Ringier Technical Assignment Akshay Maharaj

Final Report

**Introduction**

Access to raw, processed, and analyzed data is vital for data engineers and analysts. Raw data provides the foundation for understanding information, processed data is structured for analysis, and analyzed data offers insights. This access facilitates efficient data preprocessing, analysis, and decision-making, ensuring data quality, compliance, and performance optimization. It also enables iterative analysis and improvement, driving value and innovation within organizations.

**Project Objective**

The project objective is to extract data from the NewsAPI platform and transform it into three distinct stages: raw, processed, and analyzed data. The raw data will be obtained directly from NewsAPI, preserving its original format. The processed data will involve cleaning, structuring, and storing the information in a database for efficient analysis. Finally, the analyzed data will undergo natural language processing (NLP) techniques to extract insights, sentiment analysis, and topic modeling, providing valuable insights for decision-making and further exploration.

**Resources**

**Language**

The language chosen to code the solution in was Python. Multiple languages were considered for use with the most likely alternative being Java, with access to the Stanford CoreNLP Library, however Python proved to be more suitable for the applications required. API’s required were easily imported and data science resources were more readily available for reference and tutorials.

**IDE**

PyCharm is the optimal choice for this project due to its robust features tailored for data engineering and analysis tasks. With PyCharm, developers have access to powerful tools for writing and debugging code, managing dependencies, and integrating with popular libraries like pandas, scikit-learn, and NLTK. Its intuitive interface and built-in version control support streamline the development process, ensuring efficient collaboration and project management. Additionally, PyCharm's compatibility with various databases, including MySQL, facilitates seamless interaction with data storage and retrieval systems, making it an ideal environment for working with raw, processed, and analyzed data. Jupyter Notebook was the default IDE that I looked to however I encountered multiple issues with imports and compatibility resulting in the decision to revert to Pycharm.

**API**

**Requests** - offers a simplified and intuitive interface for making HTTP requests, facilitating seamless interaction with web services and APIs.

**Urllib3** - Python library for making HTTP requests, offering enhanced features for reliability and security in web scraping and API interactions.

**MySQL** - Provides tools for connecting to MySQL databases and executing SQL queries, facilitating interaction with MySQL databases in Python applications.

**VaderSentiment** - Tool for performing sentiment analysis on text data using the VADER (Valence Aware Dictionary and Sentiment Reasoner) lexicon, allowing for easy determination of sentiment polarity in textual content.

**SQLAlchemy** - Toolkit for working with SQL databases, providing a high-level interface for database interactions and supporting various database engines.

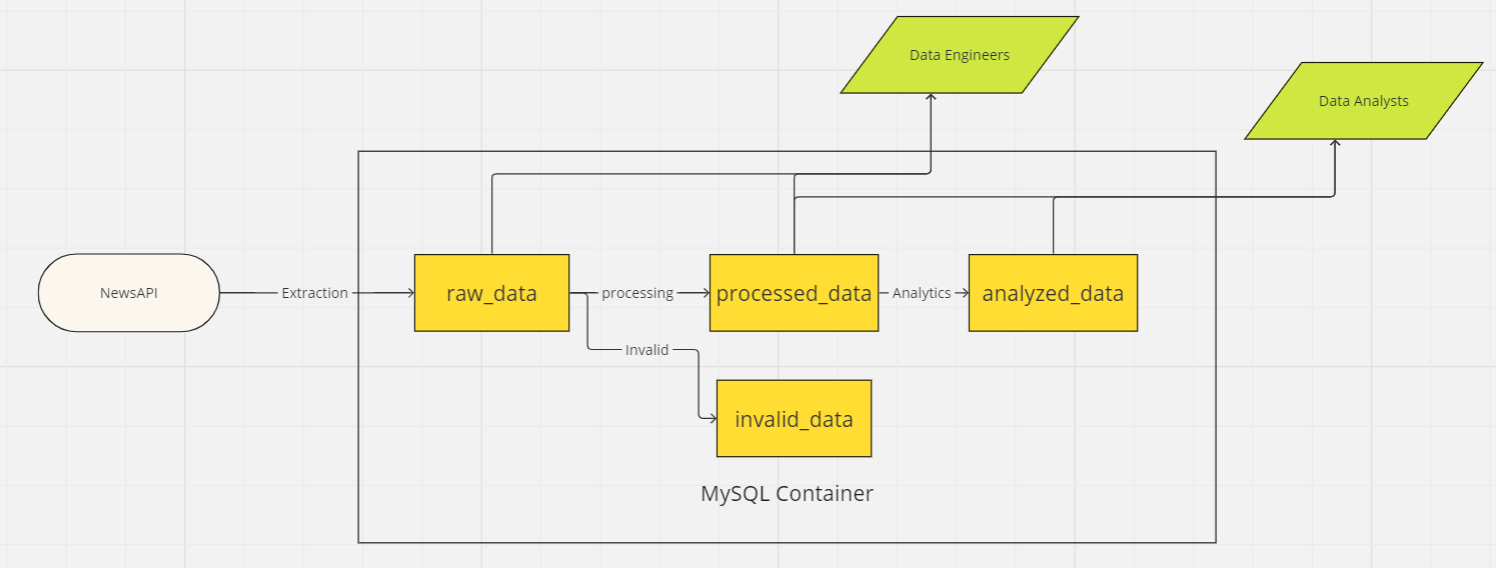
**Pandas**- pandas is an open-source library built on top of numPy providing high-performance, easy-to-use data structures and data analysis tools.

**SciKit** – SciKit contains the sklearn library which encapsulates a lot of efficient tools for machine learning and statistical modeling including classification, regression, and clustering.

**NLTK** - tools for natural language processing tasks, including tokenization, stemming, tagging, and parsing.

**Datasets**

The NewsAPI Article dataset consists of news articles in Europe over the past week.

**Process Workflow**  
  


**MySQL Container and Database Setup**  
  
Using Docker to create a MySQL container offers a portable and scalable solution for managing databases, allowing for easy deployment, isolation, and replication of database environments across different systems or environments.

Container and DB setup process:

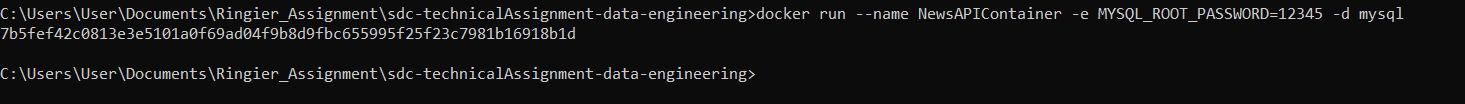
* Install Docker
* CMD: Docker pull MYSQL

A computer screen with numbers and letters

Description automatically generated

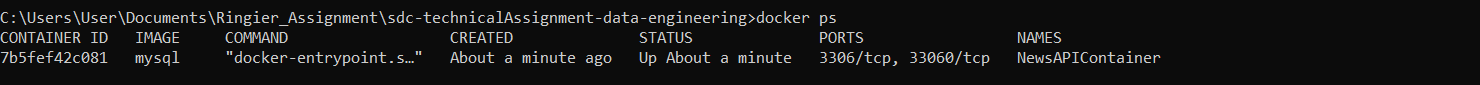
* CMD: (Instantiate MySQL Container)

docker run --name NewsAPIContainer -e MYSQL\_ROOT\_PASSWORD=12345 -d mysql



* CMD: (View Docker Container Status)

docker ps



A screenshot of a computer

Description automatically generated

* CMD: (Access Docker MySql Shell Container to create Database)

docker exec -it NewsAPIContainer mysql -uroot -pA black screen with white text

Description automatically generated

* CMD: (Create Database NewsAPI)

CREATE DATABASE NewsAPI;

A black screen with white text

Description automatically generated

We now have an active container and database within container setup.

A screenshot of a computer

Description automatically generated

**Variable and Import Initialization**

Various modules needed to perform code extraction, transformation and loading to MySQL tables required

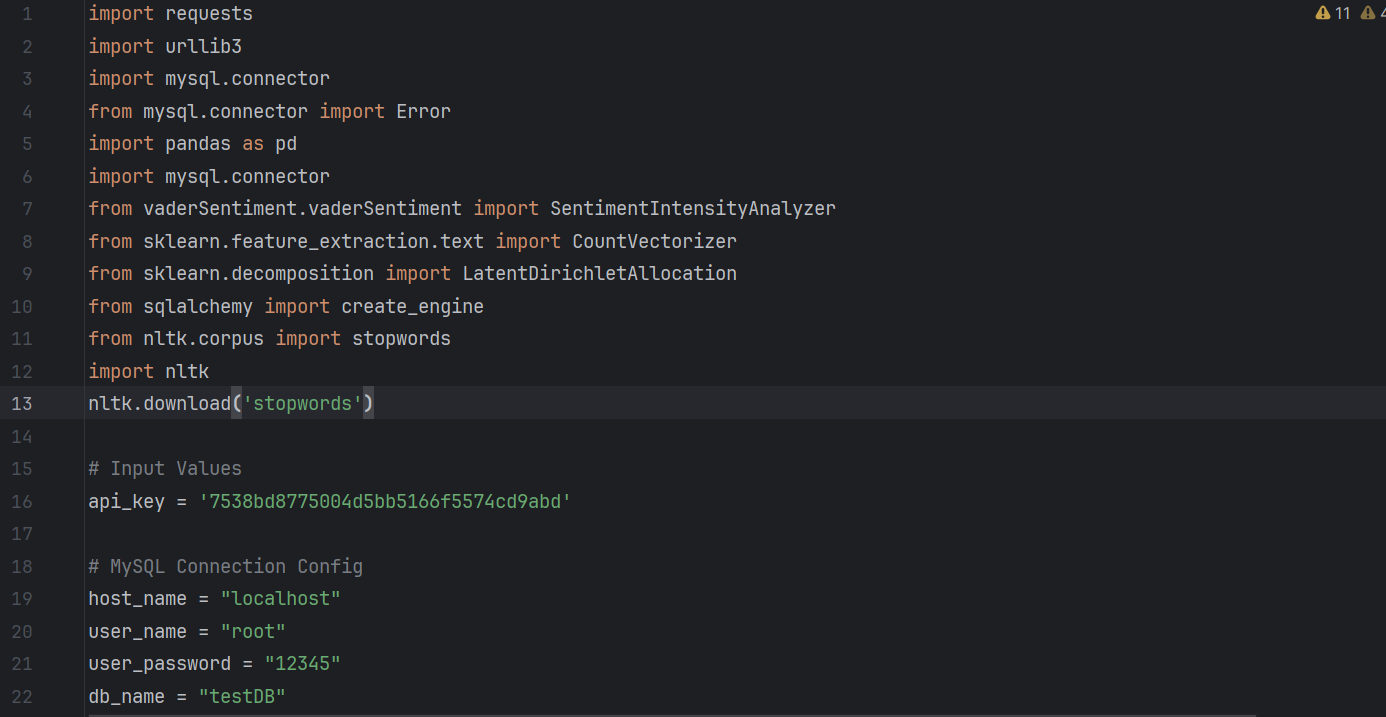


Figure 1.A – Modules and Input Values

**Data Extraction**

The required dataset resides withing **NewsAPI** (ww.newsapi.org/). Data can be restireved be defining a function that makes requests to the specified webpage to extract information of a certain filter criteria. In this case we were looking to extract all news articles all newspaper articles released in the last week across Europe.

A screen shot of a computer program

Description automatically generated  
By defining a function **fetch\_news()** and inputting requirements (Generated API\_Key, country , from\_date , to\_date) we can use the requests library to fetch the news articles and store them in an array. Error handling code included to debug previous fetch failures.

**Define Database Connection**

We need to setup a connection to our desired database so that we may load data to relevant tables. To do so we create a **create\_connection()** function which allows us to input our Database connection details(#MySQL Connection Config details mentioned in Figure1.A ) and instantiate a connection.

  
  
**Define SQL Query Function**

We define an **execute\_query()** function that allows us to run a specified query using the connection we setup to our Database  
  
A screen shot of a computer

Description automatically generated

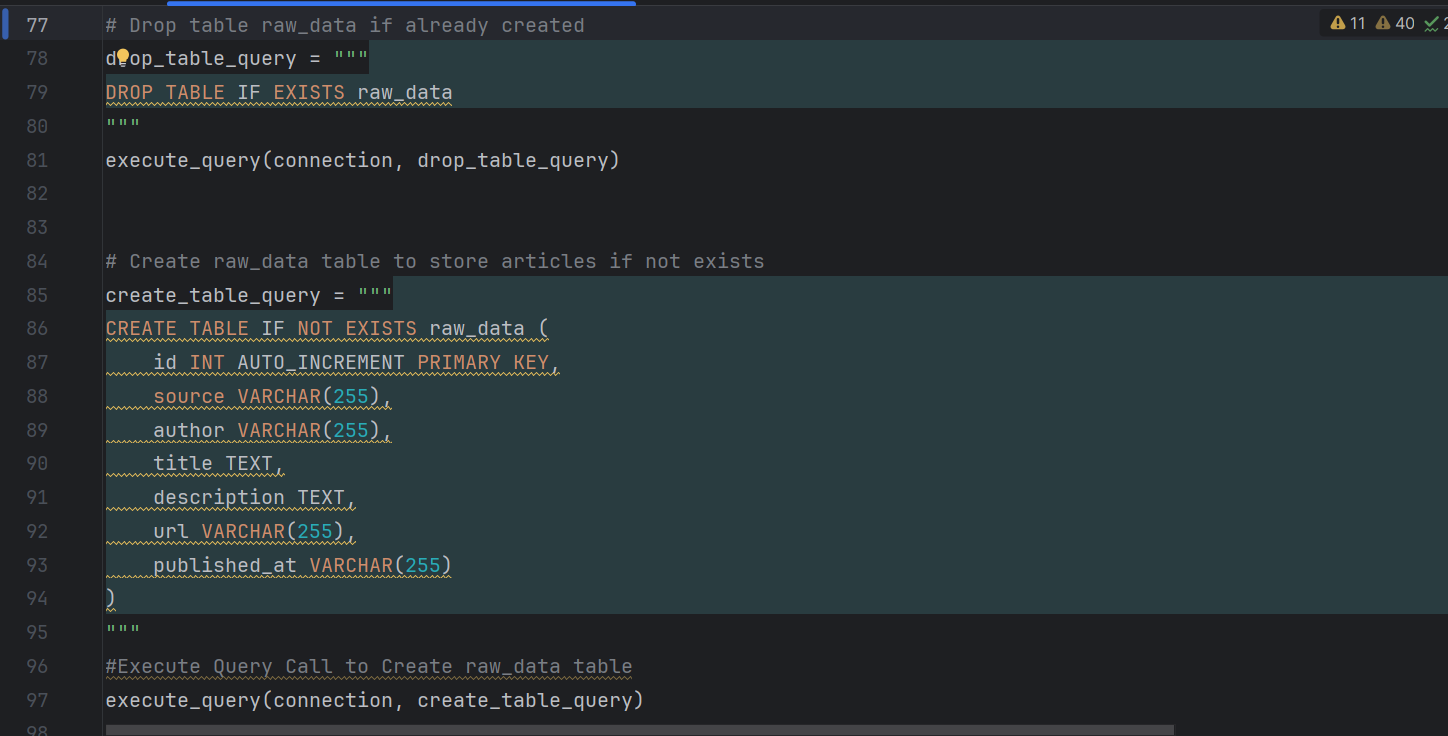
**Extraction Continued**

We continue with our data extraction but first setup our environment. We create variables “articles” and “connection’ to perform our NewsAPI extraction and instantiate a connection to the database we created in our MySQL Container.

A computer screen with text

Description automatically generated

We then proceed to setup tables that can hold our extracted data. The first table we setup is called the **“raw\_data”** table, this table shall hold the raw data we’ve extracted from NewsAPI. We create query variables which we execute using execute\_query to perform required tasks.



We then cycle through our articles columns, partitioning our data into its respective columns and inserting them into our raw\_data table.  
  


Results:

A screenshot of a computer program

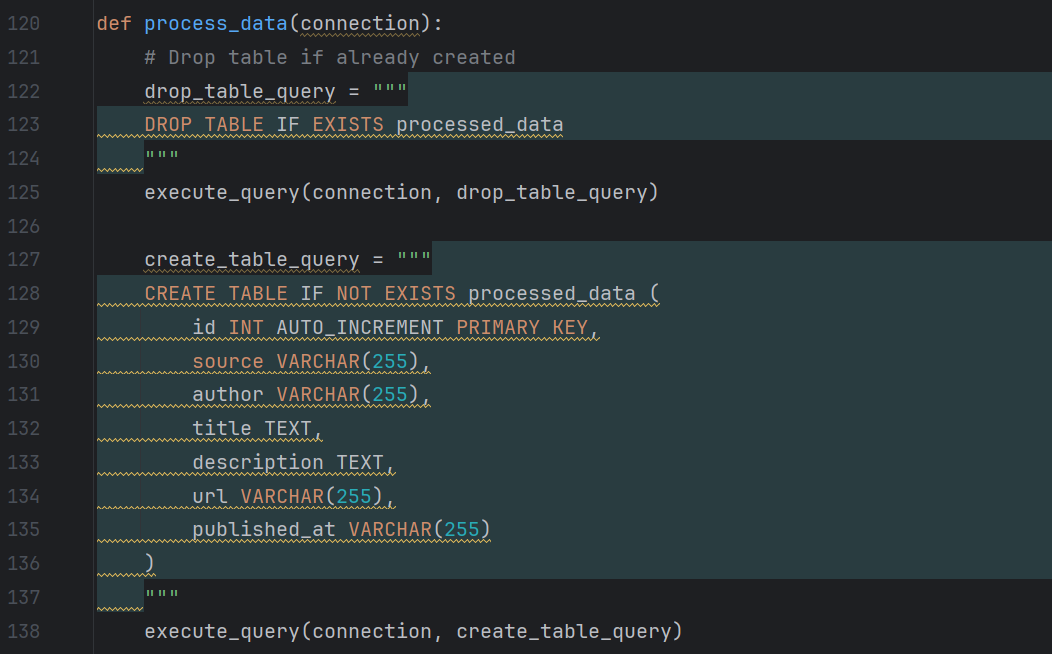
Description automatically generated

**Data Processing**

Creating a **“processed\_data”** table from raw\_data is crucial for transforming and organizing raw information into a structured format that is optimized for analysis. This process enhances data quality, improves analysis efficiency, and ensures the reliability and reproducibility of insights derived from the data. Any invalidated data to be captured in an invalid table which is important for reporting and governance.

Criteria: WHERE description != '' AND description != '[Removed]'  
Criteria selection is remedial but sufficient enough to display the process.

processed\_data creation:



A computer screen shot of a program

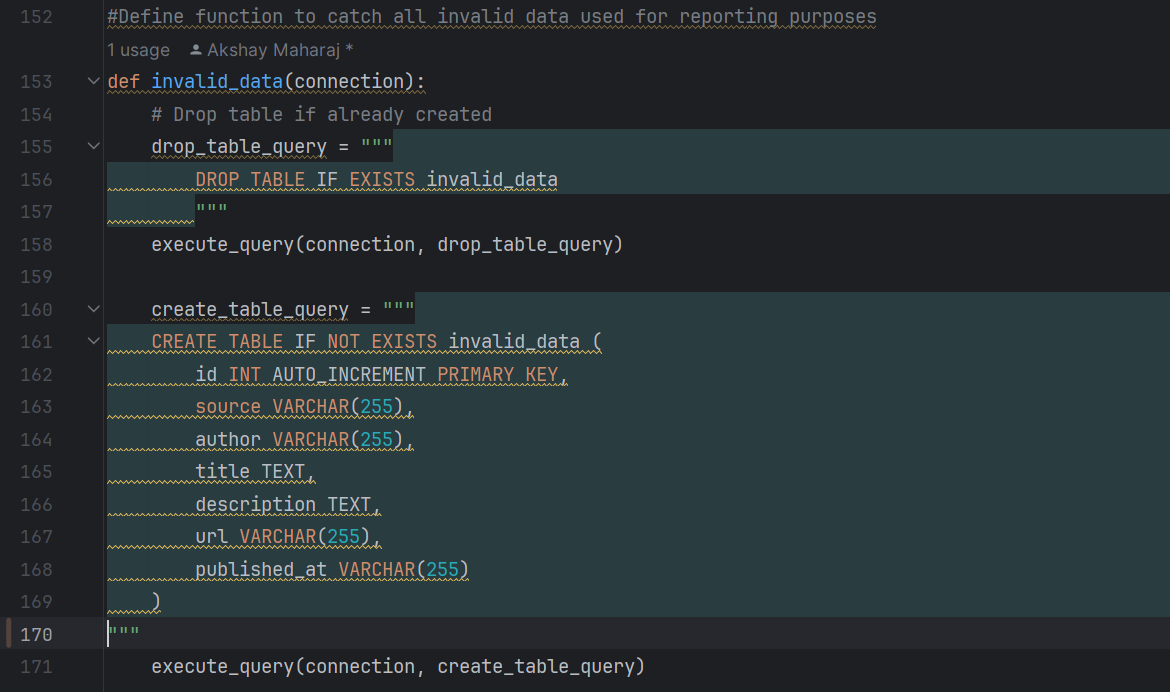
Description automatically generated

**Results:**

**A screenshot of a computer

Description automatically generated**

**invalid\_data creation**



A screenshot of a computer program

Description automatically generated

**Results:**

A screenshot of a computer

Description automatically generated

**NLP Techniques**

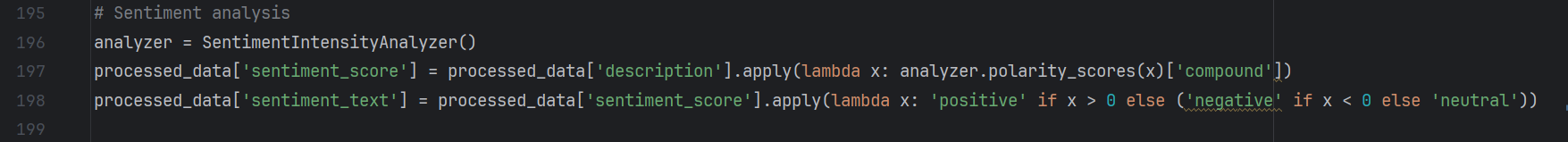
Creating an **“analyzed\_data”** table from the processed\_data table is important because the processed\_data table has already been cleaned, transformed, and structured for analysis. This saves time, ensures data integrity, and facilitates reproducibility of results. Additionally, it minimizes privacy and security risks associated with exposing raw data.

We proceed to perform Natural Language Techniques on our dataset so that we may analyse the sentiment of each article, give it a rating as well as define which topics the data relate to.

To do so we first define a **SQLAlchemy “engine”** which connects to our MySQL Database. This allows us to utilize the Pandas Module SQL functions with MySQL as a base instead.

A screen shot of a computer code

Description automatically generated  
  
By utilizing the **SentimentIntensityAnalyzer()** function and **polarity\_scores()** function in the **vaderSentiment** module, we’re able to perform rudimentary sentiment analysis on the processed\_data dataset.

We create two new columns: sentiment\_score and sentiment\_text that shall hold a score of the sentiment of the description and give a sentiment text description.  
  


**Topic Modelling**

We attempt to assign a topics\_covered value to each record which helps to group our records into similar topics. By utilizing our **sklearn** and **nltk** modules we’re able to tokenize and train our model to output a topics\_covered numerical value which is assigned to the dataset.

A screen shot of a computer program

Description automatically generated

this code segment applies the trained LDA model to the document-term matrix, identifies the dominant topic for each document, and then adds this information as a new column in the **processed\_data** dataframe.

**stopwords** is a list from the NLTK library for the English language. Stopwords are common words such as "the", "is", "and", etc., which are often filtered out during text preprocessing as they usually do not carry significant meaning.

The **CountVectorizer** is initialized. This is a part of the scikit-learn library and is used for converting text documents into a matrix of token counts. It tokenizes the text and builds a vocabulary of known words. Here, **stop\_words** parameter is passed to remove the stopwords during tokenization.

The **fit\_transform()** method of the **CountVectorizer** is applied to the description column of processed\_data. This converts the text data into a document-term matrix, where rows represent descriptions and columns represent individual words in the vocabulary. Each cell contains the count of how many times each word appears in each document.

**Latent Dirichlet Allocation (LDA)** is a popular technique for topic modelling, which discovers abstract topics within a collection of documents based on the distribution of words. The LDA model is initialized with **n\_components=8**, which specifies the number of topics to be extracted. The model is then trained on the document-term matrix **X** obtained from the previous step using the **fit()** method.

**lda\_model.transform(X)** applies the trained **lda\_model** to the document-term matrix **X**, resulting in a new matrix where each row represents a document and each column represents a topic. Each cell contains the probability that the document belongs to the corresponding topic. This step calculates the topic distribution for each document. **argmax(axis=1)** identifies the dominant topic for each document based on the topic probabilities.

A screen shot of a computer

Description automatically generated

Finally, we create the analyze\_data table and close out our connection and engine.

**Shortcomings**

* The default behaviour of NewsAPI is to return up to 100 articles per request. This limit is imposed by the API provider for performance reasons and to prevent abuse of the service.

**Fetch\_news() function could be modified to run in in batches of page\_size (default 100) until it fetches all articles available within the specified date range.**

* **Implore more advanced text preprocessing and transformations to improve data quality.**
* Sentiment Analysis performed is rudimentary, more advanced NLP techniques may be implemented to provide a more conclusive results.

**Code alterations as well as alternate NLP Libraries could be used such as Transformers or Flair to perform more complex sentiment analysis.**

* The dataset that we used to train our Topic Model is the same dataset that we used to test our Topic Model.

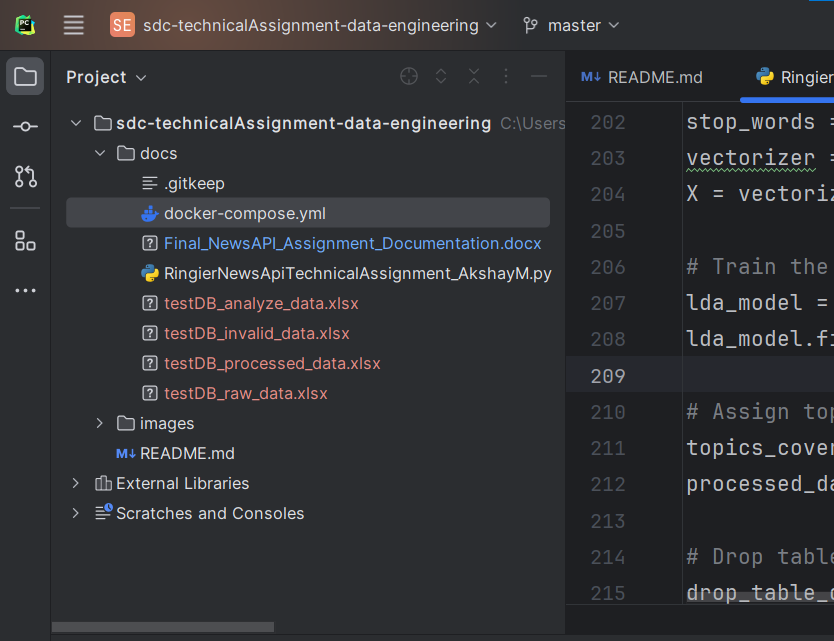
**In a typical machine learning workflow, you would split your data into separate training and test sets to evaluate the model's performance on unseen data.**

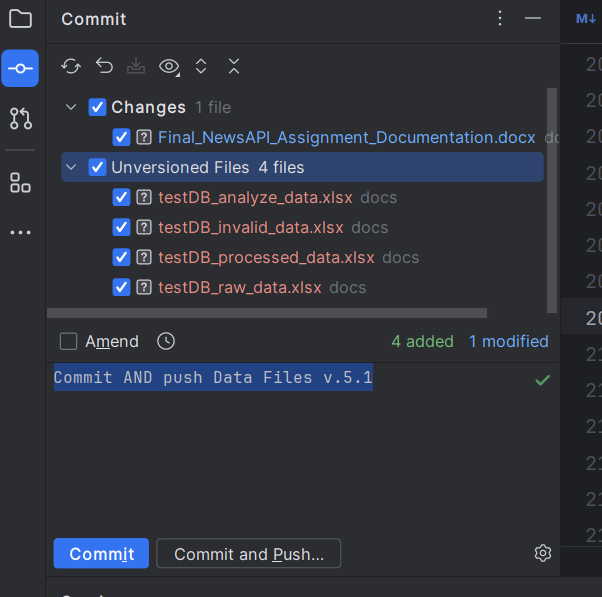
**Version Control**

Version Control has been implemented by saving our Pycharm project in our Local Repository and by connecting the project to our github remote repository. We are able to push and commit seamlessly.



Files that have not been committed or pushed:



Committing Files   


Committed to local Repository and pushed to remote repository

A screenshot of a computer

Description automatically generated

**Possible Next Steps**

(Other than fixing mentioned shortcomings):

* Implement additional NLP tasks such as machine translation, Part-of-Speech Tagging, Named Entity Recognition

**Conclusion**

Python's simplicity, versatility, and rich library ecosystem make it ideal for coding and Natural Language Processing (NLP) tasks. Docker's portability, scalability, and resource efficiency make it a powerful tool for containerizing data and applications, ensuring consistent deployment across diverse environments. Together, Python and Docker offer streamlined solutions for software development and data management.