classification

To label each comment with a sentiment, we used NLTK's pre-trained model (NLTK VADER sentiment intensity analyzer) to assign a sentiment score to each comment in order to determine the sentiment label (Positive, Neutral and Negative). After the sentiment scores are calculated for the comments, we assigned them to one of three categories: Positive, Neutral or Negative. The assigned sentiments were stored in a new column.
The values for sentiment scores range from -1 (negative) to 1 (positive). Negative sentiment occurs for scores below 0, Neutral for a score of 0, and Positive for scores above 0 (Prakash & Aloysius, 2019).

4.3 Data Preprocessing (Kaggle)



```
+--------------------+-------+--------------------+
|                 _id|ratings|             reviews|
+--------------------+-------+--------------------+
|{660ed074fbd9bf18...|      5|Reminds me of the...|
|{660ed074fbd9bf18...|      5|I really, really ...|
|{660ed074fbd9bf18...|      5|I love this app, ...|
|{660ed074fbd9bf18...|      4|6/3/2022 update. ...|
|{660ed074fbd9bf18...|      5|Freaking phenomen...|
|{660ed074fbd9bf18...|      5|Exciting game, ex...|
|{660ed074fbd9bf18...|      4|Excellent game wi...|
|{660ed074fbd9bf18...|      3|The bots.. They a...|
|{660ed074fbd9bf18...|      4|Great game but th...|
|{660ed074fbd9bf18...|      5|I really like it....|
|{660ed074fbd9bf18...|      5|Best fps experien...|
|{660ed074fbd9bf18...|      4|It's cool game I ...|
|{660ed074fbd9bf18...|      4|Great game, i lov...|
|{660ed074fbd9bf18...|      5|The game is amazi...|
|{660ed074fbd9bf18...|      5|The game's awesom...|
|{660ed074fbd9bf18...|      5|Okay so the game ...|
|{660ed074fbd9bf18...|      3|the game's quite ...|
|{660ed074fbd9bf18...|      5|Great game, but t...|
|{660ed074fbd9bf18...|      4|I find this game ...|
|{660ed074fbd9bf18...|      3|Simply great cod ...|
+--------------------+-------+--------------------+
only showing top 20 rows
```

Figure 4.3.1

Based on Figure 4.3.1 above, the Kaggle dataset consists of 3 features, which are _id, ratings and reviews. In this dataset alone, there are 3882 records.

4.3.1 Explanation of Features

Feature name	Meaning
_id	The mongoDB id referencing to MongoDB object

✓

ratings	The ratings about CODM on Google Play Store
reviews	The reviews about CODM on Google Play Store.

4.3.2 Preprocessing Steps

1. Removed the _id column because it has no meaning.
2. Removed **newline** characters, special characters & punctuations and lowercase the comments in the reviews column (Ahmed et al., 2021) .
3. **Tokenized the reviews (Kılınç, 2019)**: This involves splitting the text, helping to understand each word separately (tokens can check if there is already a column for the tokenized text and it will drop it if it exists) and creating a new column to hold the tokenized text
4. **Removed stop words (Kılınç, 2019)**: We use the filtered_reviews variable to store the cleaned reviews of which the stopwords are removed. Stop words are common words like "is," and "and " that are often removed because they do not carry much meaning for the text analysis, allowing more focus on important content.
5. **Lemmatized the reviews** (Prakash & Aloysius, 2019): We use a lemmatizer to simplify words in the text (like changing words to their basic form), helping in understanding the text better.
6. **Count vectorisation**: This involves converting text data into numerical format

✓

4.3.3 Sentiment Classification

Sentiments were assigned to each review using 2 methods:

- **Classification of Sentiments based on user ratings**

  Negative sentiment occurs for ratings between 0 and 3, Neutral for a rating of 3, and Positive for ratings above 3.
- **NLTK's VADER sentiment intensity analyzer**

  It provides a sentiment score ranging from -1 (negative) to 1 (positive). Negative sentiment occurs for scores below 0, Neutral for a score of 0, and Positive for scores above 0 (Prakash & Aloysius, 2019). For reviews of which

Some functions are duplicated across notebook. Do consider extracting the functions and organising them into separate Python classes grouped according to the purpose of the functions.

their sentiments were classified based on rating, there are 1382 Negative reviews, 570 Neutral reviews and 1930 Positive reviews. For reviews of which their sentiments were classified based on NLTK's VADER sentiment intensity analyzer, there are 1091 Negative reviews, 44 Neutral reviews and 2747 Positive reviews.

The Percentage of similarity between 'Sentiment based on ratings' and 'Sentiment based on SentimentScore' was approximately 60.41%. In the end, we decided to use the sentiments that were based on the sentiment scores assigned by VADER.

✓

5. Implementation

5.1. Data Stores

5.1.1 Hive

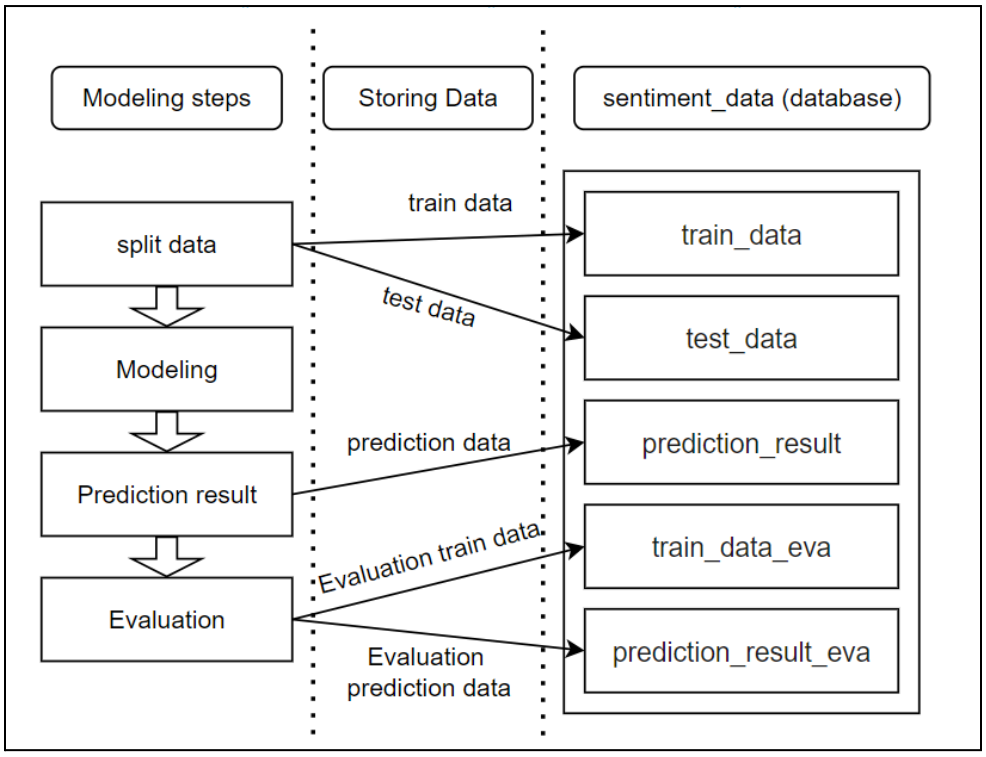


Figure 5.1.1: Flow of storing data to HiveDB in Modeling

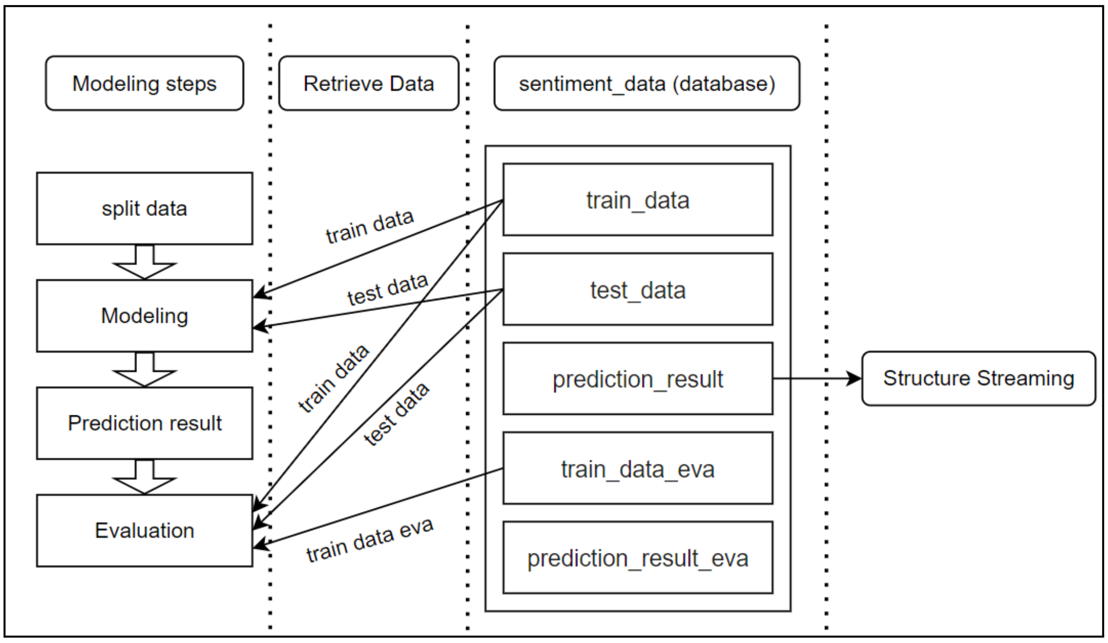
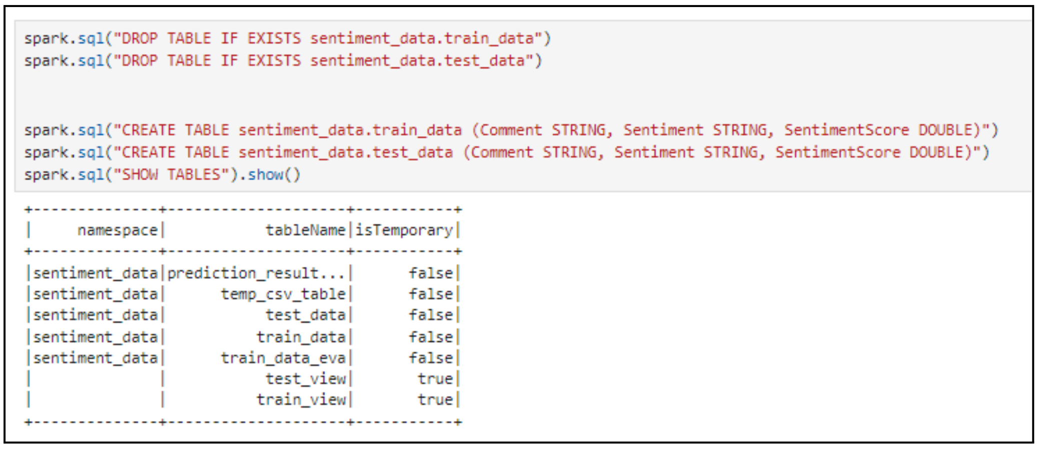


Figure 5.1.2: Flow of retrieve data from HiveDB

- **Creating database and tables for train, test and prediction result**
  It is simple to create tables with the create command in HiveDB. For this assignment, we first created a database called 'sentiment_data'.

Inside that database, we created 5 tables, which are train_data, test_data, prediction_result, train_data_eva and prediction_result_eva. The features we saved are Comment in String data type (to store the comment sentences), Sentiment in String data type (to store the sentiment, which is either Positive, Negative or Neutral, and SentimentScore/Prediction in Double data type (to store the sentiment scores).



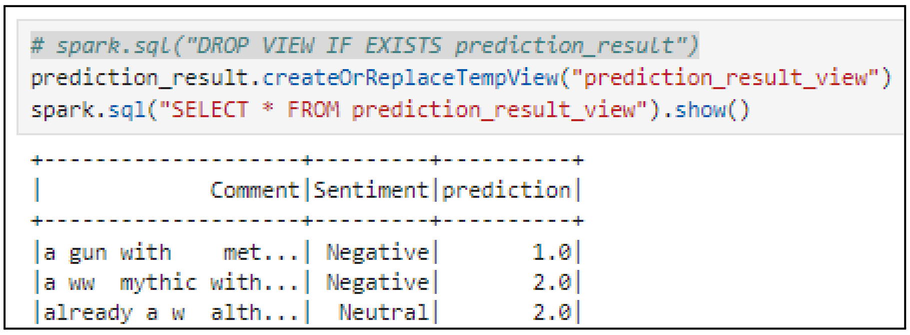
```
spark.sql("DROP TABLE IF EXISTS sentiment_data.train_data")
spark.sql("DROP TABLE IF EXISTS sentiment_data.test_data")


spark.sql("CREATE TABLE sentiment_data.train_data (Comment STRING, Sentiment STRING, SentimentScore DOUBLE)")
spark.sql("CREATE TABLE sentiment_data.test_data (Comment STRING, Sentiment STRING, SentimentScore DOUBLE)")
spark.sql("SHOW TABLES").show()
```

```
+--------------+--------------------+-----------+
|     namespace|           tableName|isTemporary|
+--------------+--------------------+-----------+
|sentiment_data|prediction_result...|      false|
|sentiment_data|      temp_csv_table|      false|
|sentiment_data|           test_data|      false|
|sentiment_data|          train_data|      false|
|sentiment_data|      train_data_eva|      false|
|              |           test_view|       true|
|              |          train_view|       true|
+--------------+--------------------+-----------+
```

Figure 5.1.3: Create table in hiveDB

- **Storing data in HiveDB for every machine learning step**
  HiveDB is useful for storing data using the create view and insert commands. HiveDB was first used to store the training (80% of the data) and testing (20% of the data) datasets. Then, we retrieved the training and testing datasets using the spark hive command, placed them into spark dataframes and proceeded to the preparation and modelling parts. In the modelling part, we stored the prediction results of Logistic Regression in HiveDB of which the results are 0.0 (Positive), 1.0 (Negative), and 2.0 (Neutral).

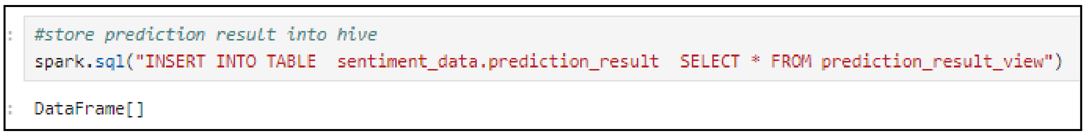


```
# spark.sql("DROP VIEW IF EXISTS prediction_result")
prediction_result.createOrReplaceTempView("prediction_result_view")
spark.sql("SELECT * FROM prediction_result_view").show()
```

```
+--------------------+---------+----------+
|             Comment|Sentiment|prediction|
+--------------------+---------+----------+
|a gun with    met...| Negative|       1.0|
|a ww  mythic with...| Negative|       2.0|
|already a w  alth...|  Neutral|       2.0|
```

Figure 5.1.4: Create View for spark dataframe

✓

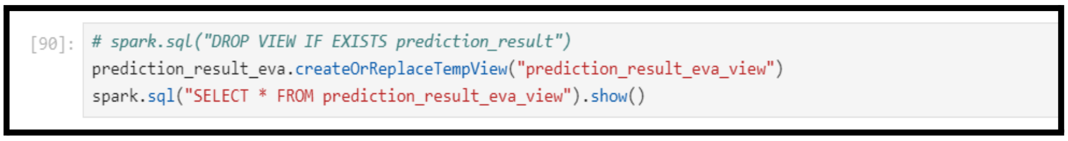


```
#store prediction result into hive
spark.sql("INSERT INTO TABLE  sentiment_data.prediction_result  SELECT * FROM prediction_result_view")

DataFrame[]
```

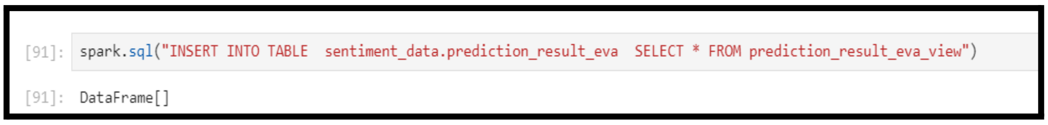
Figure 5.1.5: Insert data into table by View

Since we decided to **compare model performance** by changing the **volume of data**, we modified the existing training dataset so that it is formed from 64% of the data but the testing dataset was left as it is. After that, we stored the training dataset in HiveDB as a new table. Later on, we used the Logistic Regression model to make predictions of which the results are stored as a new table in HiveDB for comparison purposes.



```
[90]: # spark.sql("DROP VIEW IF EXISTS prediction_result")
      prediction_result_eva.createOrReplaceTempView("prediction_result_eva_view")
      spark.sql("SELECT * FROM prediction_result_eva_view").show()
```

Figure 5.1.6: Create View for spark dataframe



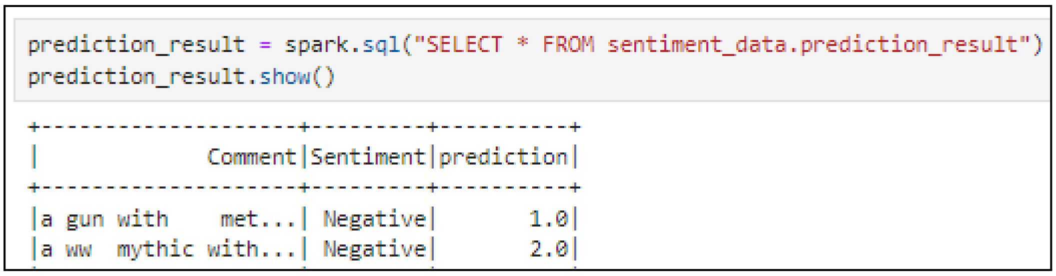
```
[91]: spark.sql("INSERT INTO TABLE  sentiment_data.prediction_result_eva  SELECT * FROM prediction_result_eva_view")

[91]: DataFrame[]
```

Figure 5.1.7: Insert data into table by View

- **Reading data from HiveDB into Data Frame for Various Uses**
  It is easy to retrieve data from HiveDB with the select command as it is similar to SQL. For the modelling part, we retrieved the training and testing data from HiveDB using the select command and performed preprocessing techniques like tokenization. We also read the prediction results from HiveDB into the data frames to perform visualisation (graph and plot) and analytics. The evaluation results, which were stored after the second modelling, were also used to perform comparisons in terms of performance metrics and sentiment predictions, visualisation and reporting. The comments that were predicted to be negative as well as the count & proportion of all sentiment categories were displayed indirectly from the Hive tables via Structured Streaming.

✓



```
prediction_result = spark.sql("SELECT * FROM sentiment_data.prediction_result")
prediction_result.show()
```

```
+--------------------+---------+----------+
|             Comment|Sentiment|prediction|
+--------------------+---------+----------+
|a gun with    met...| Negative|       1.0|
|a ww  mythic with...| Negative|       2.0|
```

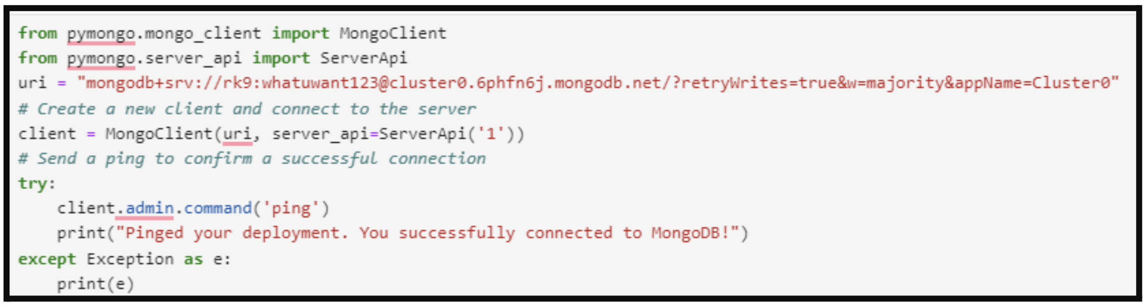
Figure 5.1.8: Retrieve data by select statement

<u>5.1.2 MongoDB</u>

- **Retrieve data from the HDFS file**
  The Raw data were stored in HDFS and needed to be extracted before it can be stored in MongoDB, and it will use the “spark.read.csv” to read the data that are stored in the HDFS. The raw web scraped Youtube data is stored in the “hdfs://10.123.51.194/user/g23/spark_hdfs_webScrapingv3.csv” and it is stored in variable “youtubePath”. For the data crawled Raw Reddit Data it is stored in the “hdfs://10.123.51.194/user/g23/spark_hdfs_data_crawlingv3” and it is stored in variable “redditpath”.

- **Writing data into the HDFS before storing into MongoDB**
  The Kaggle dataset, on the other hand, was first uploaded into jupyterhub at the path "/home/g23/CallofDuty.csv" and later stored in the "df" variable. It was then converted into a PySpark dataframe with the variable "spark_df". Before we wrote the data to HDFS as a csv file, we removed the newline character in the data frame. This is because if we had written to and read the data from HDFS, each newline character would be treated as a new record, resulting in data inconsistency. Thus, before it is stored in hdfs with the variable name "spark_df", the new lines need to be removed using " regexp_replace("reviews", "[\n\r}", "“) ". The "reviews" is the column that is being modified. After that, it will write the data and save it into the directory "hdfs://10.123.51.194/user/g23/spark_hdfs_CallOfDutyv2.csv" with the variable name "kg_df".

- **Store data into MongoDB**



```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
uri = "mongodb+srv://rk9:whatuwant123@cluster0.6phfn6j.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0"
# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))
# Send a ping to confirm a successful connection
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

The Code above is used to connect to the MongoDB atlas.

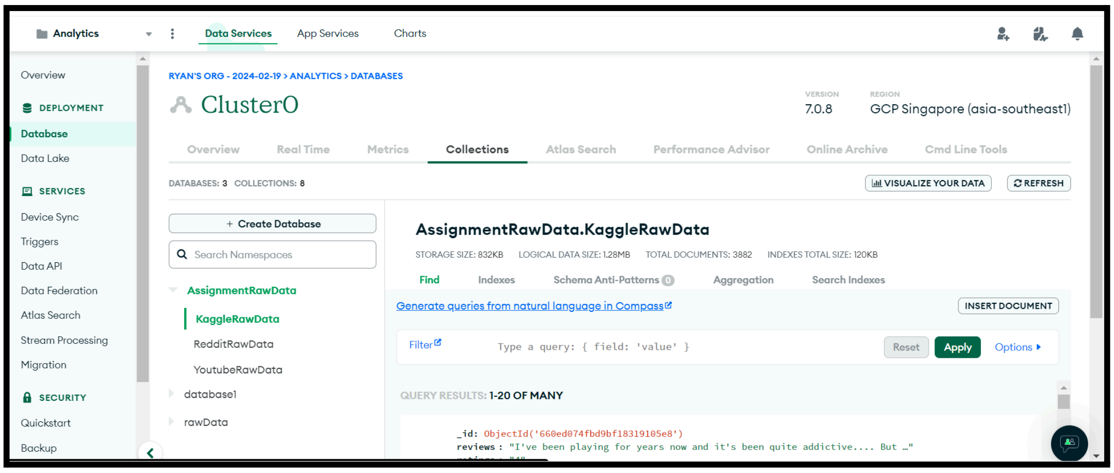
After connecting to MongoDB atlas, the "mydb=client["AssignmentRawData"] "is used to access the "AssignmentRawData" database, where all the data will be stored. The collection within the "AssignmentRawData" database that is used to store data from YouTube (Web Scraping) is called "YoutubeRawData"; for Reddit (Data Crawling), it is called "RedditRawData" and for Kaggle data, it is called "KaggleRawData". After selecting the collection, the "collect()" method is used to get all the data. The "row.asDict() for row in records" is used to convert the data from the row object into a dictionary to be inserted into MongoDB. In the end, the "insert_many( )" method is used to insert the data into the list of dictionary objects in the MongoDB database.

- **Extract Data from MongoDB and convert into Pyspark**

The same theory applies when we try to extract data from MongoDB and store it in Spark. We had to connect to MongoDB atlas and specify the database and the collection(s) to be used. After that, we used "SparkSession.builder" to initiate the spark session, which is the entry point for Spark Functionalities. After that, we set the name of the spark application to "Data Analysis", included the URI and then downloaded the MongoDB Spark connector package through "("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:3.0.1)". This is essential for spark to interact with MongoDB.

The function named "load_data_from_mongodb(collection_name)" is used to read the collection from the database, where the "collection_name" is an argument that can specify the name of the MongoDB collection to be loaded. So, in this case, the "collection_name" can be replaced by "KaggleRawData", "RedditRawData", and" YoutubeRawData" in order to read the Kaggle, Reddit (Data crawling) and YouTube (Web Scraping) data. The "readConfig" is used to specify which collection to read from, and the "spark.read.format("mongo").options(**readConfig).load()" is used to load the data from the MongoDB and load it into the data frame. The "return df" is used to return the loaded data frame. The "df1", "df2", and "df3" are used to load data into the data frames from the collections "KaggleRawData"," RedditRawData", and "YoutubeRawData".

The screenshot below shows that the database name is “AssignmentRawData” with the collection names “KaggleRawData”, “RedditRawData”, and “YoutubeRawData”.



RYAN'S ORG - 2024-02-19 > ANALYTICS > DATABASES

Cluster0

VERSION 7.0.8 REGION GCP Singapore (asia-southeast1)

DATABASES: 3 COLLECTIONS: 8

AssignmentRawData
- KaggleRawData
- RedditRawData
- YoutubeRawData

database1

rawData

AssignmentRawData.KaggleRawData

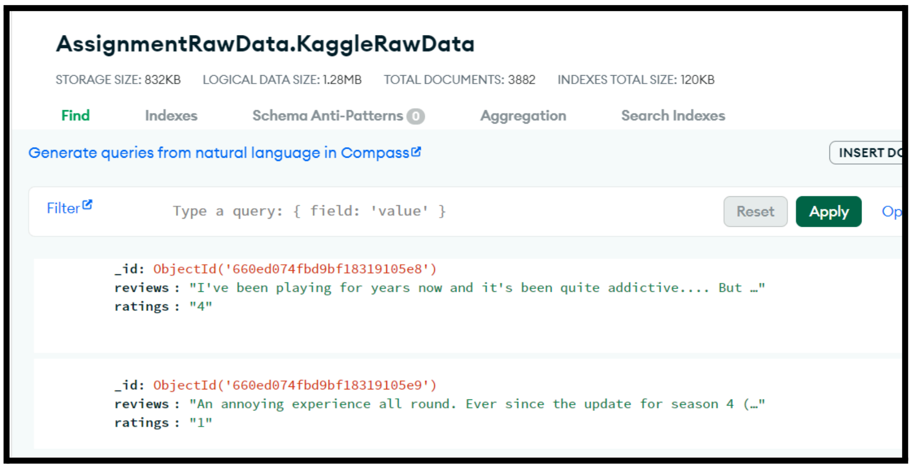
STORAGE SIZE: 832KB LOGICAL DATA SIZE: 1.28MB TOTAL DOCUMENTS: 3882 INDEXES TOTAL SIZE: 120KB

Type a query: { field: 'value' }

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('660ed074fbd9bf18319105e8')
reviews : "I've been playing for years now and it's been quite addictive.... But …"
```

The screenshot below shows the structure of the data stored in the “KaggleRawData” collection, containing 3 fields: “_id”, “reviews” and “ratings”



AssignmentRawData.KaggleRawData

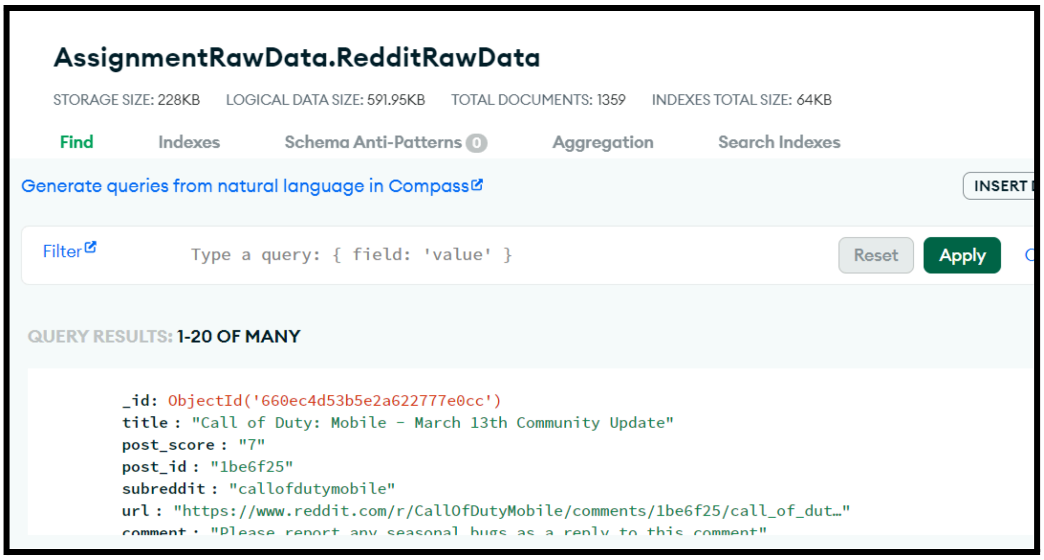
STORAGE SIZE: 832KB LOGICAL DATA SIZE: 1.28MB TOTAL DOCUMENTS: 3882 INDEXES TOTAL SIZE: 120KB

Type a query: { field: 'value' }

```
_id: ObjectId('660ed074fbd9bf18319105e8')
reviews : "I've been playing for years now and it's been quite addictive.... But …"
ratings : "4"

_id: ObjectId('660ed074fbd9bf18319105e9')
reviews : "An annoying experience all round. Ever since the update for season 4 (…"
ratings : "1"
```

The screenshot below shows the structure of the data stored in the “RedditRawData” collection, containing 11 fields: _id, title, post_score, post_id, subreddit, url, comment, comment_score, comment_id, comment_created and reply_number.


AssignmentRawData.RedditRawData

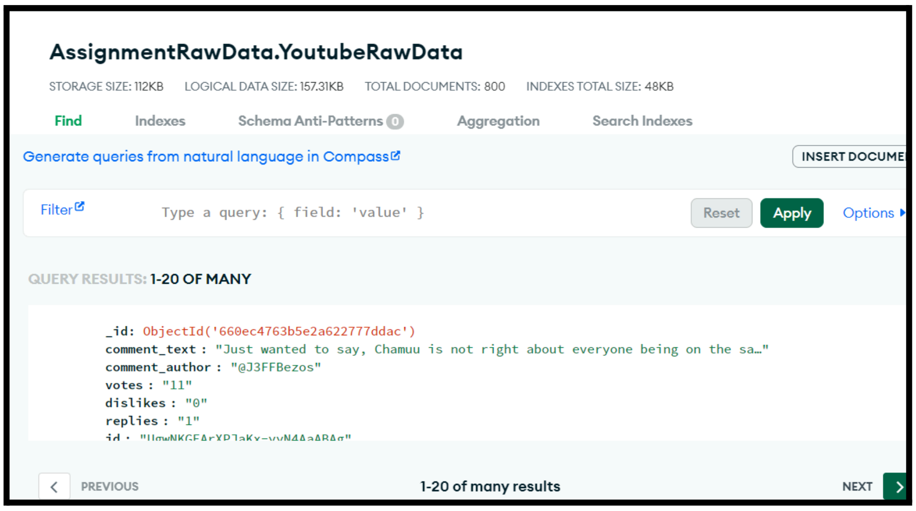
STORAGE SIZE: 228KB LOGICAL DATA SIZE: 591.95KB TOTAL DOCUMENTS: 1359 INDEXES TOTAL SIZE: 64KB

Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT
Filter
Type a query: { field: 'value' }
Reset
Apply

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('660ec4d53b5e2a622777e0cc')
title : "Call of Duty: Mobile - March 13th Community Update"
post_score : "7"
post_id : "1be6f25"
subreddit : "callofdutymobile"
url : "https://www.reddit.com/r/CallOfDutyMobile/comments/1be6f25/call_of_dut…"
comment : "Please report any seasonal bugs as a reply to this comment"
```

The screenshot below shows the structure or the data stored in the "YoutubeRawData" collection, containing 7 fields: _id, comment_text, comment_author, votes, dislikes, replies and id.


AssignmentRawData.YoutubeRawData

STORAGE SIZE: 112KB LOGICAL DATA SIZE: 157.31KB TOTAL DOCUMENTS: 800 INDEXES TOTAL SIZE: 48KB

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('660ec4763b5e2a622777ddac')
comment_text : "Just wanted to say, Chamuu is not right about everyone being on the sa…"
comment_author : "@J3FFBezos"
votes : "11"
dislikes : "0"
replies : "1"
id : "UgwNKGFArXPJaKx-vvN4AaABAg"
```

✓

5.1.3 Neo4J

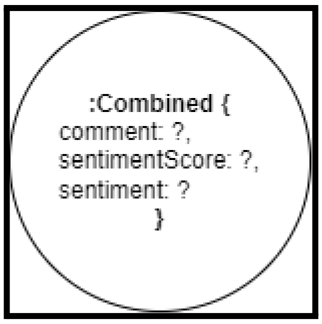
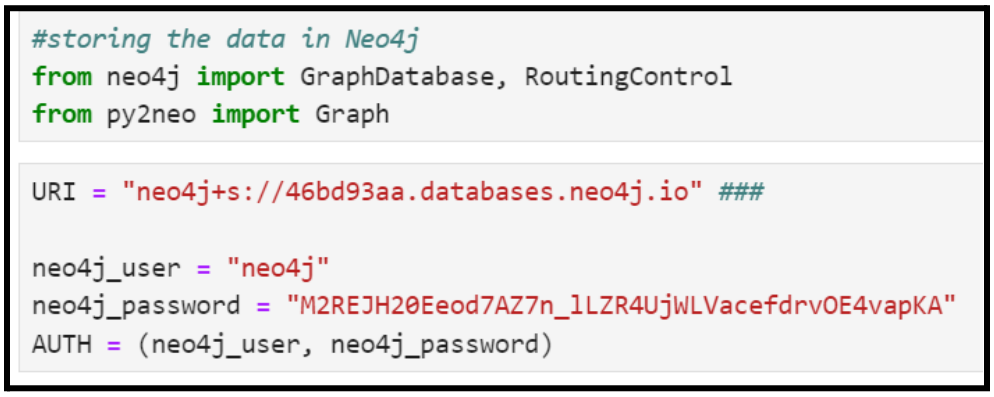


Figure 5.1.3.1: Illustration of a node and its properties inside Neo4j database

- **Getting the cleaned data from each dataset and combining them together**
  We gathered the 3 cleaned datasets and focused on their common features. These features were then concatenated appropriately so that they can be merged into a single dataframe (a unified dataset containing information from all three datasets) in a Neo4j graph database for subsequent modelling tasks.

- **Creating a instance in the Neo4j (Aura DB) website**
  In order to use Neo4j (AuraDB), we had to create a new instance in order to store the data into the AuraDB infrastructure. A username and password were generated by Neo4j for us to access the data inside the Neo4j (AuraDB).

- **Generating the code to store the combined dataset from Jupyter into Neo4j (AuraDB)**
  After we have successfully created a brand new instance in AuraDB, we had to establish a connection between the Jupyter VM node and



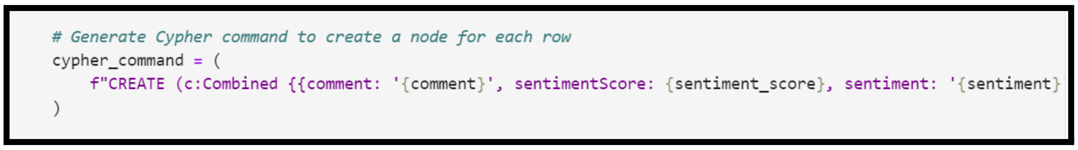
```
#storing the data in Neo4j
from neo4j import GraphDatabase, RoutingControl
from py2neo import Graph
```

```
URI = "neo4j+s://46bd93aa.databases.neo4j.io" ###

neo4j_user = "neo4j"
neo4j_password = "M2REJH20Eeod7AZ7n_lLZR4UjWLVacefdrvOE4vapKA"
AUTH = (neo4j_user, neo4j_password)
```

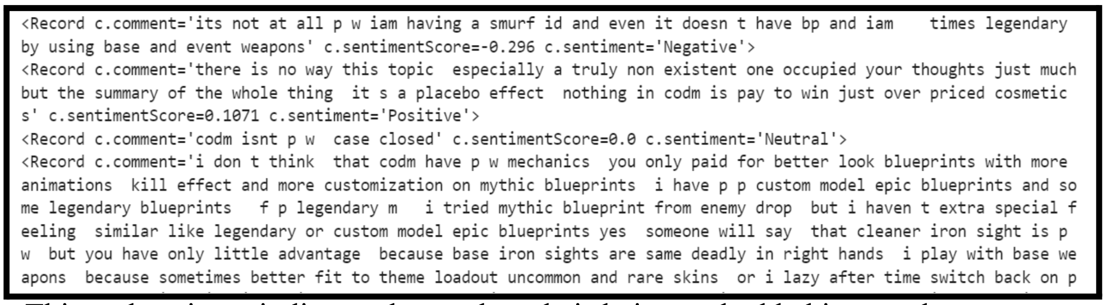
AuraDB to access the information inside AuraDB.

The codes provided here, this is how we establish a connection with AuraDB before we write code inside to AuraDB.



```
    # Generate Cypher command to create a node for each row
    cypher_command = (
        f"CREATE (c:Combined {{comment: '{comment}', sentimentScore: {sentiment_score}, sentiment: '{sentiment}
    )
```

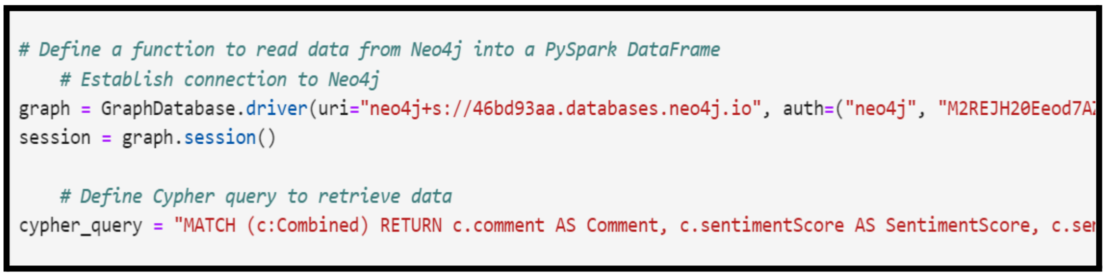
After that, we created a node named **Combined** to store the information for each of the records inside of our combined dataset into Neo4j AuraDB. From figure 5.1.3.1, illustrate the relationship between the node, properties in the unified dataset containing all the three datasets.



```
<Record c.comment='its not at all p w iam having a smurf id and even it doesn t have bp and iam    times legendary
by using base and event weapons' c.sentimentScore=-0.296 c.sentiment='Negative'>
<Record c.comment='there is no way this topic  especially a truly non existent one occupied your thoughts just much
but the summary of the whole thing  it s a placebo effect  nothing in codm is pay to win just over priced cosmetic
s' c.sentimentScore=0.1071 c.sentiment='Positive'>
<Record c.comment='codm isnt p w  case closed' c.sentimentScore=0.0 c.sentiment='Neutral'>
<Record c.comment='i don t think  that codm have p w mechanics  you only paid for better look blueprints with more
animations  kill effect and more customization on mythic blueprints  i have p p custom model epic blueprints and so
me legendary blueprints   f p legendary m   i tried mythic blueprint from enemy drop  but i haven t extra special f
eeling  similar like legendary or custom model epic blueprints yes  someone will say  that cleaner iron sight is p
w  but you have only little advantage  because base iron sights are same deadly in right hands  i play with base we
apons  because sometimes better fit to theme loadout uncommon and rare skins  or i lazy after time switch back on p
```

This code snippet indicates that each node is being embedded into each row and stores it in Neo4j.

- **Extracting the code in Neo4j and displaying using PYspark dataframe**



```
# Define a function to read data from Neo4j into a PySpark DataFrame
    # Establish connection to Neo4j
graph = GraphDatabase.driver(uri="neo4j+s://46bd93aa.databases.neo4j.io", auth=("neo4j", "M2REJH20Eeod7A
session = graph.session()

    # Define Cypher query to retrieve data
cypher_query = "MATCH (c:Combined) RETURN c.comment AS Comment, c.sentimentScore AS SentimentScore, c.se
```

In this code snippet, we wrote a cypher query to extract information about the nodes in Neo4j AuraDB. Now we are able to successfully extract the information from Neo4j to the PySpark dataframe. To check whether all the information is correctly extracted, we counted the number of records to confirm that the data was successfully extracted.

✓

5.2. Streaming

5.2.1. Kafka

The Kafka topic being used is ‘positivec’ (same as positive) and the data being streamed are positive comments. In Activision, both marketers and business analysts benefit from accessing the stream of positive comments related to Call of Duty: Mobile (CODM). Marketers leverage this data to craft targeted campaigns that resonate with players, amplifying positive sentiment to drive engagement and foster a positive community environment. Meanwhile, business analysts use sentiment analysis to evaluate the game's performance, correlating positive feedback with key metrics such as player engagement and revenue, thus informing strategic decisions that maximise the game's impact on Activision's overall business objectives.

To ensure optimal performance and reliability, the Kafka Consumer and Producer must exhibit seamless processing of all messages from the designated Kafka topic, guaranteeing zero data loss. This requires a high level of efficiency and speed in message handling. Achieving this hinges on the meticulous configuration of the consumer. In this specific instance, our target topic is 'positive' and the bootstrap server is hosted at **'10.123.51.194:9092'**. These configuration parameters play a pivotal role in enabling the consumer to establish a robust connection with the Kafka cluster, ensuring a smooth flow of data. Furthermore, thoughtful consideration of additional parameters and settings can further fine-tune the consumer's behaviour, thereby enhancing its overall effectiveness.

In order to ensure the automation of the extraction from Hive prediction_result data, We initiate a loop where the producer is continuously publishing messages of the Kafka topic "positive". Then extracts the value of the message such as comment and sentiment. Assumes that the value is encoded in UTF-8 and decodes it into a string. It treats the decoded value as a JSON string and converts it into a Python dictionary (data_dict). This allows for easier manipulation of the message data.

Lastly, the consumer will only capture the positive sentiment comment and store every message as a single JSON file in the checkpoint folder. The checkpoint data allows the marketing or business analysis department to use it to establish strategy reports or advertising.

Command execution:

- In user **g23 (password: cob0123)**
  - Open a terminal
  - Enter “source ~/de_venv/bin/activate”
  - Enter “python producer.py”
- In **user2 (password: 1)**
  - Open a terminal
  - Enter “source ~/de_venv/bin/activate”
  - Enter “python consumer.py”

5.2.2. Spark Structured Streaming

For this assignment, there are **2 use cases of** Spark Structured Streaming. Their flow is generally like below:

1. Save the prediction results of the Logistic Regression Model from the **prediction_result table** in **Hive** into a **Parquet file** in **HDFS.**
2. Read a data stream from **Parquet file in HDFS** into a **PySpark Dataframe**.
3. Write a data stream from the **PySpark Dataframe** into **Memory**. Results are displayed by querying it.

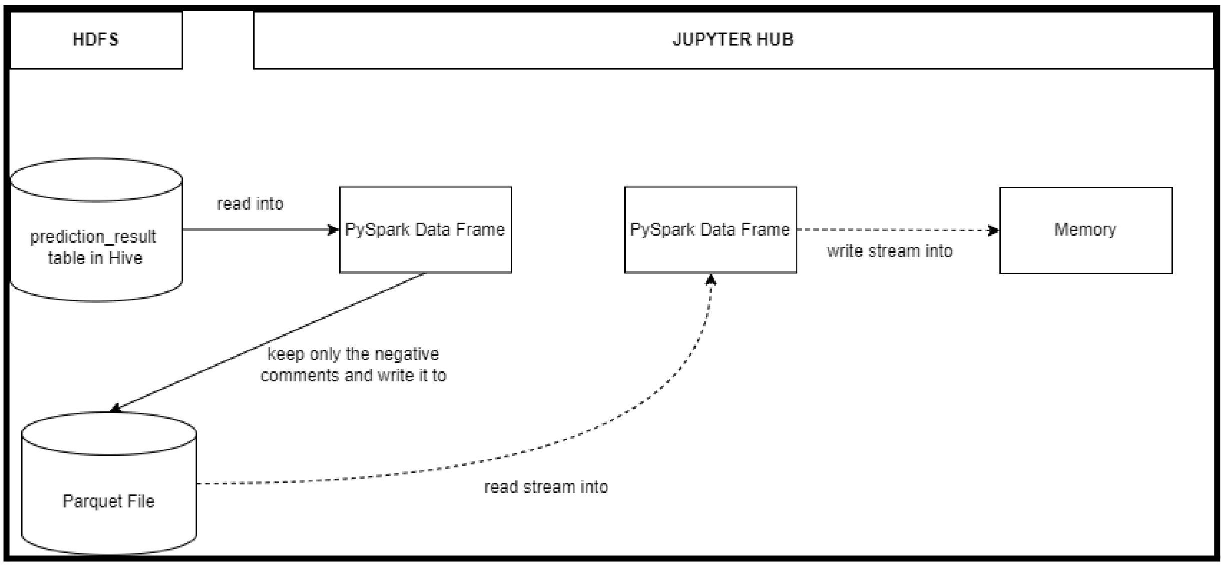
- **Use Case 1: Streaming of Negative Sentiment Comments for the Management’s Viewing**

  By streaming negative sentiments, the CODM company can gain immediate insights into player sentiment as it occurs. This enables prompt responses to any issues or concerns raised by players, leading to better user satisfaction and retention.

  For example, if players start expressing frustration or dissatisfaction with a particular game feature or level, the **game company** can quickly identify and respond to these negative sentiments. They can proactively investigate the root cause of the problem, make necessary adjustments or fixes, and communicate updates or solutions to the player community in real-time.

  As a result, this rapid response to negative sentiments can demonstrate the company's commitment to listening to its players and continuously improving the gaming experience. Refer to the figure below on how this is achieved.

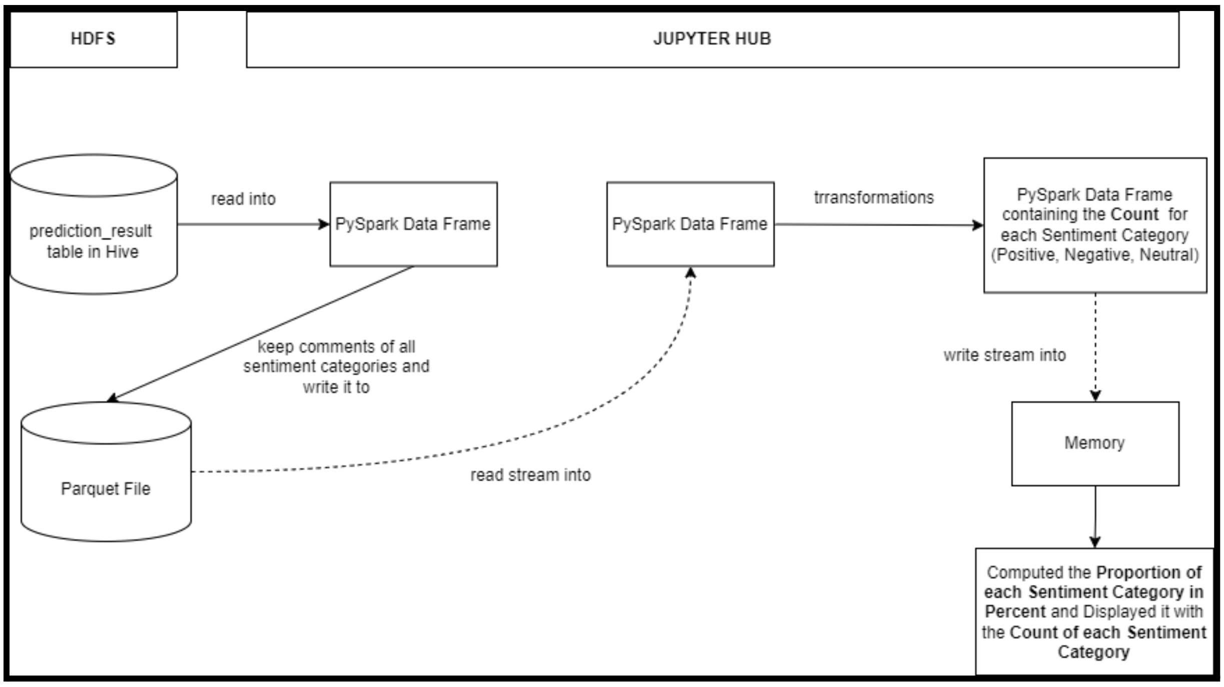
✓



- **Use Case 2: Display the Proportion of Each Sentiment i.e. Positive, Negative and Neutral**
  Displaying only the proportion of negative sentiments provides an incomplete picture of the overall sentiment landscape within the streaming data. While negative sentiments are important indicators of player dissatisfaction or issues within the game, focusing solely on negative sentiment proportions has several limitations. For example, CODM will lose out on the ability to compare the proportions of positive, negative and neutral sentiments over time or against benchmarks, losing out valuable insights into changes in player sentiment and the effectiveness of interventions or improvements.

  Therefore, Spark Structured Streaming is also used to allow the count and percentage of each sentiment category to be communicated to the **stakeholders of CODM**. With this, CODM can gain insight into how sentiments are distributed within the streaming data. This information can help in understanding the overall sentiment landscape and identifying prevalent sentiments among players, leading to informed decision-making at various levels within CODM. Refer to the figure below on how this is achieved.

✓


HDFS
JUPYTER HUB
prediction_result table in Hive
read into
PySpark Data Frame
PySpark Data Frame
trransformations
PySpark Data Frame containing the Count for each Sentiment Category (Positive, Negative, Neutral)
keep comments of all sentiment categories and write it to
write stream into
Memory
Parquet File
read stream into
Computed the Proportion of each Sentiment Category in Percent and Displayed it with the Count of each Sentiment Category

5.3. Spark Machine Learning

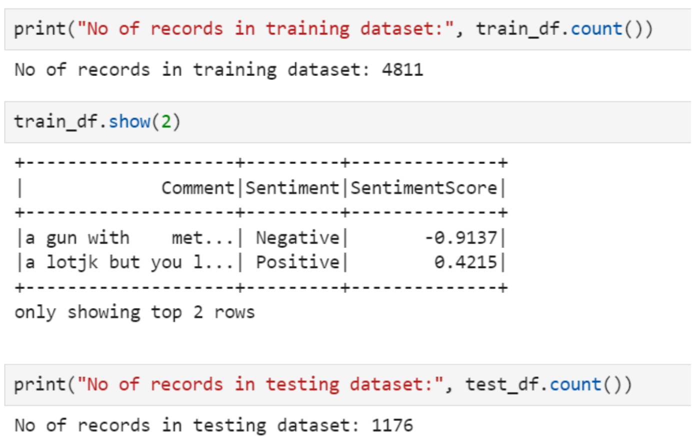
5.3.1. Experimental Design

We planned to compare the model's performance by changing the volume of data.

i) Experiment 1 (80% of the data used for training)

We planned to perform a train-test-split of 80:20, where 80% of the records (4811 records) are assigned to the training dataset while the other 20% are assigned to the test dataset (1176 records).

After training the Logistic Regression model and using it to make predictions, we planned to compute the model's accuracy as well as the precision, recall and f1-score for each of the sentiment classes.



```
print("No of records in training dataset:", train_df.count())
```

```
No of records in training dataset: 4811
```

```
train_df.show(2)
```

```
+--------------------+---------+--------------+
|             Comment|Sentiment|SentimentScore|
+--------------------+---------+--------------+
|a gun with    met...| Negative|       -0.9137|
|a lotjk but you l...| Positive|        0.4215|
+--------------------+---------+--------------+
only showing top 2 rows
```

```
print("No of records in testing dataset:", test_df.count())
```

```
No of records in testing dataset: 1176
```

Figure 5.3.1.1: Number of records for training and testing in Experiment 1

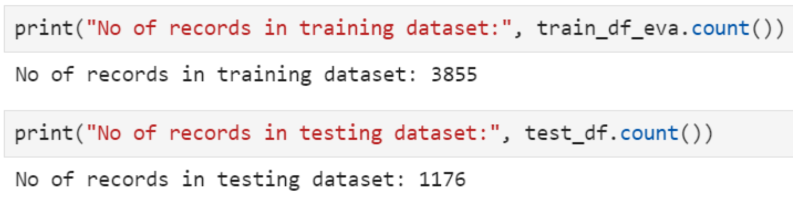
✓

ii) Experiment 2 (64% of the data used for training)

We planned to use 64% of the data for the training dataset (3855 records) but leave the testing dataset as it is (1176 records).

After training the Logistic Regression model on the reduced training dataset and using it to make predictions, we planned to compute the model's accuracy as well as the precision, recall and f1-score for each of the sentiment classes.

Lastly, we planned to compare the performance metrics between the 2 experiments.



```
print("No of records in training dataset:", train_df_eva.count())

No of records in training dataset: 3855

print("No of records in testing dataset:", test_df.count())

No of records in testing dataset: 1176
```

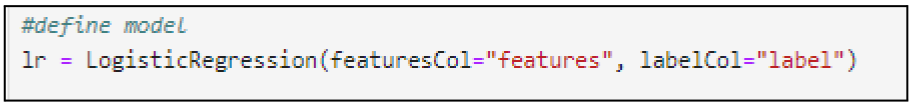
Figure 5.3.1.2: Number of records for training and testing in Experiment 2

5.3.2. Model Building

In general, we performed modelling and prediction twice. In the first round of modelling and prediction, 80% of the data was used for training while 20% was used for testing. In the second round of modelling and prediction, 64% of the data was used for training while the testing dataset remained the same.

a) Define Model

First, we need to choose an appropriate model to solve our classification problem at hand. In this case, we chose the Logistic regression model because it is a commonly used algorithm for binary and multiclass classification tasks due to its simplicity and interpretability like our assignment sentiment analysis.

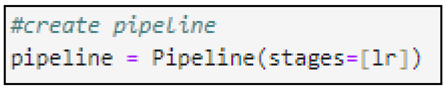


```
#define model
lr = LogisticRegression(featuresCol="features", labelCol="label")
```

Figure 5.3.2.1: Model logistic regression defined

b) Create pipeline

Then, we created a pipeline because several phases of the machine learning workflow, including data preprocessing, feature engineering, model training, and model evaluation, can be arranged and linked together more easily by using a pipeline. It guarantees that each of these actions is carried out in a reliable and consistent manner.

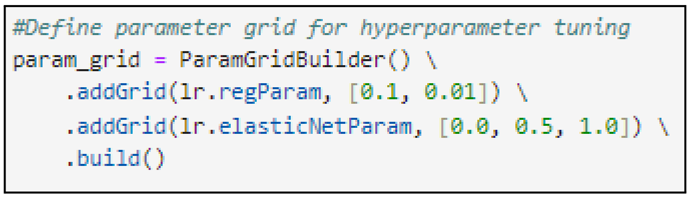


```
#create pipeline
pipeline = Pipeline(stages=[lr])
```

Figure 5.3.2.2: Pipeline create for logistic regression

c) Define Parameter Grid

Hyperparameters are configuration options that impact the model's behaviour and performance but are not learned during training. We indicated the range of hyperparameters that ought to be investigated during the hyperparameter tuning procedure by creating a parameter grid. This aids in determining the set of hyperparameters that maximises the performance of the model. We defined a parameter grid for hyperparameter tuning in **.addGrid(lr.regParam, [0.1, 0.01]) \ .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \.**

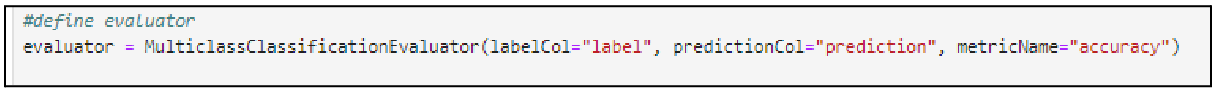


```
#Define parameter grid for hyperparameter tuning
param_grid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.1, 0.01]) \
    .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) \
    .build()
```

Figure 5.3.2.3: hyperparameter tuning parameter grid defined

d) Define Evaluator

Then, we defined the evaluator with a MulticlassClassificationEvaluator. Assessing how effectively the model generalises to previously encountered data requires evaluating its performance. The particulars of the issue and the intended result will determine which evaluation metric is used. In this instance, the percentage of correctly identified cases is gauged using accuracy as the statistic.



```
#define evaluator
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
```

Figure 5.3.2.4: Definition of evaluator

e) Define Cross-Validation

Cross-validation is a technique used to assess the performance of a model and to mitigate the risk of overfitting. We are able to acquire a more accurate assessment of the model's performance on unseen data by dividing the training data into many subsets, or folds, then training and assessing the model iteratively on various combinations of these subsets.

```
#Define cross-validation
crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=param_grid,
                          evaluator=evaluator,
                          numFolds=3)
```

Figure 5.3.2.5: cross validation definition

f) Run Cross-validation

Cross-validation compares the performance of several models trained on various subsets of the data to assist choose the optimal model configuration. This aids in determining the model that is less susceptible to random fluctuations in the training set and that generalises well to new data. After defining the model, we fit the cross-validation model with training data.

```
#Run cross-validation
cv_model = crossval.fit(final_train_data)
```

Figure 5.3.2.6: run cross validation with fit final train data into model

```
#Run cross-validation - lr
cv_model = crossval.fit(final_train_data_eva)
```

Figure 5.3.2.7: run cross validation with modified train data into model

g) Get Best Model

The model configuration that performs the best is determined by the selected evaluation metric which is the one that is chosen for additional testing on test data following cross-validation.

```
#get best model
best_model = cv_model.bestModel
```

Figure 5.3.2.7: Get the best model

h) Prediction

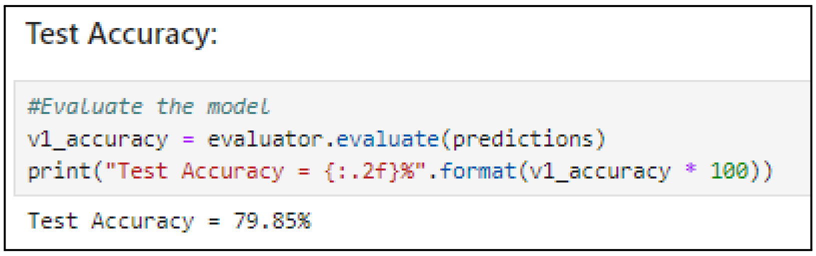
The best model is then used to make predictions on the test data to assess its performance in real-world scenarios which use the trained model to predict with the test dataset. In the evaluation we use the modified train data and prediction with the same test data.

```
# prediction
predictions = best_model.transform(final_test_data)
```

Figure 5.3.2.8: Model transform process with test data

i) Evaluate Model

Finally, the performance of the selected model is evaluated on the test data using the chosen evaluation metric to determine how well it performs in practice. In the first round of modelling and prediction, the model had a test accuracy of 79.85%. In the second round of modelling and prediction, it had a test accuracy of 79.51%. In both cases, the model's accuracy was considered good. We also evaluated the prediction results in classification reports and confusion matrices for both cases.

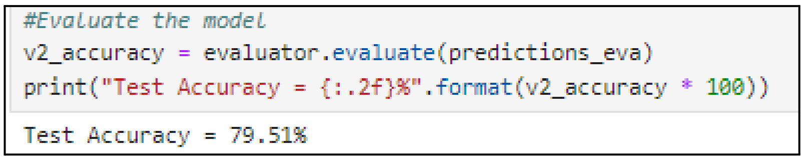


Test Accuracy:

```
#Evaluate the model
v1_accuracy = evaluator.evaluate(predictions)
print("Test Accuracy = {:.2f}%".format(v1_accuracy * 100))

Test Accuracy = 79.85%
```

Figure 5.3.2.9: Model accuracy



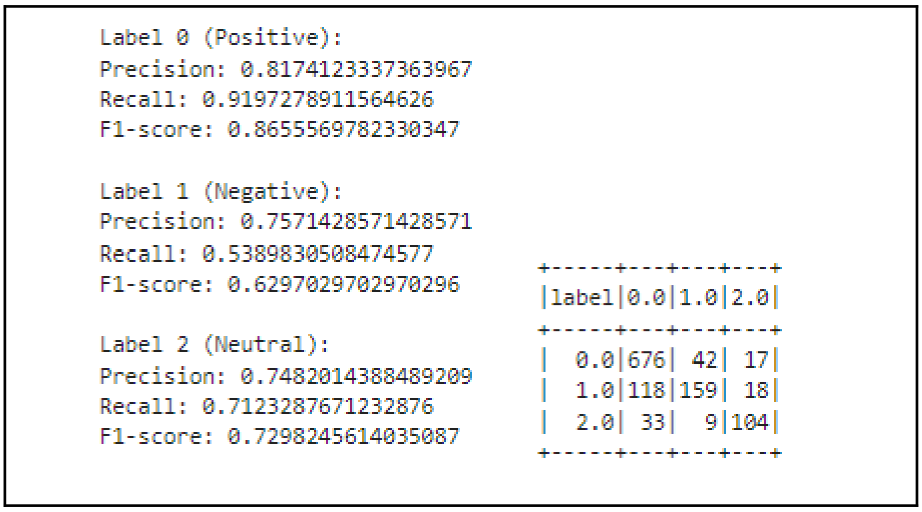
```
#Evaluate the model
v2_accuracy = evaluator.evaluate(predictions_eva)
print("Test Accuracy = {:.2f}%".format(v2_accuracy * 100))

Test Accuracy = 79.51%
```

Figure 5.3.2.10: Evaluation accuracy

j) Classification report

In the end, we use both prediction result and evaluation result to build classification reports for POSITIVE(0), NEGATIVE (1) and NEUTRAL (2) cases using precision, Recall and F1 score. We also have confusion matrices to understand more about the model.



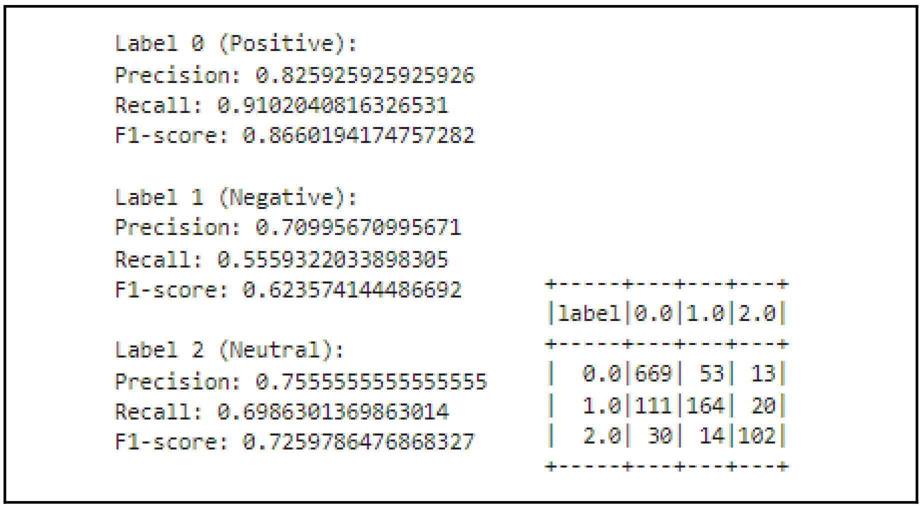
```
Label 0 (Positive):
Precision: 0.8174123337363967
Recall: 0.9197278911564626
F1-score: 0.8655569782330347

Label 1 (Negative):
Precision: 0.7571428571428571
Recall: 0.5389830508474577
F1-score: 0.6297029702970296

Label 2 (Neutral):
Precision: 0.7482014388489209
Recall: 0.7123287671232876
F1-score: 0.7298245614035087
```

```
+-----+---+---+---+
|label|0.0|1.0|2.0|
+-----+---+---+---+
|  0.0|676| 42| 17|
|  1.0|118|159| 18|
|  2.0| 33|  9|104|
+-----+---+---+---+
```

Figure 5.3.2.11: Classification report and confusion matrix of Model (Round 1).



```
Label 0 (Positive):
Precision: 0.825925925925926
Recall: 0.9102040816326531
F1-score: 0.8660194174757282

Label 1 (Negative):
Precision: 0.70995670995671
Recall: 0.5559322033898305
F1-score: 0.623574144486692

Label 2 (Neutral):
Precision: 0.7555555555555555
Recall: 0.6986301369863014
F1-score: 0.7259786476868327
```

```
+-----+---+---+---+
|label|0.0|1.0|2.0|
+-----+---+---+---+
|  0.0|669| 53| 13|
|  1.0|111|164| 20|
|  2.0| 30| 14|102|
+-----+---+---+---+
```

Figure 5.3.2.12: Classification report and confusion matrix of evaluation Model (Round 2).

5.3.3. <u>Results (Visualisation & Analysis)</u>

<u>Experiment 1 VS Experiment 2</u>

<u>1) Accuracy Performance Difference</u>

Referring to Figures 5.3.3.1, 5.3.3.2 and 5.3.3.3, the accuracy of the Logistic Regression model increases as the volume of data increases. In Experiment 2, the accuracy was around 79.51% with 3855 records. In Experiment 1, the accuracy was around 79.85% with 4811 records.

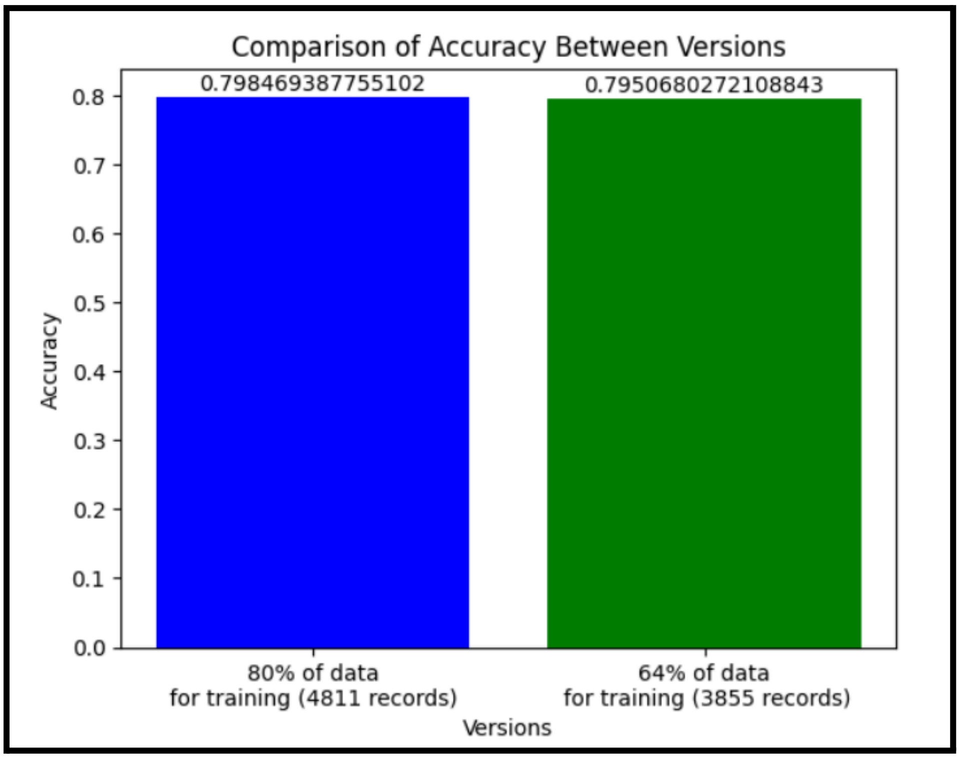
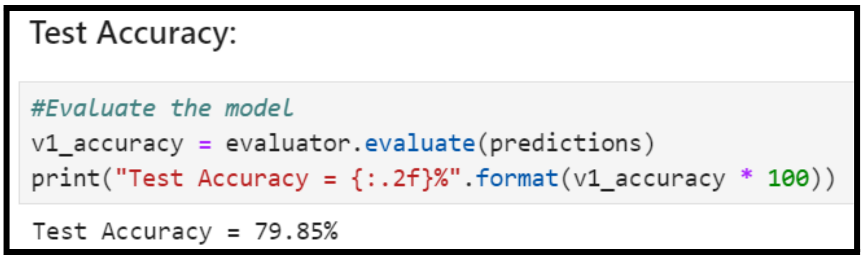


Figure 5.3.3.1 Visualisation Test Accuracy for Both Experiments

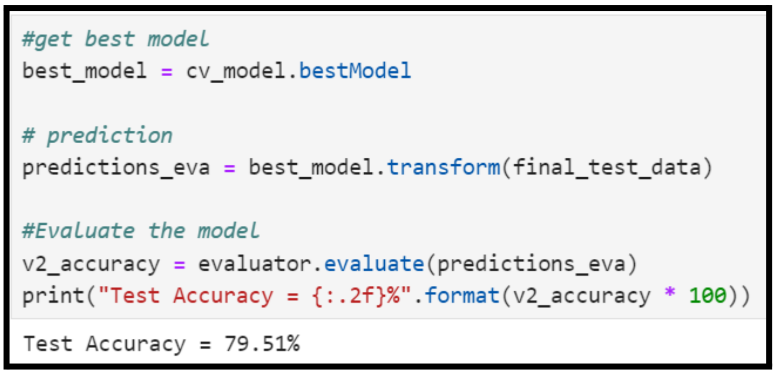


Test Accuracy:

```
#Evaluate the model
v1_accuracy = evaluator.evaluate(predictions)
print("Test Accuracy = {:.2f}%".format(v1_accuracy * 100))
```

```
Test Accuracy = 79.85%
```

Figure 5.3.3.2 Test Accuracy for Experiment 1



```
#get best model
best_model = cv_model.bestModel

# prediction
predictions_eva = best_model.transform(final_test_data)

#Evaluate the model
v2_accuracy = evaluator.evaluate(predictions_eva)
print("Test Accuracy = {:.2f}%".format(v2_accuracy * 100))
```

```
Test Accuracy = 79.51%
```

Figure 5.3.3.3 Test Accuracy for Experiment 2

<u>2) Comparing Precision, Recall & F1-Score for Positive Sentiment Class</u>

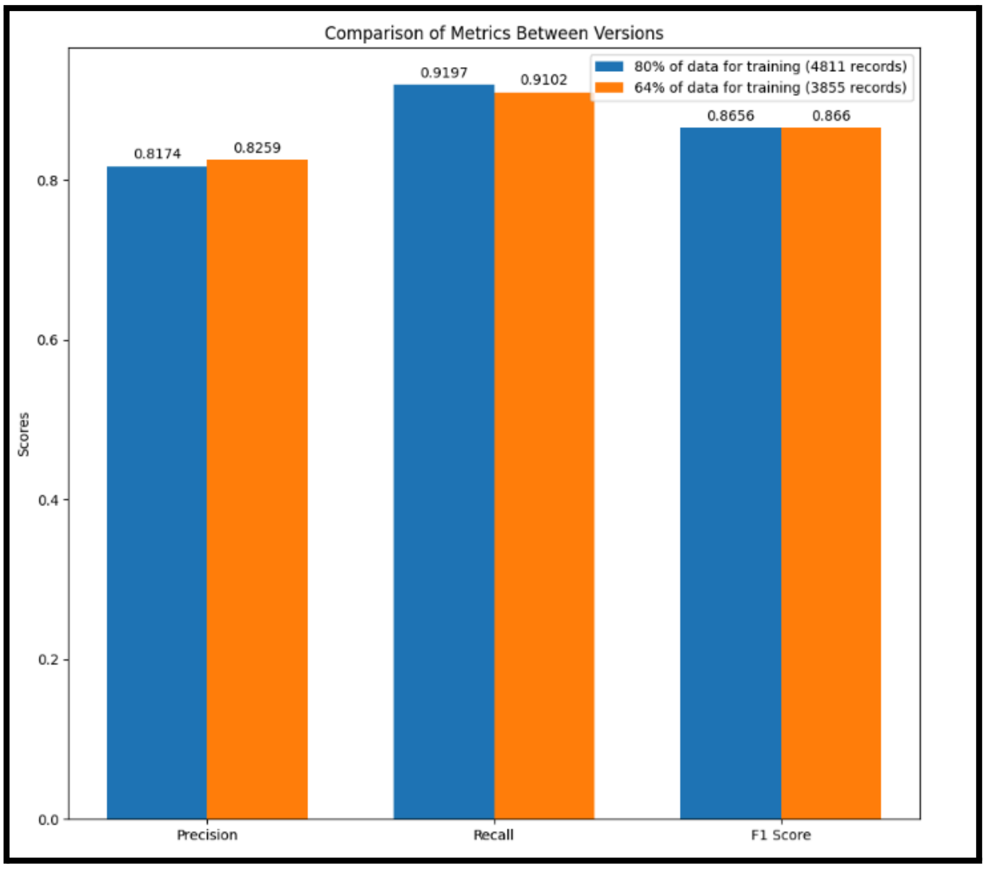


Figure 5.3.3.4 Precision, Recall & F1-Score for Positive Sentiment

Referring to Figure 5.3.3.4, it can be seen that Experiment 1 resulted in lower precision and f1-score when compared to Experiment 2. However, Experiment 1 achieved higher recall than Experiment 2.

3) Comparing Precision, Recall & F1-Score for Negative Sentiment Class

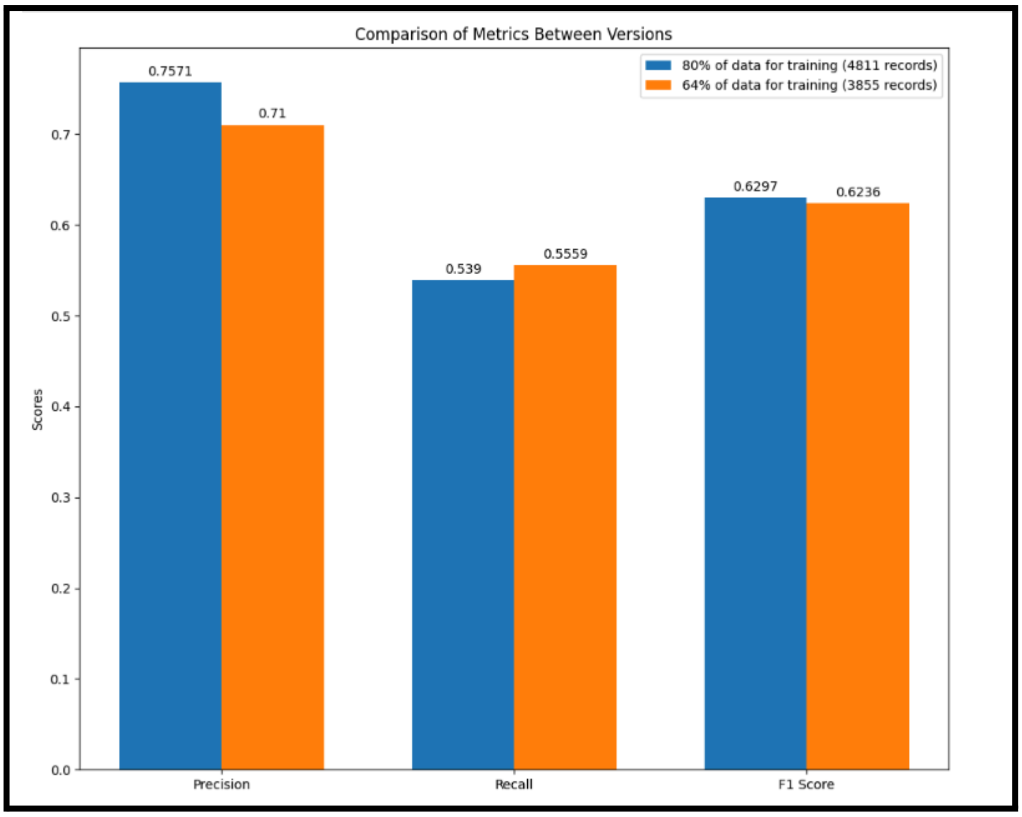


Figure 5.3.3.5 Precision, Recall & F1-Score for Negative Sentiment

Referring to Figure 5.3.3.5, it can be seen that Experiment 1 resulted in higher precision and f1-score when compared to Experiment 2. However, Experiment 2 achieved higher recall than Experiment 1.

<u>4) Comparing Precision, Recall & F1-Score for Neutral Sentiment Class</u>

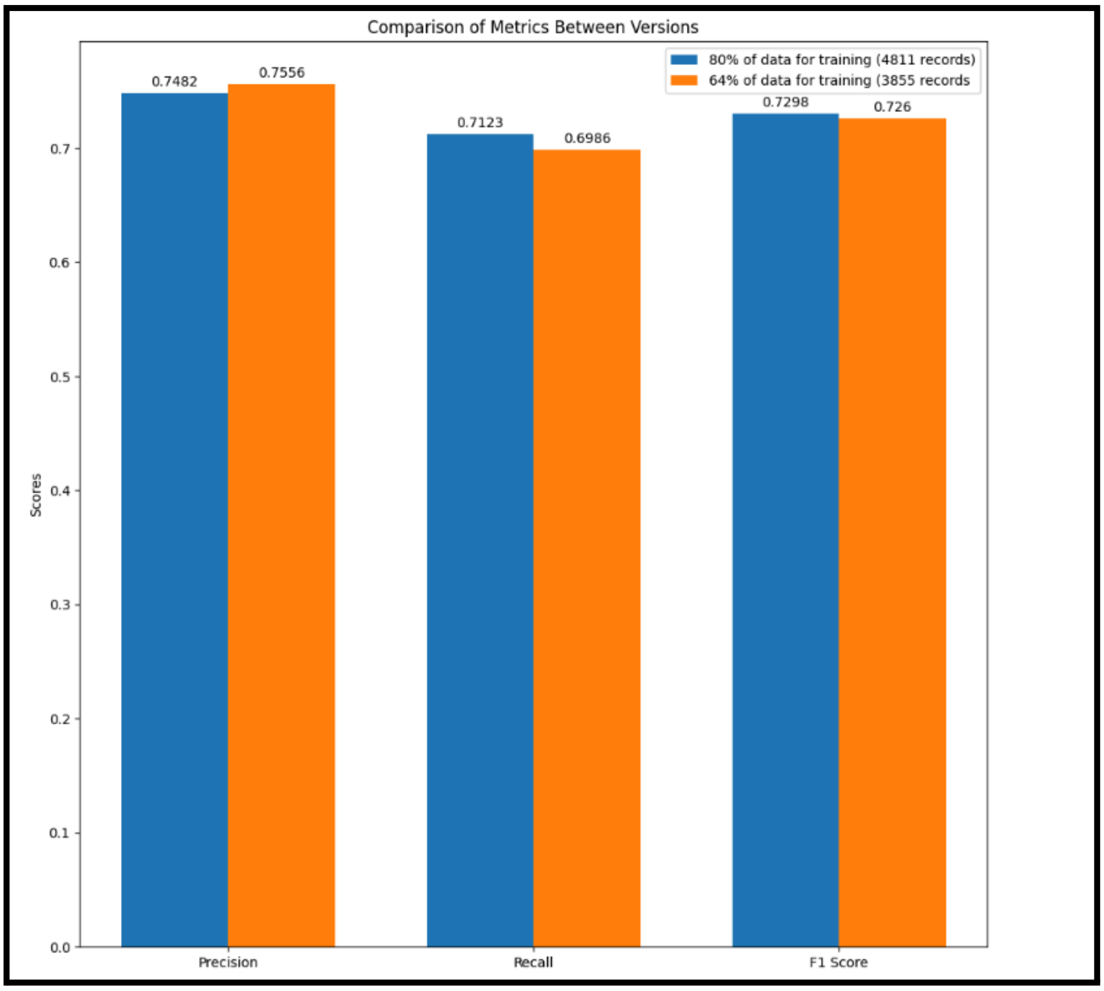


Figure 5.3.3.6 Precision, Recall & F1-Score for Neutral Sentiment

Referring to Figure 5.3.3.6, it can be seen that Experiment 1 resulted in higher recall and f1-score when compared to Experiment 2. However, Experiment 2 achieved higher precision than Experiment 1.

5) Comparing Count for Each Sentiment Category using Bar Chart

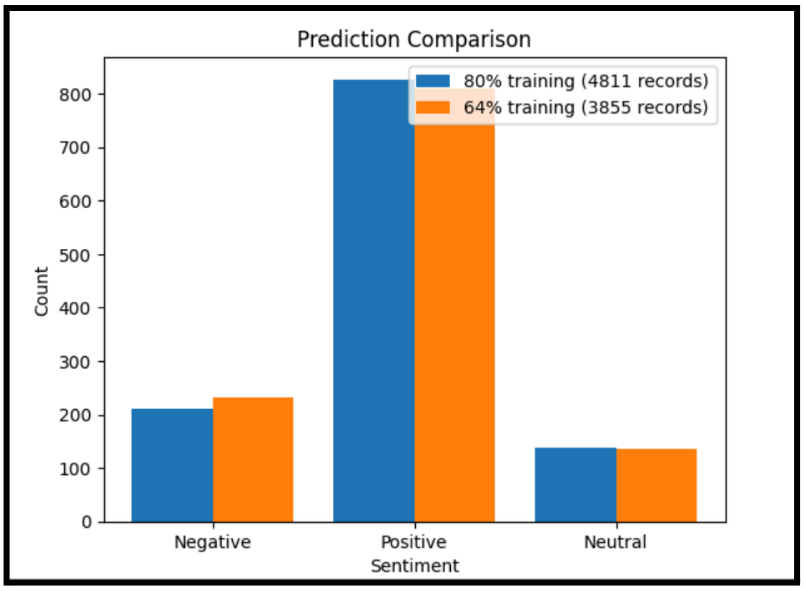


Figure 5.3.3.7 Count for Each Sentiment Category (Side by Side version)

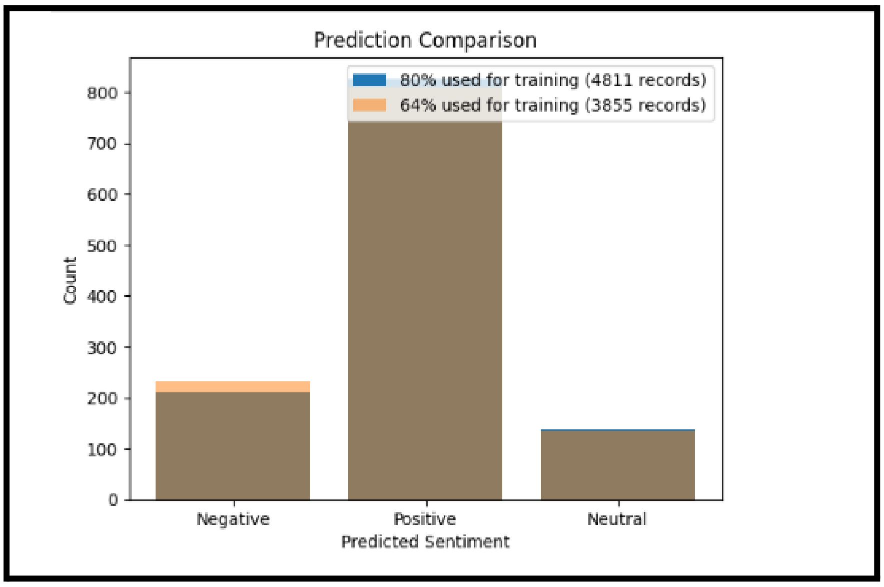


Figure 5.3.3.8 Count for Each Sentiment Category (Overlapping version)

Referring to Figures 5.3.3.7 and 5.3.3.8, the bar charts above show the count for each sentiment category between the training using 80% of the data and the the training using 64% of the data. Based on the bar charts, we can see that the count for the negative sentiment of "the training using 64% of the data" is more than the "training using 80% of the data". It suggests that the model trained with 64% of the data may be biased towards predicting negative sentiments. After that, we can

✓

see that the count for the positive sentiment for “training using 80% of the data” is more than the “training using 64% of the data”. It suggests that the smaller training dataset might lead to a less biased model towards positive sentiments. Lastly, you can see that the counts for neutral sentiment for both experiments are almost the same. It suggests that the model's prediction of neutrality remains consistent regardless of the size change of the training dataset, indicating stability in handling neutral sentiment across different datasets.

6) Comparing Proportion for Each Sentiment Category using Pie Chart

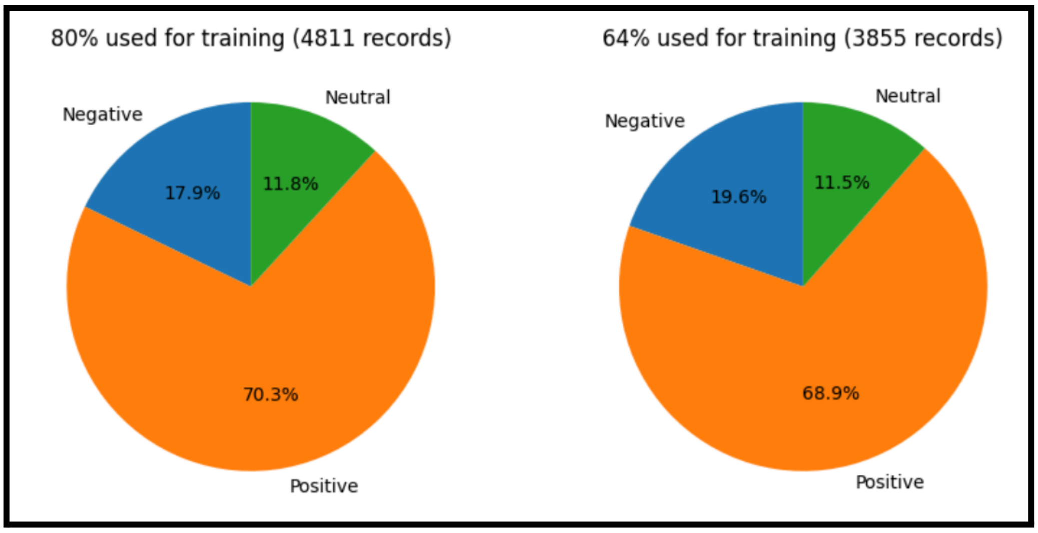


Figure 5.3.3.9 Comparing Proportion for Each Sentiment Category

The pie chart shows the percentage of the sentiment between “training using 80% of the data” and “training using 64% of the data”. In the training using 80% of the data, 17.9% of the predictions were negative, 70.3% were positive, and 11.8% were neutral. In the training using 64% of the data, 19.6% were negative, 68.9% were positive, and 11.5% were neutral. It suggests that the proportions of neutral sentiment are similar. After that, we can see that the training using 64% of the data has more negative sentiments and the training using 80% of the data has more positive sentiments.

Overall findings: Based on the results for performance metrics (accuracy, precision, recall, f1-score), the model’s performance improves as the volume of data increases.

✓

6. Conclusion

Our sentiment analysis program represents a significant advancement in understanding player feedback across various platforms, including YouTube, Reddit, and Kaggle, enabling Activision to make informed decisions to enhance Call of Duty: Mobile (CODM). By predicting the sentiment of user comments as positive, neutral, or negative, we provide valuable insights into player sentiments, aiding Activision in tailoring updates that resonate with player demands, thus fostering a more enjoyable and engaging gaming experience.

For **data collection,** we obtained data from **YouTube** through web scraping, **Reddit** via data crawling and **Kaggle** for CODM Google Play Store reviews dataset. The raw data we extracted was then stored in **MongDB** and **HDFS**. The three datasets were then processed and **transformed** using a range of techniques such as tokenization, lemmatization, and feature engineering to calculate sentiment scores. They were then stored in **Neo4J AuraDB**.

As for the training and testing datasets, they were stored in **HiveDB**. Since we decided to compare model performance by changing the volume of data, we created 2 versions of the training dataset in order to carry out 2 experiments. One training dataset was created using 80% of the data while the other one was created using 64% of the data. As for the testing dataset, it was created using 20% of the data and it remained the same in both experiments. To effectively predict the sentiment categories of the comments, we used PySpark's **Logistic Regression** model, which resulted in 2 prediction results. These data were then stored as 2 separate tables in **HiveDB**. Overall, based on the results for performance metrics (accuracy, precision, recall, f1-score), the model's performance improves as the volume of data increases.

For streaming, we utilised 2 technologies, which were Structured Streaming and Kafka. Our **structured streaming** approach will allow for real-time monitoring of negative comments and sentiment proportions across positive, neutral and negative. Additionally, utilising **Kafka** streaming, we aim to highlight positive comments, thereby accentuating the game's strengths and fostering a positive community atmosphere.

By employing advanced analytics and leveraging cutting-edge technologies, our sentiment analysis program empowers Activision to maintain CODM's market leadership position for the first person shooting game category, driving continuous improvement and innovation in the ever-evolving landscape of mobile gaming.

✓

References

Ahmed, H. M., Awan, M. J., Khan, N. S., Yasin, A., & Shehzad, H. M. F. (2021). Sentiment Analysis of Online Food Reviews using Big Data Analytics. Ilkogretim Online - Elementary Education Online, 20(2), 827–836. https://www.researchgate.net/publication/350836421_Sentiment_Analysis_of_Online_Food_Reviews_using_Big_Data_Analytics

Fadheli, A. (2022, May). How to extract YouTube comments in Python - the python code. Python Code. https://thepythoncode.com/article/extract-youtube-comments-in-python

Fandom. (n.d.). Call of duty: Mobile. Call of Duty Wiki. https://callofduty.fandom.com/wiki/Call_of_Duty:_Mobile#cite_note-7

Gameopedia. (2023, January 16). Sentiment analysis for games: Opinion mining. https://www.gameopedia.com/our-services/sentiment-analysis-for-games/

IBM. (n.d.). What is sentiment analysis? https://www.ibm.com/topics/sentiment-analysis

Kılınç, D. (2019). A spark‐based Big Data Analysis Framework for real‐time sentiment prediction on streaming data. Software: Practice and Experience, 49(9), 1352–1364. https://doi.org/10.1002/spe.2724

Neo4j. (n.d.). What is a graph database? - getting started. Neo4j Graph Data Platform. https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/

Prakash, T. N., & Aloysius, A. (2019). DATA PREPROCESSING IN SENTIMENT ANALYSIS USING TWITTER DATA. INTERNATIONAL EDUCATIONAL APPLIED RESEARCH JOURNAL (IEARJ), 3(7), 89–92. https://www.researchgate.net/publication/334670363_DATA_PREPROCESSING_IN_SENTIMENT_ANALYSIS_USING_TWITTER_DATA

Saniya. (2023, October 20). YouTube is a video-sharing platform that allows users to upload, watch, and interact with videos. Medium.https://medium.com/@saniya15383/youtube-is-a-video-sharing-platform-that-allows-users-to-upload-watch-and-interact-with-videos-bd0bd27680b4

ZenRows. (2023, October 14). Dynamic web pages scraping with Python: Guide to scrape all content - zenrows. https://www.zenrows.com/blog/dy

✓