Data Analytics

Exploring the Impact of Choice Award Books on Reader Stress Through NLP:

A Data Analysis Study

Xiaobo TENG

April,2024

Table of Content

Introduction 4

Data and data sources 5

Data Collection and Preparation 6

Collection Methods ... 6

Data Processing ... 10

Database Type Selection and Database Creation 15

Relational Database vs Non-Relational Database ... 15

MySQL Database Setup ... 16

MySQL Data Queries ... 18

API Development and Data Exposition via API 19

Flask API Development ... 19

API Endpoint Construction ... 19

Documentation ... 20

Exploratory Data Analysis (EDA) 21

Exploratory Data Analysis ... 21

Visualization ... 23

Development and Application of NLP Models 27

Model Development ... 27

Model Evaluation ... 28

Implementation....29

Advanced Analysis and Further Insights **30**

Advanced Analysis of Stress in Book Reviews....30

Concluding Insights....33

Conclusion **34**

Introduction

Business Case

The analysis conducted in this study aims to uncover the relationship between award-winning books and reader stress levels. By delving into the emotional impact of literature, publishers and authors can gain valuable insights to refine their strategies and enhance reader engagement.

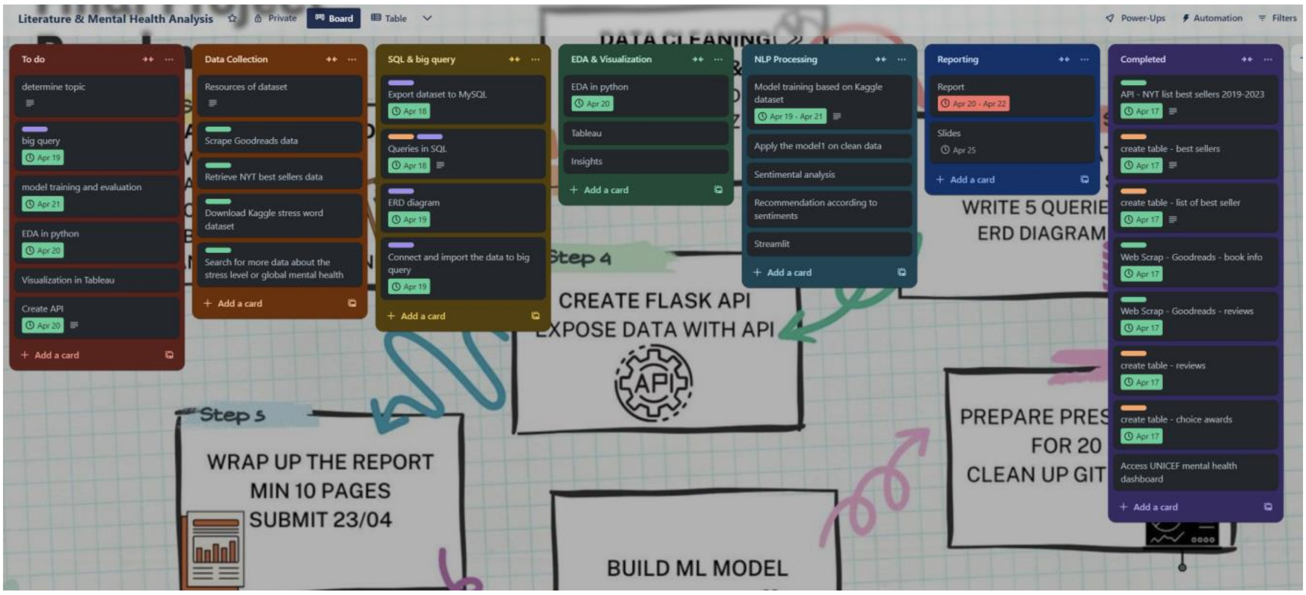
By leveraging insights from this data analysis study, publishers and authors can foster deeper emotional connections with readers and optimize their literary offerings to resonate more profoundly. Ultimately, understanding the interplay between award-winning books and reader stress levels can lead to more impactful storytelling and enhanced reader satisfaction.

Study Objectives

The main objective of this analysis is to investigate the relationship between award-winning books and reader stress levels. By examining the comments associated with these books and analyzing whether they contain stress, we aim to understand if there is any correlation between the literary quality of award-winning books and the emotional impact they have on readers.

Plan

During the execution of this project, Trello is used for project planning and management.



Data and data sources

Data:

Goodreads Choice Awards: Information on award-winning books from 2019 to 2023.

Book Comments: The top ten comments for each book, along with conclusions from a model indicating whether the text contains stress.

NYT BestSellers: All the bestsellers from NYT books from 2019 to 2023, gathered by NYT Books API

Stress Analysis in Social Media: The dataset downloaded from Kaggle, used for model training (Stress Analysis in Social Media)

Data sources:

Best Books 2023 — Goodreads Choice Awards

Best Books 2022 — Goodreads Choice Awards

Best Books 2021 — Goodreads Choice Awards

Best Books 2020 — Goodreads Choice Awards

Best Books 2019 — Goodreads Choice Awards

Books API | Dev Portal (nytimes.com)

Stress-Detection-with-Machine-Learning (kaggle.com)

GitHub repository (in progress):

https://github.com/txiao9331/LitMental.git

Data Collection and Preparation

Collection Methods

The primary data sources for analysis are web scraping and the NYT Books API. The former is used to gather all the winning books from the Goodreads Choice Awards from 2019 to 2023, including information about each book and the top ten reader reviews. The latter collects information on all bestsellers from 2019 to 2023. Additionally, open dataset from Kaggle about the stress analysis in social media is used for model training.

1. Web Scraping

I chose to use web scraping to collect book information from Goodreads because this method is more flexible, allowing me to selectively gather the content I need. To facilitate effective data collection, I defined several functions that incorporate error handling to manage situations where data cannot be retrieved, which ensures that the collection process is not interrupted by errors. These functions perform different tasks:

- fetching the URL for each award-winning book from a specified URL;
- extracting book details such as title, author, and genre from the book's URL;
- scraping the category of award and the number of books for each award;
- collecting the top ten reader reviews for each book using its URL.

The following illustration demonstrates examples of these functions and how error handling is implemented:

```python
def get_format(soup):
    try:
        pagesFormat = soup.find('p', attrs={'data-testid': 'pagesFormat'}).get_text()
        format_parts = pagesFormat.split()
        if len(format_parts) >= 3:
            return format_parts[2]  # Typically the third element is the format
        return "Format not specified"
    except AttributeError:
        return "Format not specified"
    except IndexError:
        return "Format not specified"

def get_publish_info(soup):
    try:
        pub_info = soup.find('p', attrs={'data-testid': 'publicationInfo'}).get_text()
        return pub_info.strip()  # Clean whitespace for consistency
    except AttributeError:
        return "Publication info not available"


def get_description(soup):
    try:
        description_element = soup.find('span', class_="Formatted")
        if description_element:
            return description_element.get_text().strip()  # Remove excess whitespace
        return "No description available"
    except AttributeError:
        return "No description available"
```

I separately collected information on the Choice Award-winning books from 2019 to 2023, creating an independent DataFrame for each year, which I then saved as CSV files. Below are the results of the data collection, using 2023 as an example:

```python
books_2023_df.head(3)
```

	title	ISBN	author	genre	rating	number_of_ratings	number_of_reviews	pages	format	publish info	description	Votes
0	Yellowface	9780063250833	R.F. Kuang	Fiction, Contemporary, Audiobook, Literary Fic...	3.80	434,299	59,749	336	Hardcover	First published May 25, 2023	Alternate cover edition of ISBN 9780063250833....	200722
1	Hello Beautiful	ISBN not found	Ann Napolitano	Fiction, Historical Fiction, Audiobook, Contem...	4.18	300,811	28,486	416	Kindle	First published March 14, 2023	An emotionally layered and engrossing story of...	60171
2	The Wishing Game	ISBN not found	Meg Shaffer	Fiction, Fantasy, Romance, Contemporary, Books...	4.12	105,324	16,843	304	Hardcover	First published May 30, 2023	Make a wish. . . .Lucy Hart knows better than ...	57702

The DataFrame contains book information across 12 columns, detailing various attributes of each book:

- Title: The title of the book.
- ISBN: The International Standard Book Number, a unique identifier for the book.
- Author: The author or authors of the book.
- Genre: The genre or category under which the book is classified.
- Rating: The average rating given to the book by readers.
- Number of Reviews: The total count of reviews that the book has received.
- Number of Ratings: The total number of ratings that the book has received.
- Pages: The total number of pages in the book.
- Format: The format of the book (e.g., paperback, hardcover, ebook).
- Publish Info: Information about the book's publication, such as the publisher and publication date.
- Description: A brief description or synopsis of the book.
- Votes: The number of votes the book has received for awards or in rankings.

2. NYT Books API

Another source of the data used in this report is from the NYT Books API, which provides information about book reviews and The New York Times Best Sellers lists. For this analysis, I collected all the Best Sellers lists from 2019 to 2023, along with the information on the books listed. The data is returned in JSON format, hence, I saved all the data as JSON files for subsequent processing.

To effectively automate data collection, I designed a series of functions that include error handling mechanisms and set a delay of 12 seconds to comply with the API rate limits. Below is an example of one of these functions:

```python
def fetch_best_sellers_monthly(api_key, year):
    base_url = "https://api.nytimes.com/svc/books/v3/lists/full-overview.json"
    results = {}
    start_date = datetime(year, 1, 1)

    for month in range(1, 13):  # For each month in the year
        try:
            date = start_date.strftime('%Y-%m-%d')
            params = {
                'api-key': api_key,
                'published_date': date
            }
            response = requests.get(base_url, params=params)
            if response.status_code == 200:        You, 6 days ago • work17/04
                results[date] = response.json()
                print(f"Data fetched for {date}")
            else:
                print(f"Failed to fetch data for {date}: {response.status_code}")

            # Add a delay of 12 seconds to manage API rate limits
            print("Waiting 12 seconds before next API call...")
            time.sleep(12)

            start_date += timedelta(days=31)  # Move to the next month
            start_date = start_date.replace(day=1)  # Ensure we start at the first of the month
        except Exception as e:
            print(f"Error processing date {date}: {str(e)}")

    return results
```

3. Open Source: Kaggle

For the dataset from open source, I discovered a dataset on Kaggle titled "Stress Analysis in Social Media." This dataset includes extensive multi-domain social media data from five different categories of Reddit communities, aimed at identifying stress. It primarily consists of comment texts from Reddit communities along with their associated stress labels (0 for no stress, 1 for stress). The dataset is divided into a test set (715 records) and a training set (2833 records). Before use, we will merge these two files and then re-split them into new test and training sets. The primary purpose of this dataset is for training NLP models that can be applied to the main data set.

Dataset Source: [Stress-Detection-with-Machine-Learning (kaggle.com)](kaggle.com)

Below is an excerpt from the data files:

About this file

dreaddit-test

id	subreddit	post_id	sentence_range	text	label
2 – 55.8k	anxiety 21% relationships 20% Other (426) 60%	586 unique values	[0, 5] 14% (0, 5) 13% Other (527) 74%	715 unique values	0
896	relationships	7nu7as	[50, 55]	Its like that, if you want or not." ME: I have no problem, if it takes longer. But you asked my frie...	0
19059	anxiety	680i6d	(5, 10)	I man the front desk and my title is HR Customer Service Representative. About 50% of my job is spen...	0
7977	ptsd	8eeu1t	(5, 10)	We'd be saving so much money with this new housr...its such an expensive city.... I did some googlin...	1

4. BigQuery

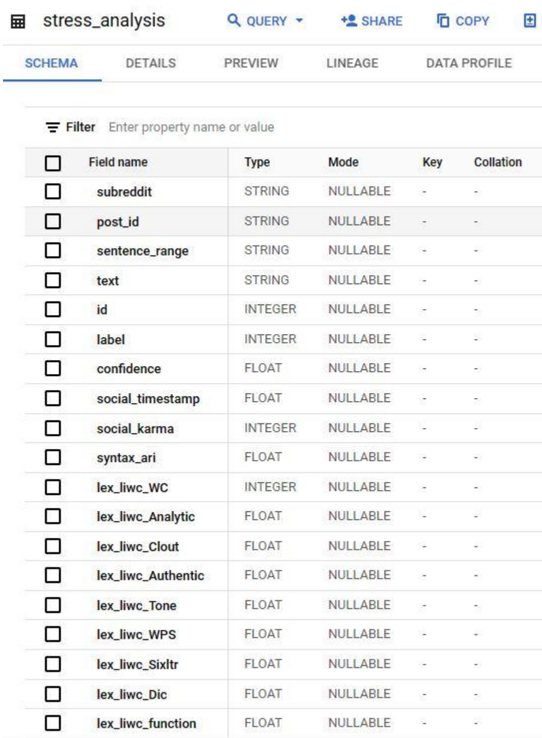
Since I was unable to locate an appropriate database on BigQuery, I opted to upload a cleaned CSV file, prepared for model training, to BigQuery for query execution. I established a connection between Python and BigQuery, uploaded the CSV file, and then conducted operations directly within BigQuery.

Codes on how to establish a connection from Python to BigQuery and upload the CSV.

```
import pandas as pd
from google.cloud import bigquery
# load csv
df = pd.read_csv("../data/clean_data/model_training.csv")
df.to_gbq('xiaobo.stress_analysis',project_id="da-bootcamp-2023",if_exists='replace')
```

```
100%|██████████| 1/1 [00:00<?, ?it/s]
```

Information on the schema setup and detailed description of the data structure.



Field name	Type	Mode	Key	Collation
subreddit	STRING	NULLABLE	-	-
post_id	STRING	NULLABLE	-	-
sentence_range	STRING	NULLABLE	-	-
text	STRING	NULLABLE	-	-
id	INTEGER	NULLABLE	-	-
label	INTEGER	NULLABLE	-	-
confidence	FLOAT	NULLABLE	-	-
social_timestamp	FLOAT	NULLABLE	-	-
social_karma	INTEGER	NULLABLE	-	-
syntax_ari	FLOAT	NULLABLE	-	-
lex_liwc_WC	INTEGER	NULLABLE	-	-
lex_liwc_Analytic	FLOAT	NULLABLE	-	-
lex_liwc_Clout	FLOAT	NULLABLE	-	-
lex_liwc_Authentic	FLOAT	NULLABLE	-	-
lex_liwc_Tone	FLOAT	NULLABLE	-	-
lex_liwc_WPS	FLOAT	NULLABLE	-	-
lex_liwc_Sixltr	FLOAT	NULLABLE	-	-
lex_liwc_Dic	FLOAT	NULLABLE	-	-
lex_liwc_function	FLOAT	NULLABLE	-	-

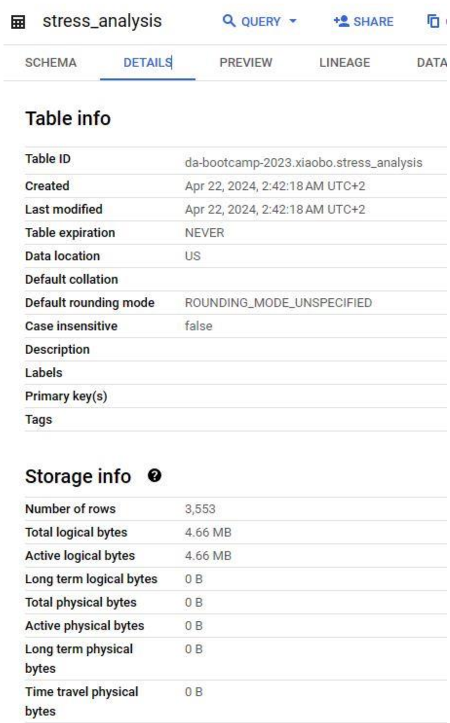


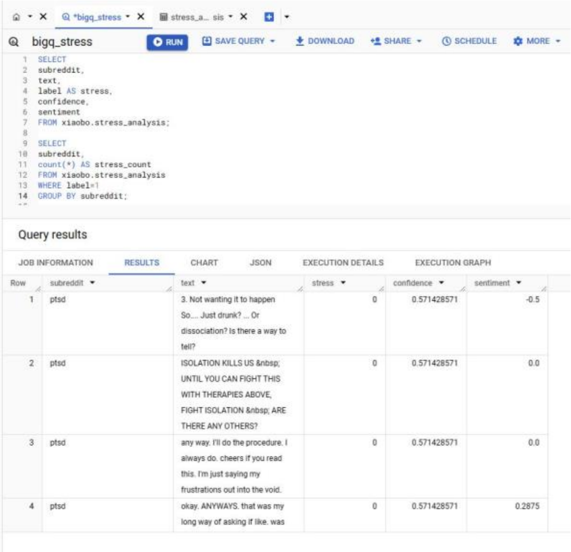
Table info

Table ID	da-bootcamp-2023.xiaobo.stress_analysis
Created	Apr 22, 2024, 2:42:18 AM UTC+2
Last modified	Apr 22, 2024, 2:42:18 AM UTC+2
Table expiration	NEVER
Data location	US
Default collation	
Default rounding mode	ROUNDING_MODE_UNSPECIFIED
Case insensitive	false
Description	
Labels	
Primary key(s)	
Tags	

Storage info

Number of rows	3,553
Total logical bytes	4.66 MB
Active logical bytes	4.66 MB
Long term logical bytes	0 B
Total physical bytes	0 B
Active physical bytes	0 B
Long term physical bytes	0 B
Time travel physical bytes	0 B

Examples of SQL Queries:

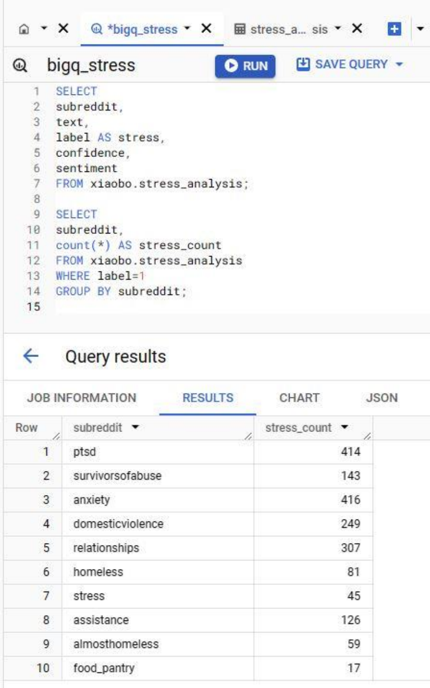


bigq_stress

```sql
SELECT
subreddit,
text,
label AS stress,
confidence,
sentiment
FROM xiaobo.stress_analysis;

SELECT
subreddit,
count(*) AS stress_count
FROM xiaobo.stress_analysis
WHERE label=1
GROUP BY subreddit;
```

Query results

Row	subreddit	text	stress	confidence	sentiment
1	ptsd	3. Not wanting it to happen So.... Just drunk? ... Or dissociation? Is there a way to tell?	0	0.571428571	-0.5
2	ptsd	ISOLATION KILLS US UNTIL YOU CAN FIGHT THIS WITH THERAPIES ABOVE, FIGHT ISOLATION ARE THERE ANY OTHERS?	0	0.571428571	0.0
3	ptsd	any way. I'll do the procedure. I always do. cheers if you read this. I'm just saying my frustrations out into the void.	0	0.571428571	0.0
4	ptsd	okay. ANYWAYS. that was my long way of asking if like. was	0	0.571428571	0.2875



bigq_stress

```sql
SELECT
subreddit,
text,
label AS stress,
confidence,
sentiment
FROM xiaobo.stress_analysis;

SELECT
subreddit,
count(*) AS stress_count
FROM xiaobo.stress_analysis
WHERE label=1
GROUP BY subreddit;
```

Query results

Row	subreddit	stress_count
1	ptsd	414
2	survivorsofabuse	143
3	anxiety	416
4	domesticviolence	249
5	relationships	307
6	homeless	81
7	stress	45
8	assistance	126
9	almosthomeless	59
10	food_pantry	17

Data Processing

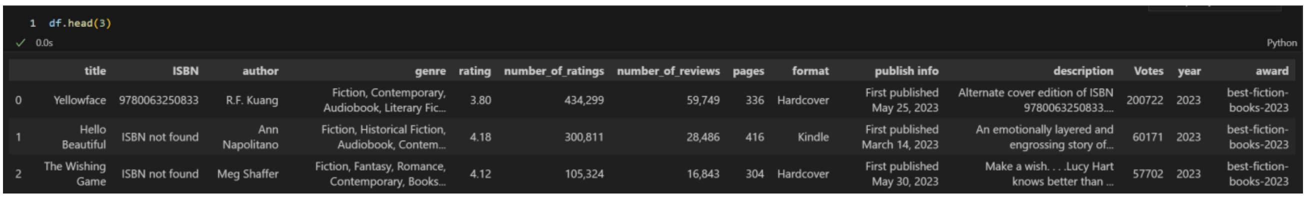
1. Data Cleaning

Data cleaning primarily involved three sets of data:

- All Choice Award-winning book information;
- The first 10 reader comments for each book;
- All Best Sellers book information.

Since this data was collected and saved annually, the first step required data integration.

For the book information (Book_info), I began by loading each year's data files separately, adding two columns, "year" and "award," to each dataset. Once the data was loaded, I merged them using the concat function.



```python
df.head(3)
```

	title	ISBN	author	genre	rating	number_of_ratings	number_of_reviews	pages	format	publish info	description	Votes	year	award
0	Yellowface	9780063250833	R.F. Kuang	Fiction, Contemporary, Audiobook, Literary Fic...	3.80	434,299	59,749	336	Hardcover	First published May 25, 2023	Alternate cover edition of ISBN 9780063250833....	200722	2023	best-fiction-books-2023
1	Hello Beautiful	ISBN not found	Ann Napolitano	Fiction, Historical Fiction, Audiobook, Contem...	4.18	300,811	28,486	416	Kindle	First published March 14, 2023	An emotionally layered and engrossing story of...	60171	2023	best-fiction-books-2023
2	The Wishing Game	ISBN not found	Meg Shaffer	Fiction, Fantasy, Romance, Contemporary, Books...	4.12	105,324	16,843	304	Hardcover	First published May 30, 2023	Make a wish. . . .Lucy Hart knows better than ...	57702	2023	best-fiction-books-2023

In the combined dataset, I noticed a large number of entries in the ISBN column labeled

"ISBN not found." Considering these entries do not impact our analysis, I chose to remove them. Similarly, I also removed records where the rating was 0.

```
df = df.drop(columns=['ISBN'])
```

```
df_filtered = df[df['number_of_ratings'] != 0]
df_filtered.head(3)
```

Next, I processed the "publish_info" column by extracting the publication date and converting it into datetime format, preparing for the subsequent creation of a database.

```
df['publish_date'] = df['publish info'].str.extract(r'(\w+ \d{1,2}, \d{4})')
df['publish_date'] = pd.to_datetime(df['publish_date'])

# drop "publish info"
df = df.drop(columns=['publish info'])
```

Additionally, we needed a list of awards. I read the previously web-scraped "award_category.json" file, converting it into a DataFrame. On this basis, I added a "year" column and an additional "award_id" column, and saved this DataFrame as "choice_award.csv".

```
award_df_clean['award_id'] = [str(i) for i in range(1, len(award_df_clean) + 1)]
```

By merging the award DataFrame (award_df) with the book information DataFrame (book_info_df), I added the "award_id" to book_info_df.

Upon revisiting the data, I noticed that the "genre" column was overly detailed, which could complicate future analyses. Therefore, I decided to implement category refinement for this column:

Identify some common book genre categories and map the existing detailed genres to these broader categories. Below are some examples of basic category mappings:

- Fantasy/Sci-Fi/Horror
- Literature/Fiction
- Nonfiction/Education
- Romance/Drama
- Young Adult/Children

This step was implemented by defining a function:

```python
def map_genre_to_main_category(genre):

    if any(sub in genre.lower() for sub in ['fantasy', 'sci-fi', 'science fiction', 'horror', 'paranormal']):
        return 'Fantasy/Sci-Fi/Horror'
    elif any(sub in genre.lower() for sub in ['fiction', 'novel', 'literature']):
        return 'Literature/Fiction'
    elif any(sub in genre.lower() for sub in ['nonfiction', 'education', 'biography', 'memoir', 'self-help']):
        return 'Nonfiction/Education'
    elif any(sub in genre.lower() for sub in ['romance', 'drama']):
        return 'Romance/Drama'
    elif any(sub in genre.lower() for sub in ['young adult', 'children', 'ya']):
        return 'Young Adult/Children'
    else:
        return 'Other'
```

Finally, I modified the format (converted all to lowercase) and order of each column name, resulting in the following DataFrame content:

	title	author	genre	publish_date	pages	format	rating	rating_counts	reviews_counts	votes	year	description	award_id	main_genre
0	The Testaments	Margaret Atwood	Fiction, Dystopia, Science Fiction, Feminism, ...	2019-09-10	422.0	Hardcover	4.19	352440	31334	98291	2019	When the van door slammed on Offred's future a...	70	Fantasy/Sci-Fi/Horror
1	Normal People	Sally Rooney	Fiction, Romance, Contemporary, Literary Ficti...	2018-08-28	273.0	Hardcover	3.81	1388906	135364	40081	2019	At school Connell and Marianne pretend not to ...	70	Literature/Fiction
2	Where the Forest Meets the Stars	Glendy Vanderah	Fiction, Contemporary, Mystery, Romance, Fanta...	2019-03-01	332.0	Hardcover	4.14	192265	16504	29342	2019	In this gorgeously stunning debut, a mysteriou...	70	Fantasy/Sci-Fi/Horror

Next, I cleaned the bestsellers data. Since the data was saved in JSON format, I first analyzed the structure of the data. Then, I wrote a new function to process these files, calling this function on each file to extract the necessary book information.

```python
def get_bestsellers_info_df(data):
    books_info = []
    for item in data.values():
        results = item['results']
        bestsellers_date = results['bestsellers_date']
        for list_info in results['lists']:
            list_name = list_info['list_name_encoded']
            for book in list_info['books']:
                book_details = {
                    "title": book['title'],
                    "rank": book['rank'],
                    "publisher": book['publisher'],
                    "weeks_on_list": book['weeks_on_list'],
                    "primary_isbn13": book['primary_isbn13'],
                    "list_name": list_name,
                    "bestsellers_date": bestsellers_date
                }
                books_info.append(book_details)
    books_info_df = pd.DataFrame(books_info)
    return books_info_df
```

Finally, merge the data from all years into a DataFrame:

```
1 df.head(3)
```

	title	rank	publisher	weeks_on_list	primary_isbn13	list_name	bestsellers_date
0	THE RECKONING	1	Doubleday	9	9780385544153	combined-print-and-e-book-fiction	2018-12-22
1	EVERY BREATH	2	Grand Central	10	9781538728529	combined-print-and-e-book-fiction	2018-12-22
2	FIRE AND BLOOD	3	Bantam	5	9781524796280	combined-print-and-e-book-fiction	2018-12-22

Similarly, to facilitate the creation of a database later, we have extracted the list_name separately and added a list_id to form a table of book lists.

2. Text Cleaning

In this step, we will consolidate the reader comments into a new dataset, which will include the title of each book, comment ID, reader's name, time of the comment, and the text.

This is still done by defining a function and calling it to read information from the JSON files:

```python
def review_to_df(data):
    reviews = []

    # Iterate through each book and its reviews
    for book_id, reviews_dict in data.items():
        for review_id, review_content in reviews_dict.items():
            # Extracting the reviewer's name and review text
            # Here we assume the name ends at the first digit, which is extremely simplistic
            reviewer_name = ''.join([char for char in review_content if not char.isdigit()]).split(',')[0]
            review_text = review_content.split('followers')[-1]  # This is a simplistic split

            # Append the extracted information as a dictionary to the list
            reviews.append({
                'title_id': book_id,
                'review_id': review_id,
                'reviewer_name': reviewer_name.strip(),
                'review_text': review_text.strip()
            })
    df_reviews = pd.DataFrame(reviews)
    return df_reviews
```

Upon examination, it can be observed that the comment time and the comment text are mixed together, so it is necessary to extract the comment time separately.

```python
# Regular expression to match the date format
# Define the date pattern
date_pattern = r'(January|February|March|April|May|June|July|August|September|October|November|December)\s\d{1,2},\s\d{4}'

# Function to extract and remove dates
def extract_and_remove_date(text):
    match = re.search(date_pattern, text)
    if match:
        date = match.group(0)  # Extract the date
        cleaned_text = re.sub(date_pattern, '', text)  # Remove the date from the text
        return date, cleaned_text
    return None, text  # Return None for date and original text if no date found
```

Then, merge the data from each year to obtain the total data as follows:

	title	review_id	reviewer_name	date	review_text
0	the testaments	1	Tatiana	May 29, 2020	I guess I'll have to be the one who says what ...
1	the testaments	2	Emily May	September 13, 2019	I can sum it up simply:this book is not needed...
2	the testaments	3	Marchpane reviews	January 22, 2020	Return to GileadCheck your expectations at the...
3	the testaments	4	Nilufer Ozmekik	August 15, 2021	Winner of best fiction category but it's not m...
4	the testaments	5	Sean Barrs	December 4, 2019	A review in 5 words:Unnecessary. Pointless. Ru...

Since we need to perform natural language processing (NLP) later, we will preprocess the text data. This involves importing the nltk library and performing tokenization, removing stopwords, and applying lemmatization to the text.

```
def preprocess_text(text):
    # Tokenize text
    tokens = word_tokenize(text.lower())  # Convert to lower case
    # Remove stopwords and lemmatize
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens if token.isalpha() and token not in stop_words]
    return ' '.join(lemmatized_tokens)
```

	review	processed_text
0	[3.5] There's some disagreement within the CF ...	disagreement within cf community whether book ...
1	Edit: For the love of God I wrote this review ...	edit love god wrote review two year ago still ...
2	Can't say I'm surprised I didn't enjoy this bo...	ca say surprised enjoy book part goodreads rea...

Database Type Selection and Database Creation

Relational Database vs Non-Relational Database

When deciding on database technology, comparing SQL (Structured Query Language) and NoSQL (non-SQL or non-relational databases) is essential. These two database technologies have significant differences in data storage, querying, scalability, and use cases. Here's a brief comparison:

	SQL (Relational Databases)	NoSQL (Non-Relational Databases)
Structure	Uses strict predefined schemas with data stored in table formats, and tables can have complex relationships.	Lacks a fixed schema, capable of storing structured, semi-structured, or unstructured data.
Query Language	Utilizes SQL for querying data, a powerful tool for executing complex queries and data manipulations.	Not uniform and depends on the specific database product, generally does not support SQL.
Scalability	Primarily supports vertical scaling (upgrading existing hardware).	Designed to support horizontal scaling, suitable for handling large-scale data distribution.
Data Integrity	Highly emphasizes ACID properties (Atomicity, Consistency, Isolation, Durability), suitable for applications requiring strict data consistency.	Most adopt an eventual consistency model, not guaranteeing immediate consistency.
Use Cases	Financial services, enterprise applications, any scenario that requires complex transactional processing.	Big data applications, real-time processing applications, content management systems, etc.

In summary, SQL (Structured Query Language) is typically associated with relational databases. Relational databases, such as MySQL, PostgreSQL, Oracle, and Microsoft SQL Server, use SQL to manage and manipulate the data stored in tables that are related to each other through foreign keys and other constraints. This type of database is designed to handle structured data and supports complex queries and transactions.

In our case, our tables are related to each other, to better handle this data, we have chosen to use SQL.

MySQL Database Setup

In order to facilitate linking between tables, we needed to assign an ID to each book. Therefore, I extracted all book titles from the book_info and bestsellers lists into a single list, standardized the titles to lowercase, and dropped duplicates. This process resulted in a table containing all books, to which I added a title_id.

Consequently, after data cleaning and integration, we ended up with the following six tables:

- book_info: books information of Choice Award
- bestsellers: books information of NYT bestsellers
- books_title: list of the titles of all the books
- choice_award: category of award of each year
- lists_bestsellers: lists of NYT bestsellers
- reviews: first 10 reviews for each award book

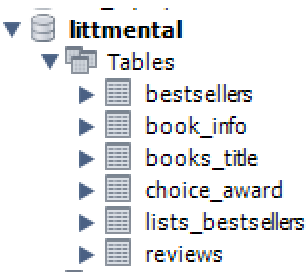
Then I connected Python to MySQL in order to export datasets.

Example of codes for create database and tables:

```
with engine.connect() as conn:
    conn.execute(text("CREATE DATABASE IF NOT EXISTS LittMental"))
```

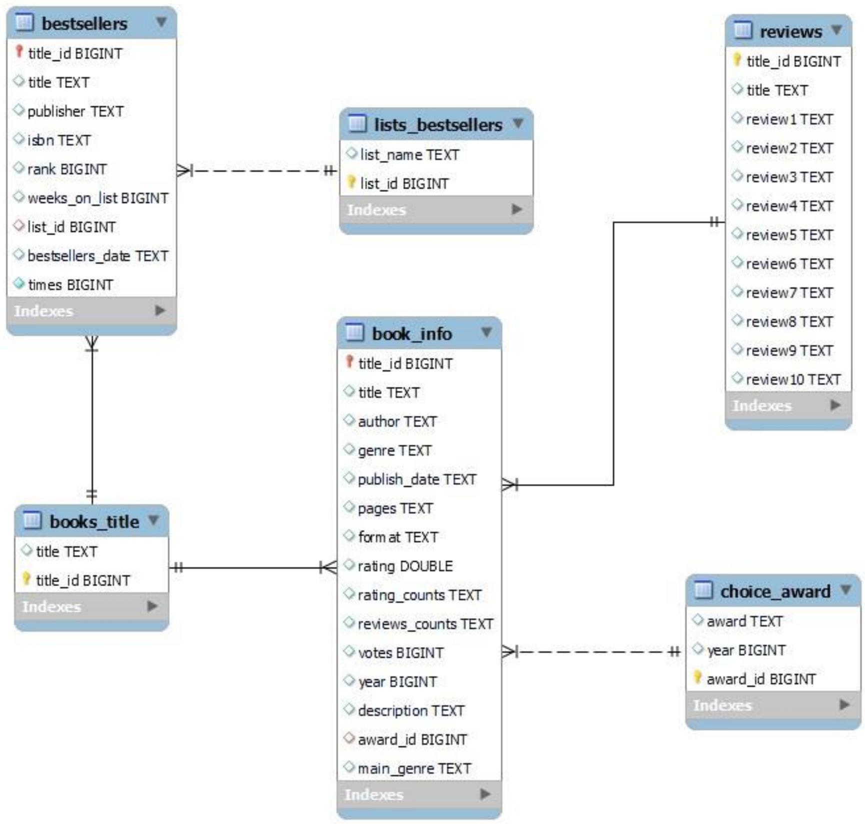
```
book_info=pd.read_csv('../data/database/book_info.csv')
book_info.to_sql('book_info',engine, 'LittMental', if_exists='replace', index=False)
```

After creation of database, I added primary key and foreign key for each table in MySQL in order to define the relationships and create ERD diagram.




```sql
ALTER TABLE bestsellers
ADD PRIMARY KEY (title_id);

ALTER TABLE book_info
ADD PRIMARY KEY (title_id);

ALTER TABLE lists_bestsellers
ADD PRIMARY KEY (list_id);

ALTER TABLE books_title
ADD PRIMARY KEY (title_id);

ALTER TABLE reviews
ADD PRIMARY KEY (title_id);

ALTER TABLE choice_award
ADD PRIMARY KEY (award_id);

ALTER TABLE bestsellers
ADD FOREIGN KEY (list_id) REFERENCES lists_bestsellers(list_id);

ALTER TABLE bestsellers
ADD FOREIGN KEY (title_id) REFERENCES books_title(title_id);

ALTER TABLE book_info
ADD FOREIGN KEY (title_id) REFERENCES books_title(title_id);

ALTER TABLE book_info
ADD FOREIGN KEY (title_id) REFERENCES reviews(title_id);

ALTER TABLE book_info
ADD FOREIGN KEY (award_id) REFERENCES choice_award(award_id);
```

Here is the ERD diagram:



MySQL Data Queries

In MySQL, I utilized basic queries such as SELECT statements, along with more advanced operations including joins, aggregate functions, and the creation of views, to further refine the tables. For example, I added an 'is_bestseller' column to the 'book_info' table. Below are the queries I used:

```
-- count the times of being bestsellers and create new table
CREATE TABLE IF NOT EXISTS clean_bestsellers AS
    SELECT *, count(*) AS times FROM bestsellers
    GROUP BY title_id;
DROP TABLE bestsellers;
RENAME TABLE clean_bestsellers TO bestsellers;

-- join the reviews table with title_id and create new table
CREATE TABLE IF NOT EXISTS new_reviews AS
    SELECT bt.title_id, r.*
    FROM reviews AS r
    JOIN books_title AS bt
    ON r.title = bt.title
    GROUP BY r.title
    ORDER BY bt.title_id;
DROP TABLE reviews;
RENAME TABLE new_reviews TO reviews;
```

```
-- join the book_info table with title_id and create new table
CREATE TABLE IF NOT EXISTS new_book_info AS
    SELECT bt.title_id, b.*
    FROM book_info AS b
    JOIN books_title AS bt
    ON b.title = bt.title
    GROUP BY b.title
    ORDER BY bt.title_id;
DROP TABLE book_info;
RENAME TABLE new_book_info TO book_info;

-- in book_info, check if the book is bestsellers
DROP VIEW IF EXISTS clean_book_info;
CREATE VIEW clean_book_info AS
    SELECT b.*,
        COUNT(DISTINCT bs.title) AS is_bestseller,
        c.award
    FROM book_info AS b
    LEFT JOIN bestsellers AS bs ON b.title = bs.title
    JOIN choice_award AS c ON b.award_id = c.award_id
    GROUP BY b.title
    ORDER BY b.year, b.votes desc;
```

API Development and Data Exposition via API

Flask API Development

The initial step in developing our "bookapi" was to set up and configure the Flask environment. Flask is a lightweight and powerful micro web framework for Python, well-suited for creating web applications and APIs.

We created a new Python script called bookapi.py to serve as the entry point of our application. And we configured the Flask app to run in debug mode for development, enabling live reloading and better error tracking. This was set directly in the “app.run()“ method by passing “debug=True”.

API Endpoint Construction

With the environment set up, we focused on the development of API endpoints for "bookapi". These endpoints were designed to manage book data, including retrieving book records by title_id, and retrieving lists of awards by year.

There are 4 endpoints designed:

- Get books (GET /v1/books): Retrieves a list of all books;
- Get books by id (GET /v1/books/{title_id}):Retrieves a specific book by title_id;
- Get list of award (GET /v1/awardslist): Retrieve all the lists of ChoiceAwards from 2019 to 2023;
- Get list of award by year (GET /v1/awardslist/{year}): Access list of award of a specific year

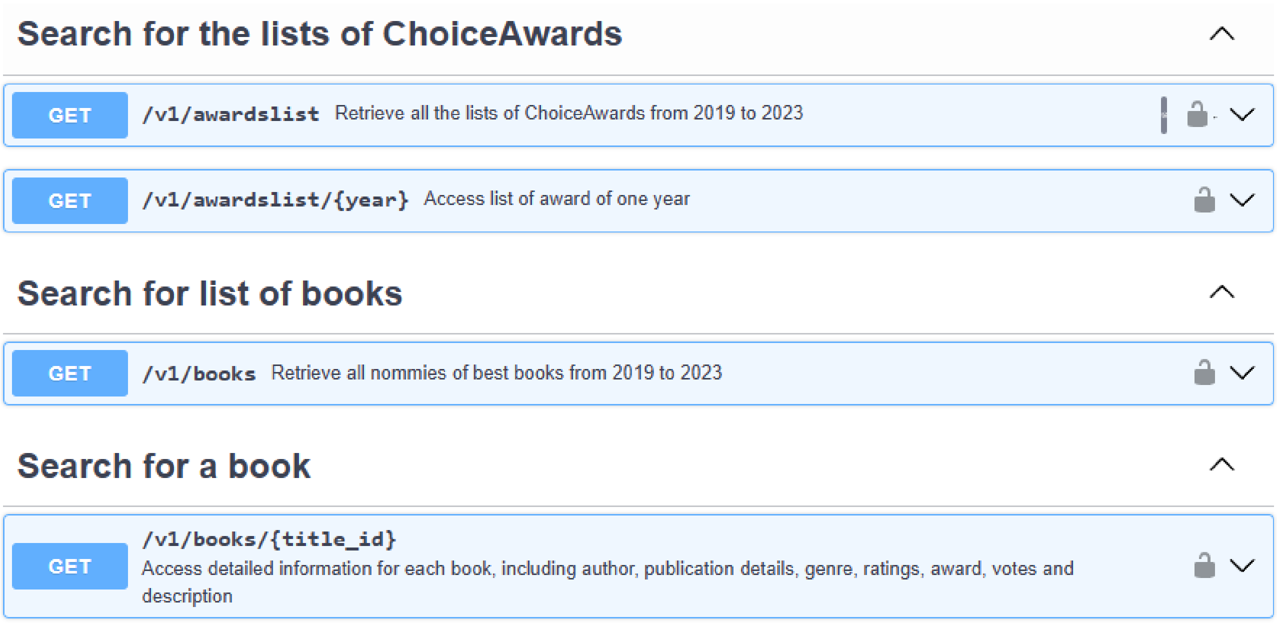
Here is the example of exposition of data:


```
{
    "Title": "Five Feet Apart",
    "author": "Rachael Lippincott",
    "award": "best-young-adult-fiction-books-2019",
    "format": "Hardcover",
    "genres": "Romance, Young Adult, Contemporary, Fiction, Audiobook, Realistic Fiction, Young Adult Romance",
    "more_info": {},
    "pages": "304.0",
    "publish_date": "2018-11-20",
    "rating": 4.18,
    "title_id": 111
}
```

Documentation

We documented the API endpoints using Swagger, which provides an interactive UI for exploring and testing the API. This step was crucial for ensuring that future developers and users could understand and use the API effectively.

Here is how API looks like in swagger:



This setup and endpoint development provided a robust foundation for our "bookapi", enabling easy access and manipulation of book data in a structured and efficient manner. The API is now ready for further integration and use in client applications.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis

Our project primarily revolves around the "clean_book_info" dataset, which serves as the cornerstone for our subsequent analyses and API development. To begin the data handling process, we first loaded the "book_info.csv" file into our working environment. This initial step was crucial for setting the stage for a comprehensive data review and manipulation process.

First of all, we loaded the CSV file, after loading the data, it was essential to understand its structure and contents. We utilized various pandas functions to inspect the dataset: shape, dytpes, nunique, describe().

```
1  book_info.shape

(1686, 15)
```

```
1  book_info.dtypes

title_id            int64
title              object
author             object
publish_date       object
pages             float64
format             object
rating            float64
rating_counts       int64
reviews_counts      int64
votes               int64
year                int64
description        object
main_genre         object
is_bestseller       int64
award              object
dtype: object
```

```
1  book_info.nunique()

title_id          1686
title             1686
author            1304
publish_date       404
pages              416
format               8
rating             132
rating_counts     1660
reviews_counts    1547
votes             1606
year                 5
description       1686
main_genre           5
is_bestseller        2
award               89
dtype: int64
```

```
1  book_info.isnull().sum()

title_id          0
title             0
author            0
publish_date      0
pages             0
format            0
rating            0
rating_counts     0
reviews_counts    0
votes             0
year              0
description       0
main_genre        0
is_bestseller     0
award             0
dtype: int64
```

```
1  book_info.describe(include='all')
```

Python

	title_id	title	author	publish_date	pages	format	rating	rating_counts	reviews_counts	votes	year	description	main_genre	is_bestseller	award
count	1686.000000	1686	1686	1686	1686.000000	1686	1686.000000	1.686000e+03	1686.000000	1686.000000	1686.000000	1686	1686	1686.000000	1686
unique	NaN	1686	1304	404	NaN	8	NaN	NaN	NaN	NaN	NaN	1686	5	NaN	89
top	NaN	The Testaments	Brandon Sanderson	2021-05-04	NaN	Hardcover	NaN	NaN	NaN	NaN	NaN	When the van door slammed on Offred's future a...	Literature/Fiction	NaN	best-fiction-books-2019
freq	NaN	1	7	18	NaN	1393	NaN	NaN	NaN	NaN	NaN	1	1082	NaN	20
mean	3276.955516	NaN	NaN	NaN	345.276987	NaN	4.046459	7.901742e+04	9173.467972	14746.444840	2020.848161	NaN	NaN	0.254448	NaN
std	1165.691971	NaN	NaN	NaN	137.123717	NaN	0.266701	1.833264e+05	18631.789505	21167.234433	1.403076	NaN	NaN	0.435680	NaN
min	111.000000	NaN	NaN	NaN	0.000000	NaN	3.160000	8.200000e+01	19.000000	383.000000	2019.000000	NaN	NaN	0.000000	NaN
25%	2937.750000	NaN	NaN	NaN	273.000000	NaN	3.840000	7.302250e+03	1080.250000	2844.250000	2020.000000	NaN	NaN	0.000000	NaN
50%	3585.500000	NaN	NaN	NaN	338.000000	NaN	4.050000	2.400350e+04	3586.500000	7365.500000	2021.000000	NaN	NaN	0.000000	NaN
75%	4150.750000	NaN	NaN	NaN	406.000000	NaN	4.240000	7.305325e+04	9216.750000	19772.750000	2022.000000	NaN	NaN	1.000000	NaN
max	4697.000000	NaN	NaN	NaN	1242.000000	NaN	4.750000	2.683087e+06	234321.000000	397565.000000	2023.000000	NaN	NaN	1.000000	NaN

```
1 book_info.describe()
```

	title_id	pages	rating	rating_counts	reviews_counts	votes	year	is_bestseller
count	1686.000000	1686.000000	1686.000000	1.686000e+03	1686.000000	1686.000000	1686.000000	1686.000000
mean	3276.955516	345.276987	4.046459	7.901742e+04	9173.467972	14746.444840	2020.848161	0.254448
std	1165.691971	137.123717	0.266701	1.833264e+05	18631.789505	21167.234433	1.403076	0.435680
min	111.000000	0.000000	3.160000	8.200000e+01	19.000000	383.000000	2019.000000	0.000000
25%	2937.750000	273.000000	3.840000	7.302250e+03	1080.250000	2844.250000	2020.000000	0.000000
50%	3585.500000	338.000000	4.050000	2.400350e+04	3586.500000	7365.500000	2021.000000	0.000000
75%	4150.750000	406.000000	4.240000	7.305325e+04	9216.750000	19772.750000	2022.000000	1.000000
max	4697.000000	1242.000000	4.750000	2.683087e+06	234321.000000	397565.000000	2023.000000	1.000000

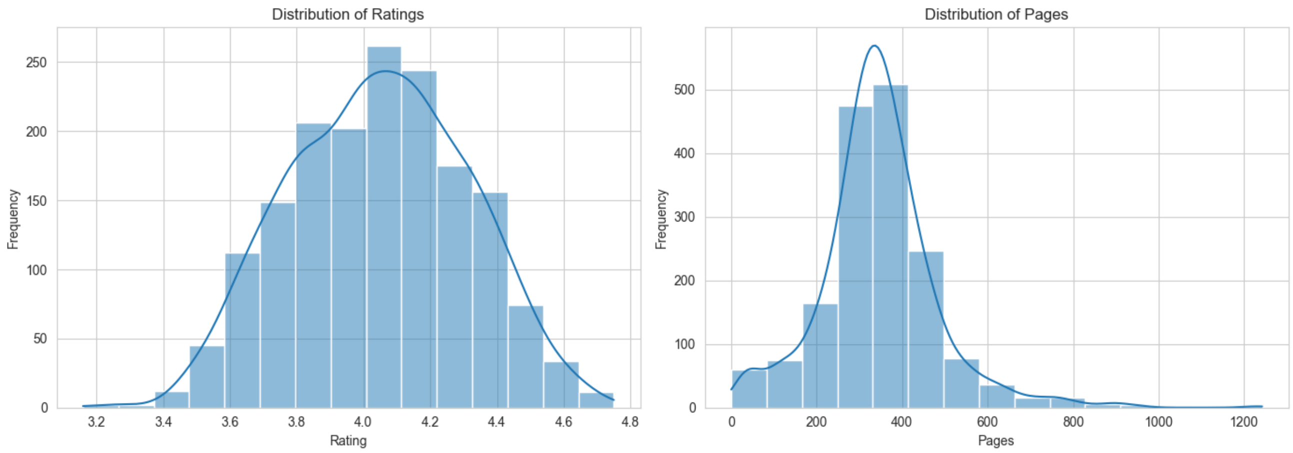
From the basic statistics, we can see that there are 1686 entries and each entry has details spanning 15 columns.We can also observe the following points:

- **Book Titles and Authors:**
  - There are 1686 unique titles, each authored by one of 1304 different authors, with Brandon Sanderson being the most frequent.
- **Publication Dates:**
  - Dates range from February 3, 2012, to November 13, 2023, with the average publication year around early 2021.
- **Pages:**
  - The number of pages ranges from 0 (possibly an error or placeholder) to 1242, with an average of approximately 345 pages.
- **Formats:**
  - The majority of the books are hardcovers.
- **Ratings and Reviews:**
  - Ratings range from 3.16 to 4.75, with an average rating of about 4.05.
  - The number of ratings varies greatly, indicating varying popularity, with some books receiving as many as 2,683,087 ratings.
  - Reviews also vary widely, pointing to differing levels of engagement.
- **Votes and Bestseller Status:**
  - A broad range of votes, reflecting reader engagement, from 383 to 397,565.
  - About 25.4% of books are marked as bestsellers.
- **Genres and Awards:**
  - The dataset covers five main genres, with 'Literature/Fiction' being the most common.
  - 89 different awards are represented, highlighting a diverse range of recognized works.

Visualization

In order to better understand the distribution and relationships within the data, we created several different visualizations:

- Histograms for the distribution of ratings and pages.
- Bar charts for the format and main genre categories.
- Pie chart for the proportion of bestsellers vs. non-bestsellers
- Bar chart to compare average ratings between bestsellers and non-bestsellers
- Correlation heatmap to examine relationships between numerical variables.



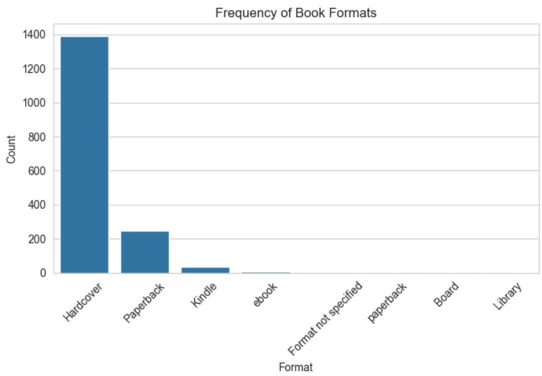
The histograms showing the distribution of ratings and pages within dataset:

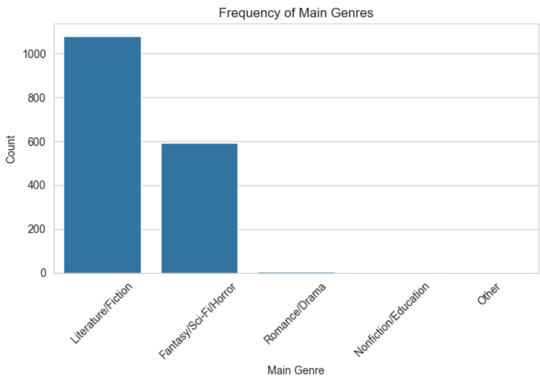
1. Ratings Distribution:

The ratings are fairly normally distributed with a slight left skew. Most books have ratings between 3.8 and 4.3, with a peak around 4.0 to 4.1. This indicates a generally positive reception of books in the dataset.

2. Pages Distribution:

The distribution of pages is right-skewed, showing that most books have fewer pages, typically ranging between 200 to 400 pages. There are fewer books with a very high page count, which is typical for general literature.





The bar charts illustrate the distribution of book formats and main genres in dataset:

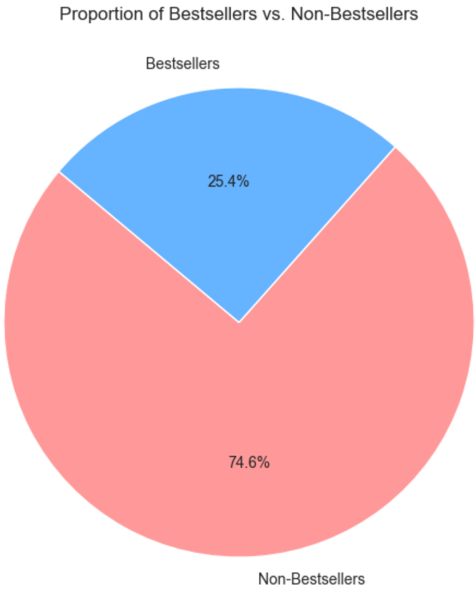
1. **Book Formats:**

The most common format is Hardcover, followed by Paperback. This shows a preference or predominance of these formats in the publication of books in your dataset. Other formats like Audiobook, E-book, and others are less frequent.

2. **Main Genres:**

Literature/Fiction dominates the genre distribution, indicating that a significant portion of the books falls within this category. Other genres like Fantasy/Sci-Fi/Horror and Nonfiction are also represented, but to a lesser extent.

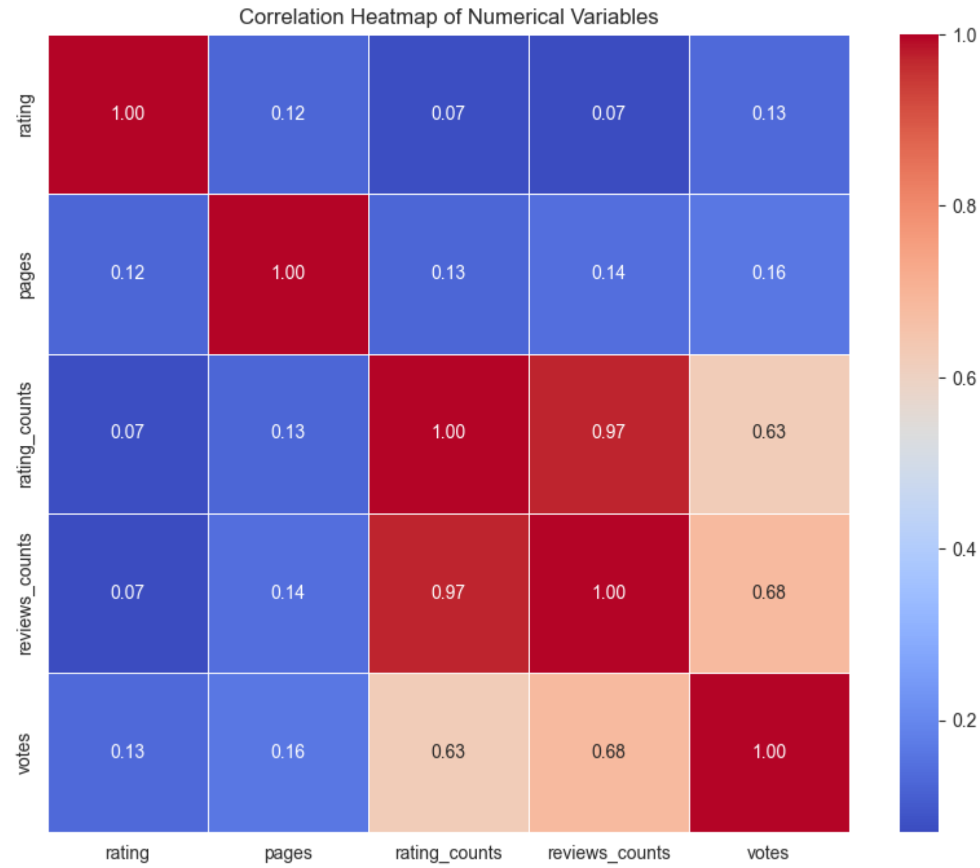
Next, the visualization to explore the distribution of books that are marked as bestsellers.



The Pie Chart illustrates the proportion of bestsellers versus non-bestsellers. It shows that 25.4% of the books in your dataset are bestsellers, while 74.6% are not. This gives a clear picture of the distribution of bestsellers in your collection.

The Bar Chart displays the average ratings for bestsellers and non-bestsellers. There is a visible difference in average ratings, suggesting that bestsellers tend to have higher ratings compared to non-bestsellers. This could indicate that higher-rated books have a better chance of becoming bestsellers, or that becoming a bestseller might positively influence the ratings of a book.

Lastly, I created a correlation heatmap to explore the relationships between the numerical variables such as ratings, pages, rating_counts, reviews_counts, and votes. This can help identify any interesting patterns or dependencies between these factors



The correlation heatmap provides insights into how different numerical variables in your dataset are related:

1. **Rating vs. Rating Counts, Reviews Counts, and Votes**: These variables have low to moderate positive correlations with the rating. This suggests that higher-rated books tend to get more ratings, reviews, and votes, although the relationship is not very strong.

2. **Rating Counts vs. Reviews Counts and Votes**: There's a strong positive correlation between the number of ratings a book receives and the number of reviews and votes. This indicates that books that are rated more frequently also tend to have more reviews and votes.

3. **Reviews Counts vs. Votes**: There is a very strong correlation between reviews counts and votes, which makes sense as both metrics reflect the level of reader engagement and popularity.

These correlations help in understanding the dynamics between different measures of book popularity and reception. Such insights could be valuable for decisions related to marketing, stocking, or recommending books.

Development and Application of NLP Models

Model Development

1. Dataset Preparation

In this phase of the project, we utilized the "Stress Analysis in Social Media" dataset, which comprised separate test and train files. These were merged into a single DataFrame, thereby consolidating our data and facilitating the creation of a comprehensive dataset for model training purposes.

2. Text Preprocessing

Prior to training our model, we performed several essential preprocessing steps on the textual data. We employed the predefined functions mentioned earlier, which encompassed:

- **Tokenization**: The text was broken down into individual words or tokens, enabling us to analyze the text at a granular level.

- **Stopwords Removal**: Common words that typically do not contribute much meaning to a sentence (e.g., "the", "is", "in") were removed to reduce noise in the dataset.

- **Lemmatization**: Words were reduced to their base or dictionary form, aiding in the normalization of the dataset for more effective training.

These preprocessing steps were critical to refine our dataset, making it more amenable to machine learning algorithms by removing irrelevant variations in the data. As mentioned earlier, we processed the texts by calling defined functions.

3. Model Training

Following the preprocessing steps, we partitioned the data into training and test subsets, ensuring that the model would be evaluated on unseen data, thus providing an unbiased assessment of its performance. We allocated 80% of the data for training and reserved 20% for testing, adhering to a commonly accepted practice for model evaluation.

We constructed a text classification pipeline that employed TF-IDF (Term Frequency-Inverse Document Frequency) for vectorization, paired with Logistic Regression, a robust and widely used algorithm for binary classification tasks. The Logistic Regression model was configured to iterate up to 1000 times to ensure convergence.

```
# Prepare data
X = stress['processed_text']
y = stress['stress']  # Using the 'stress' column as the target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a text classification pipeline with TF-IDF and Logistic Regression
pipeline = make_pipeline(TfidfVectorizer(), LogisticRegression(max_iter=1000))

# Train the model
pipeline.fit(X_train, y_train)
```

Model Evaluation

The model's training phase proceeded without incident, and we then applied the trained model to the test data, obtaining predictions. We printed classification report to show the performance metrics for the model.

	precision	recall	f1-score	support
0	0.72	0.72	0.72	323
1	0.77	0.77	0.77	388
accuracy			0.75	711
macro avg	0.75	0.75	0.75	711
weighted avg	0.75	0.75	0.75	711

These results suggest that the model performs fairly well, with balanced precision and recall across both classes. The accuracy of 0.75 indicates that the model correctly predicts stress 75% of the time, which is a respectable figure given the complexity of natural language processing and the challenges inherent in classifying such nuanced data.

The model demonstrates promising capabilities in identifying stress in social media text, though there is room for improvement. Future work might include exploring alternative machine learning algorithms, tuning hyperparameters, or employing more sophisticated natural language processing techniques to enhance the model's performance.

Implementation

The model was saved as a Joblib file, which was then loaded to be applied to our review dataset. Below are the results following its application.

```python
reviews_data['predicted_stress'] = model.predict(reviews_data['processed_text'])
reviews_data[['processed_text', 'predicted_stress']].head()
```

✓ 1.2s

	processed_text	predicted_stress
0	disagreement within cf community whether book ...	0
1	edit love god wrote review two year ago still ...	0
2	ca say surprised enjoy book part goodreads rea...	1
3	subjected mental emotional pain could also def...	1
4	two thing start really really really wanted li...	1

```python
reviews_data['predicted_stress'].value_counts()
```

✓ 0.0s

```
predicted_stress
0    14973
1     1887
Name: count, dtype: int64
```

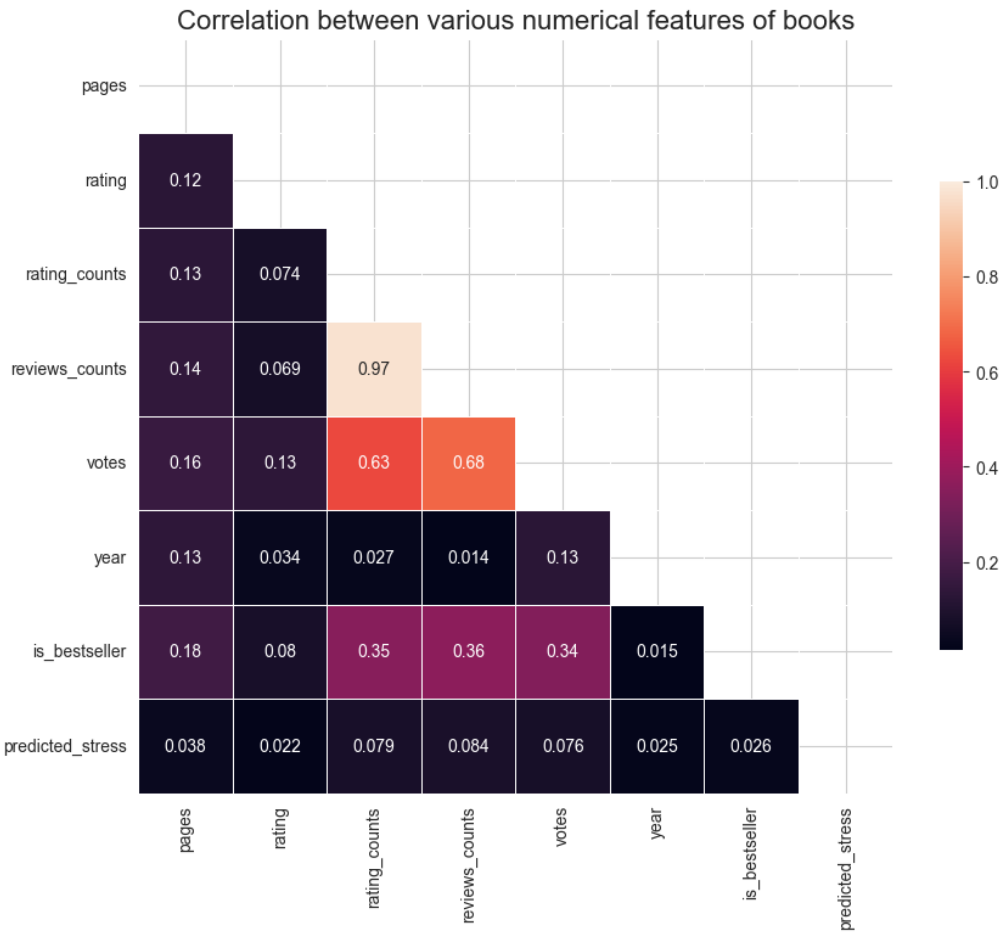
Advanced Analysis and Further Insights

Incorporating the newly developed “review_with_stress” DataFrame, which labels stress within book reviews, and merging it with our original “clean_book_info” data has opened avenues for more nuanced analysis.

Advanced Analysis of Stress in Book Reviews

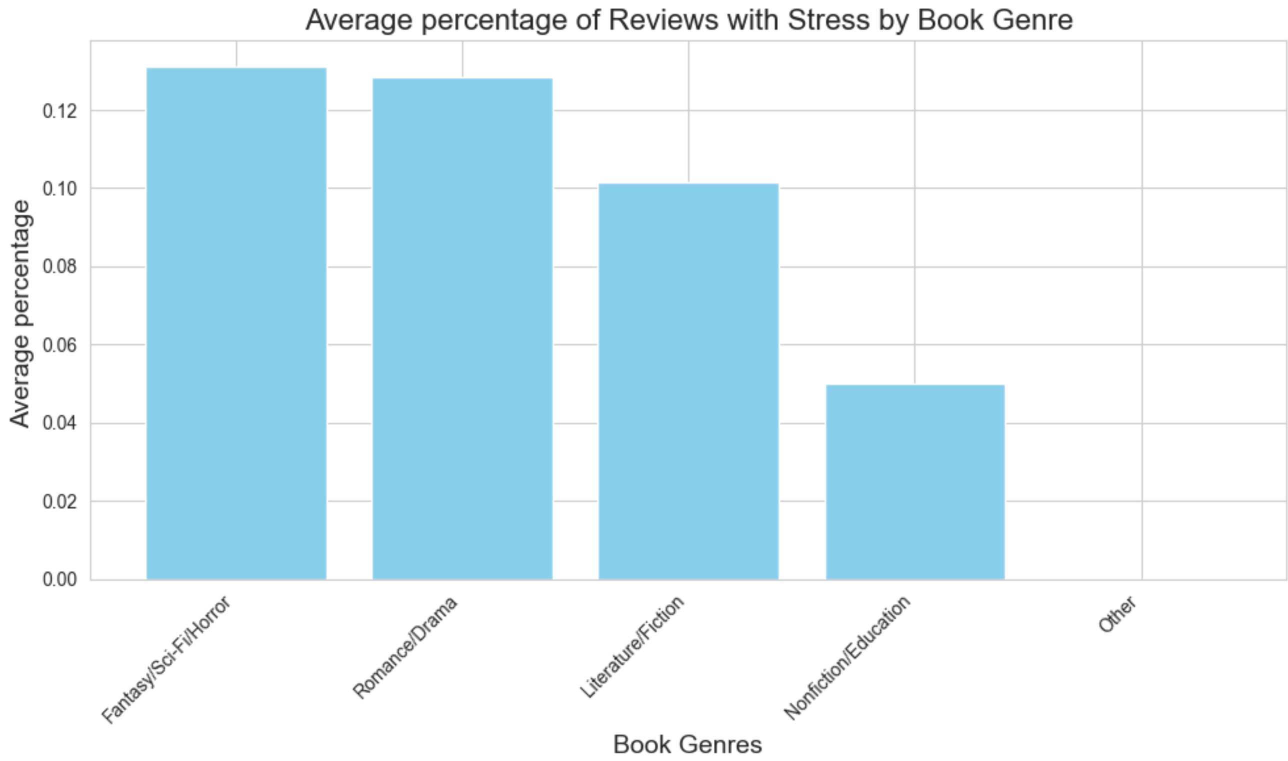
Our exploration into the relationship between stress in social media and literary works took a significant step forward with the integration of the review_with_stress DataFrame into our existing book information dataset. The dataset, now enriched with stress indicators derived from our NLP model, includes 16,860 reviews, allowing for a detailed analysis of stress mentions across ten reviews per book.

1. Correlation



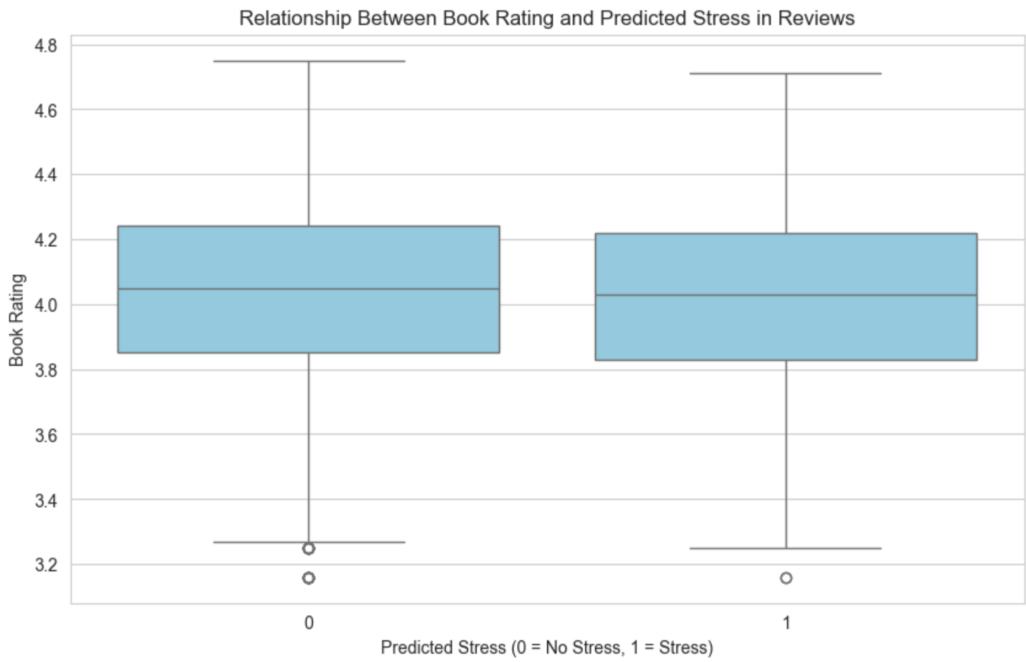
Our initial examination involved a comprehensive correlation analysis between various numerical features of books, including page counts, ratings, and reader engagement metrics (rating_counts, reviews_counts, votes). We visualized these correlations using a heatmap, which revealed moderate relationships between engagement metrics and the incidence of stress. Particularly noteworthy was the correlation between stress predictions and votes, suggesting that books which garnered more votes tended to have a higher incidence of stress within their reviews.

2. Genre-Specific Stress Analysis



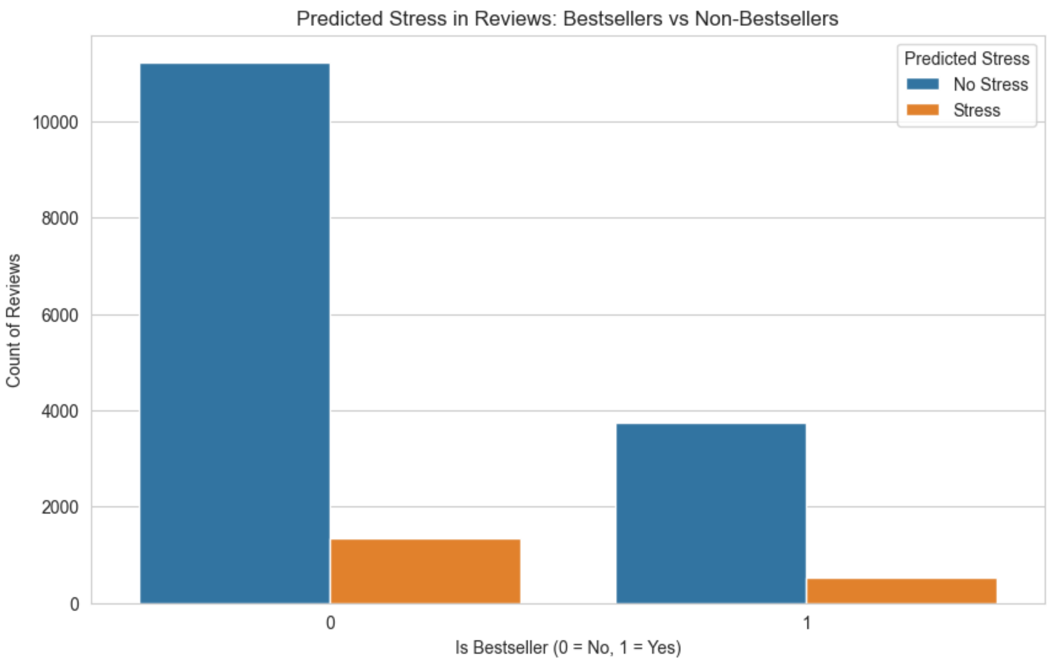
We proceeded to dissect the data further by conducting a genre-specific stress analysis. By averaging the percentage of reviews with stress by book genre, we discovered that genres such as Fantasy/Sci-Fi/Horror and Romance/Drama exhibit a marginally higher presence of stress mentions. This suggests that the content or reader responses to these genres might inherently evoke more stress-related discussions.

3. Rating and Stress Relationship



To interrogate the relationship between book ratings and the predicted stress in reviews, we employed box plots for a visual comparison. The results indicated no significant variance in book ratings regardless of the stress levels predicted in the reviews, underscoring the complexity of how readers' emotional responses might not directly affect their overall rating of a book.

4. Bestsellers and Stress Mentions



Further, we delved into the distribution of predicted stress in reviews, comparing bestsellers with non-bestsellers. Our findings highlighted a distinct pattern; bestsellers had a higher count of reviews predicted as stress-related. This could be interpreted as bestsellers eliciting stronger emotional responses from readers, or perhaps their wider readership naturally leads to a greater volume of stress-related discourse.

Concluding Insights

The advanced visualizations and analyses conducted post-integration have yielded several key insights:

1. Stress mentions in reviews are modestly correlated with reader engagement metrics, hinting at the potential impact of emotional content on reader involvement.

2. Certain genres are more prone to discussions of stress, possibly due to the nature of their content or the emotional investment they demand from readers.

3. The presence of stress in reviews does not appear to adversely affect the overall rating of books, indicating that readers' appreciation of literature transcends momentary emotional states.

4. Bestselling books show a heightened association with stress, which could serve as an indicator of their compelling nature or the deep resonance they find within their audience.

Conclusion

This project has been a comprehensive journey through the realms of data collection, database construction, API development, and data analysis. Leveraging a blend of web scraping techniques, API calls to the New York Times Books API, and open-source datasets, we have successfully curated a rich dataset encompassing a wide array of book information.

In the database development phase, we employed MySQL to establish a robust and relational structure, ensuring the integrity and accessibility of our data. We further enhanced the utility and interaction with our dataset by creating a dedicated book API, “bookapi”, which facilitates the retrieval and manipulation of book-related data, enriching the user experience for potential developers and end-users.

The analytical portion of our project involved meticulous exploratory data analysis (EDA) and advanced visualization techniques. We dove deep into the textual data of social media for stress analysis, employing natural language processing (NLP) to classify and interpret the sentiment within book reviews. Through our model's predictions, we unveiled patterns and correlations that weave together reader engagement metrics and the emotional undertones of stress.

Our findings revealed that certain book genres are more conducive to discussions centered around stress, while the impact of stress mentions on book ratings remained minimal. Notably, bestsellers were more likely to invoke stress-related commentary, perhaps a testament to their evocative nature and broad readership.

In conclusion, our project stands as a testament to the power of data-driven analysis in the literary domain. We have demonstrated that with the right tools and approaches, it is possible to distill meaningful insights from vast textual data. These insights not only shed light on reader sentiment but also offer valuable implications for authors, publishers, and marketers in understanding the complex interplay between a book's content and its reception in the public domain.