

SP22: MGMT ACCESS USE BIG DATA – FINAL PROJECT

Name: Akhila Sakiramolla

Username: asakiram

UID: 2000886005

INTRODUCTION:

One of the world's most influential reading sites, Goodreads offers a forum for those interested in books to discuss them. Book recommendations are driven by word-of-mouth, and Goodreads has amplified the word-of-mouth effect. I'm an avid reader and I have always been interested in how I am being recommended books on various platforms based on my reading history and interests. My aim for this project is to analyze the book's information to understand the user's reading habits and to understand what factors lead to the increasing popularity of certain books. I also implemented a recommendation system which will help users get a list of recommendations based on their previous reads.

Data description:

For this, I analyzed the dataset which contains various information about the books on the website of the world's largest book archive and book proposal site GoodReads. The Goodreads dataset was taken from Kaggle and it was extracted using the Goodreads API to obtain a well-cleaned dataset that only contained features that were deemed promising. The dataset contains information on books by around 6,000 authors and in 27 different languages. It gives more information like the number of pages, the number of ratings, average ratings, publishing date, publisher, etc., The dataset has 12 columns and 11,131 rows. Below is the description of all the variables:

1. **bookID** - A unique Identification number for each book. Dataset has 11,131 unique IDs.
2. **title** - The name under which the book was published. Dataset has 10,353 unique titles.
3. **authors** - Names of the authors of the book. Multiple authors are delimited with -. Dataset has 6,644 unique authors.
4. **average_rating** - The average rating of the book received in total.
5. **isbn** - Another unique number to identify the book, is the International Standard Book Number. Dataset has 11,127 unique isbn codes.
6. **isbn13** - A 13-digit ISBN to identify the book, instead of the standard 11-digit ISBN. Dataset has 11,128 unique isbn13 codes.
7. **language_code** - Helps understand what the primary language of the book is. For instance, eng is standard for English. Dataset has 32 unique language codes.

8. **num_pages** - Number of pages the book contains.
9. **ratings_count** - Total number of ratings the book received.
10. **text_reviews_count** - Total number of written text reviews the book received.
11. **publication_date** - Date on which the book is published.
12. **publisher** - The publisher who published the book. Dataset has 2,295 unique publishers.

BACKGROUND:

A recommender system is a machine learning system that helps users discover new products and services. It guides you every time you shop online towards the most likely product you should purchase. Our digital world is filled with recommender systems because users are often overwhelmed by choice and need assistance finding what they're looking for. Customers are thus happier, which leads to more sales. These systems are like salesmen who know what you like based on your history and preferences. We should view recommender systems not as a way to further increase online sales, but rather as a renewable resource for continuously improving customer insights and our own. So, I wanted to understand how books can be recommended based on users interests and their past reading habits, for example, the ratings a book receives. I aim to get a fair understanding of the relationship between the multiple characteristics a book might have, such as the average rating of each book, the popularity of the authors over the years, and the number of languages it comes in.

I primarily tried to address the following questions with this project:

1. Where do majority of the books lie, in terms of ratings?
2. Is there any relationship between ratings and total ratings given?
3. Do number of pages make an impact on ratings and popularity?
4. Can books be recommended based on ratings?

METHODOLOGY:

In order to develop a recommendation system, I have primarily worked on Google Cloud Platform, as a cloud platform is very instrumental in building efficient and scalable applications. To account for increasing amount of data, I have built this recommendation system in PySpark as it is a distributed computing framework and does efficient real-time, large scale data processing by leveraging parallel computing. To use Python with PySpark framework, I made use of the API calls to Spark. For increased speed on data processing, it is capable of both memory and disk computations. By running on top of Hadoop distributed file systems, it can process structured data.

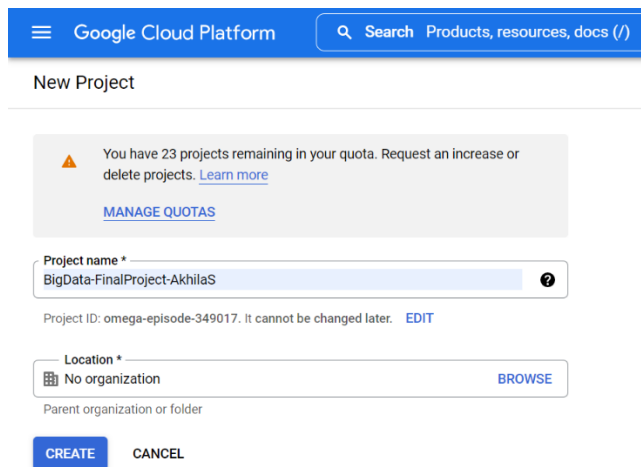
I have followed the following steps to implement this project:

1. Created a project on Google Cloud Platform
2. Collected data and stored it using Buckets on Google Cloud Storage
3. Created Hadoop clusters using Dataproc on Google Cloud Platform
4. Data loading, cleaning and quality assurance techniques using PySpark
5. Data transformation and analysis with visualizations using Python libraries in Jupyter Notebook
6. Built a recommendation system

Below is the detailed description of each of these steps:

1. Project and Hadoop clusters creation

On Google Cloud Platform, I first created a project with the name **BigData-FinalProject-AkhilaS**.



The screenshot shows the 'New Project' form in the Google Cloud Platform console. At the top, there is a blue header with the Google Cloud Platform logo and a search bar. Below the header, the text 'New Project' is displayed. A warning message states: 'You have 23 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)'. Below this, there is a link to 'MANAGE QUOTAS'. The 'Project name' field is filled with 'BigData-FinalProject-AkhilaS'. Below the project name, the 'Project ID' is shown as 'omega-episode-349017', with a note that it cannot be changed later and an 'EDIT' link. The 'Location' field is set to 'No organization', with a 'BROWSE' link. At the bottom, there are 'CREATE' and 'CANCEL' buttons.

2. Data collection and storage

I have used the Goodreads dataset from Kaggle which was extracted using the Goodreads API to obtain a well-cleaned dataset that only contained features that were deemed promising. I created a single CSV file from this data after downloading from Kaggle and uploaded this file in Google Cloud Storage. For this purpose, I created a bucket with the name **bucket-finalproject-akhila** and required configurations on Google Cloud Storage and uploaded the CSV file in it.

Google Cloud Platform BigData-FinalProject-AkhilaS Search Products, resources, docs (/)

Cloud Storage Create a bucket HELP ASSISTANT

- Name your bucket**
Pick a globally unique, permanent name. [Naming guidelines](#)
bucket-finalproject-akhila
Tip: Don't include any sensitive information
LABELS (OPTIONAL)
CONTINUE
- Choose where to store your data**
Location: us (multiple regions in United States)
Location type: Multi-region
- Choose a default storage class for your data**
Default storage class: Standard
- Choose how to control access to objects**
Public access prevention: Off
Access control: Uniform

Good to know

Location pricing
Storage rates vary depending on the storage class of your data and location of your bucket. [Pricing details](#)

Current configuration: Multi-region / Standard

Item	Cost
us (multiple regions in United States)	\$0.026 per GB-month

ESTIMATE YOUR MONTHLY COST

3. Hadoop clusters creation using Dataproc

In order to create a cluster using Dataproc, we first need to create a Virtual Private Cloud (VPC) network which provides connectivity for virtual machine instances including clusters. It is a virtual version of a physical network and also helps in connecting to on-premises networks and distributes traffic from Google cloud.

VPC Network

I created a VPC network with the name **vpc-finalproject-akhilas** with the required configurations as shown below:

Google Cloud Platform BigData-FinalProject-AkhilaS Search vpc networks

VPC network Create a VPC network

Name *
vpc-finalproject-akhilas
Lowercase letters, numbers, hyphens allowed

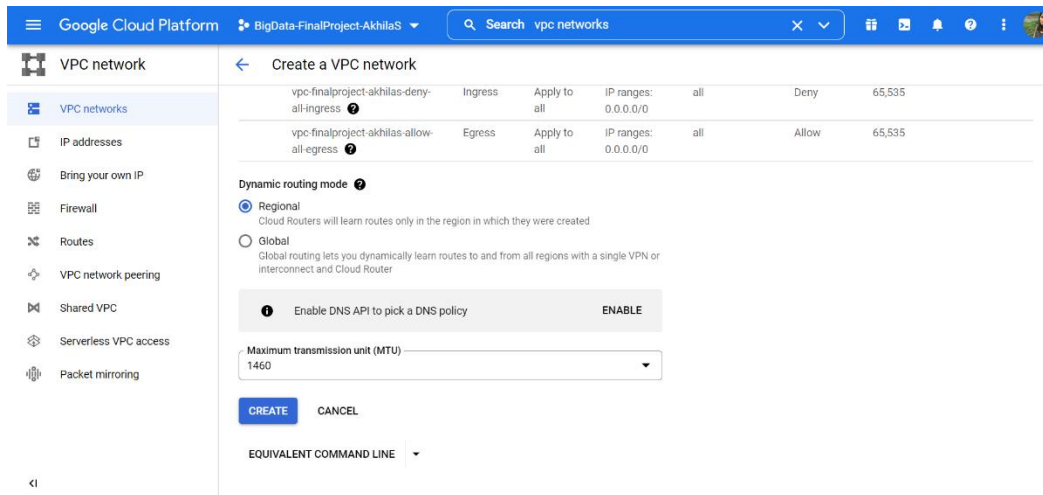
Description

Subnets
Subnets let you create your own private cloud topology within Google Cloud. Click Automatic to create a subnet in each region, or click Custom to manually define the subnets. [Learn more](#)

Subnet creation mode
☐ Custom
☒ Automatic

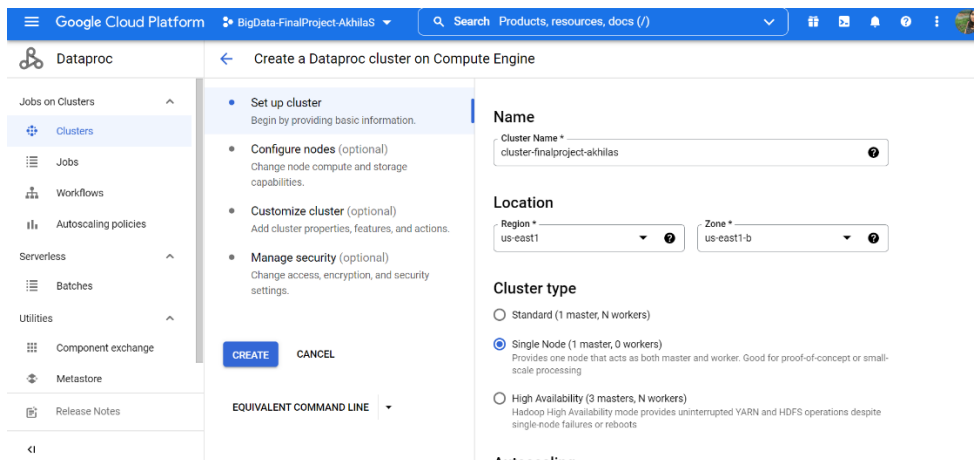
IP stack type
IPv4 (single-stack)

These IP address ranges will be assigned to each region in your VPC



Cluster using Dataproc

I was then able to create a cluster **cluster-finalproject-akhilas** by selecting Single node with 1 master. I enabled component gateway and selected Jupyter Notebook as an optional component. I configured my cluster with the specifications – n1-standard-2 with 2 vCPU and 7.5 GB Memory. I have also customized my cluster by selecting the primary network as the VPC network created (in the previous section) and by selecting the storage stating bucket as the bucket I created (in the earlier section).



Google Cloud Platform

BigData-FinalProject-AkhilAS

Search

Products, resources, docs (/)

Dataproc

Jobs on Clusters

Clusters

Jobs

Workflows

Autoscaling policies

Serverless

Batches

Utilities

Component exchange

Metastore

Release Notes

1

Create a Dataproc cluster on Compute Engine

Set up cluster

Begin by providing basic information.

Configure nodes (optional)

Change node compute and storage capabilities.

Customize cluster (optional)

Add cluster properties, features, and actions.

Manage security (optional)

Change access, encryption, and security settings.

CREATE

CANCEL

EQUIVALENT COMMAND LINE

Components

Component Gateway

☒ Enable component gateway

Provides access to the web interfaces of default and selected optional components on the cluster. [Learn more](#)

Optional components

Select one or multiple components. [Learn more](#)

☐ Anaconda
 ☐ Hive WebHCat
 ☒ Jupyter Notebook
 ☐ Zeppelin Notebook
 ☐ Druid
 ☐ Presto
 ☐ ZooKeeper
 ☐ Ranger
 ☐ HBase
 ☐ Flink

Airplane mode off

inalProject-AkhilAS

Search

Products, resources, docs (/)

Dataproc

Jobs on Clusters

Clusters

Jobs

Workflows

Autoscaling policies

Serverless

Batches

Utilities

Component exchange

Metastore

Release Notes

1

Create a Dataproc cluster on Compute Engine

Set up cluster

Begin by providing basic information.

Configure nodes (optional)

Change node compute and storage capabilities.

Customize cluster (optional)

Add cluster properties, features, and actions.

Manage security (optional)

Change access, encryption, and security settings.

CREATE

CANCEL

EQUIVALENT COMMAND LINE

Master node

Contains the YARN Resource Manager, HDFS NameNode, and all job drivers.

Machine family

GENERAL-PURPOSE

COMPUTE-OPTIMIZED

MEMORY-OPTIMIZED

GPU

Machine types for common workloads, optimized for cost and flexibility

Series

N1

Powered by Intel Skylake CPU platform or one of its predecessors

Machine type

n1-standard-2 (2 vCPU, 7.5 GB memory)

vCPU

2

Memory

7.5 GB

CPU PLATFORM AND GPU

Primary disk size *

500 GB

Primary disk type

Standard Persistent Disk

Google Cloud Platform

BigData-FinalProject-AkhilAS

Search

Products, resources, docs (/)

Dataproc

Jobs on Clusters

Clusters

Jobs

Workflows

Autoscaling policies

Serverless

Batches

Utilities

Component exchange

Metastore

Release Notes

1

Create a Dataproc cluster on Compute Engine

Set up cluster

Begin by providing basic information.

Configure nodes (optional)

Change node compute and storage capabilities.

Customize cluster (optional)

Add cluster properties, features, and actions.

Manage security (optional)

Change access, encryption, and security settings.

CREATE

CANCEL

EQUIVALENT COMMAND LINE

Network Configuration

Establishes connectivity for the VM instances in this cluster.

☒ Networks in this project
 ☐ Networks shared from host project: *

Choose a shared VPC network from project that is different from this cluster's project. [Learn more](#)

Primary network

vpc-finalproject-akhila

Subnetwork

vpc-finalproject-akhila

Network tags

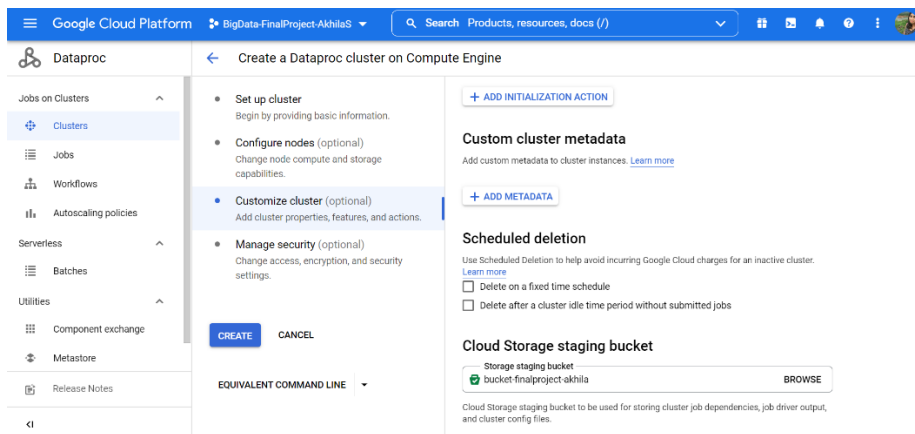
Network tags are text attributes you can add to make firewall rules and routes applicable to specific VM instances.

Internal IP only

☐ Configure all instances to have only internal IP addresses. [Learn more](#)

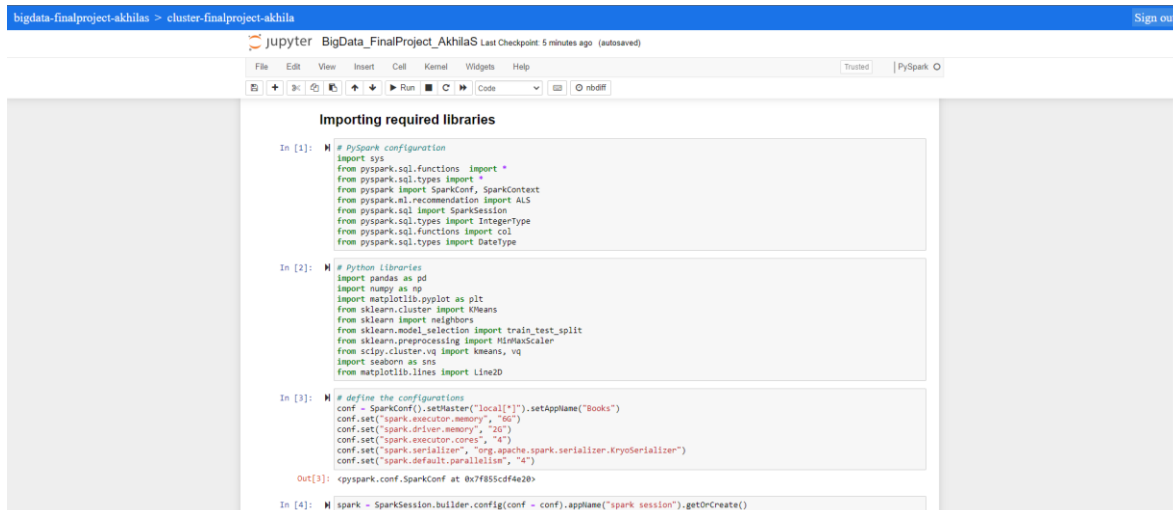
Dataproc Metastore

Configure Dataproc to use Dataproc Metastore as its Hive metastore. [Learn more](#)



4. Data loading, cleaning, and quality assurance techniques

Now that the cluster is up and running, I opened the Jupyter notebook in the cluster from web interfaces section. Created a new folder in local disk folder and created a new PySpark notebook in this folder. I imported all the required libraries for both PySpark and Python. Read the data stored in a bucket in Google Cloud Storage and did quality assurance checks like checking for duplicate values, null values, and outliers.



Loading the data

```
In [14]: # Loading the data and viewing it

books = spark.read.load("gs://bucket-finalproject-akhila/books.csv",
                        format = 'csv',
                        sep = ',',
                        header = 'true',
                        inferSchema = 'true').cache()

books.show(3)
```

22/05/02 21:31:43 WARN org.apache.spark.sql.execution.CacheManager: Asked to cache already cached data.

bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date	publisher
1	Harry Potter and ...	J.K. Rowling/Mary...	4.57	439785960	9.78E12	eng	652	2095690			
2	Harry Potter and ...	J.K. Rowling/Mary...	4.49	439358078	9.78E12	eng	870	2153167			
4	Harry Potter and ...	J.K. Rowling	4.42	439554896	9.78E12	eng	352	6333			

only showing top 3 rows

```
In [15]: print((books.count(), len(books.columns)))

(11127, 12)
```

```
In [16]: books.printSchema()

root
 |-- bookID: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- authors: string (nullable = true)
 |-- average_rating: double (nullable = true)
 |-- isbn: string (nullable = true)
 |-- isbn13: double (nullable = true)
 |-- language_code: string (nullable = true)
 |-- num_pages: integer (nullable = true)
 |-- ratings_count: integer (nullable = true)
 |-- text_reviews_count: integer (nullable = true)
 |-- publication_date: string (nullable = true)
 |-- publisher: string (nullable = true)
```

Checking for null values

```
In [31]: books.select([count(when(iscn(c) | col(c).isNull(), c)).alias(c) for c in books.columns]).show()
```

bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date	publisher
0	0	0	0	0	0	0	0	0	0	0	0

Checking for duplicate values

```
In [32]: distinct_books = books.distinct()
print("Distinct count: "+str(distinct_books.count()))

Distinct count: 11127

In [33]: distinct_books1 = books.dropDuplicates()
print("Distinct count: "+str(distinct_books1.count()))

Distinct count: 11127
```

We observe that there are no null values or duplicate records in the data.

Data summary

```
In [34]: from pyspark.sql.functions import format_number
result = books.select(["average_rating", "num_pages", "ratings_count", "text_reviews_count"]).describe()
result.select(result['summary'],
               format_number(result['average_rating'].cast('float'), 2).alias('average_rating'),
               result['num_pages'].cast('int').alias('num_pages'),
               result['ratings_count'].cast('int').alias('ratings_count'),
               result['text_reviews_count'].cast('int').alias('text_reviews_count')
               ).show()
```

summary	average_rating	num_pages	ratings_count	text_reviews_count
count	11,127.00	11127	11127	11127
mean	3.93	336	17936	541
stddev	0.35	241	112479	2576
min	0.00	0	0	0
max	5.00	6576	4597666	94265

```
In [36]: books.select("bookID").distinct().count()
```

```
Out[36]: 11127
```

```
In [37]: books.select("title").distinct().count()
```

```
Out[37]: 10352
```

```
In [38]: books.select("authors").distinct().count()
```

```
Out[38]: 6643
```

```
In [39]: books.select("language_code").distinct().count()
```

```
Out[39]: 27
```

```
In [40]: books.select("publisher").distinct().count()
```

```
Out[40]: 2292
```

5. Data transformation and analysis with visualizations

To visualize the data, I converted the PySpark dataframe to Pandas dataframe. After doing that I noticed that the data type of data is object, so I changed that to date. I then extracted **year** from the **publication date** for further analysis.

Data preprocessing

```
In [41]: books1 = books.toPandas()
df_books = pd.DataFrame(books1)
df_books.head(2)
```

```
Out[41]:
```

	bookID	title	authors	average_rating	isbn	isbn13	language_code	num_pages	ratings_count	text_reviews_count	publication_date
0	1	Harry Potter and the Half-Blood Prince (Harry ...	J.K. Rowling/Mary GrandPré	4.57	439785960	9.7800000e+12	eng	652	2095690	27591	9/16/2006
1	2	Harry Potter and the Order of the Phoenix (Har...	J.K. Rowling/Mary GrandPré	4.49	439358078	9.7800000e+12	eng	870	2153167	29221	9/1/2004

```

In [42]: df_books.dtypes

Out[42]: bookID            int32
title              object
authors            object
average_rating     float64
isbn              object
isbn13            float64
language_code      object
num_pages          int32
ratings_count      int32
text_reviews_count int32
publication_date   object
publisher          object
dtype: object

In [43]: # Changing the data type of publication date from object to date
df_books['publication_date'] = pd.to_datetime(df_books['publication_date'], errors='coerce')
df_books['Year'] = df_books['publication_date'].dt.year

In [44]: df_books['publication_date'].min()

Out[44]: Timestamp('1900-01-01 00:00:00')

In [45]: df_books['publication_date'].max()

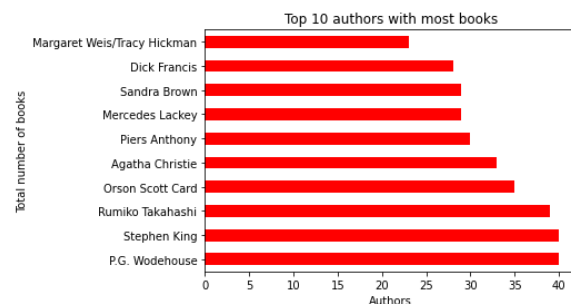
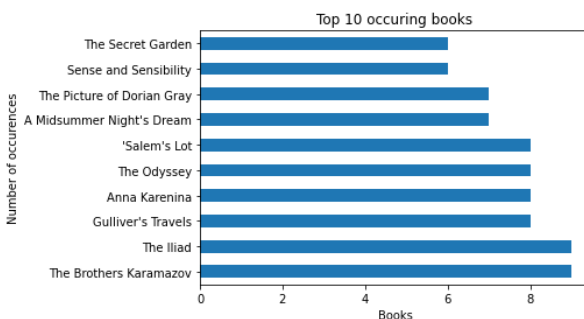
Out[45]: Timestamp('2020-03-31 00:00:00')

```

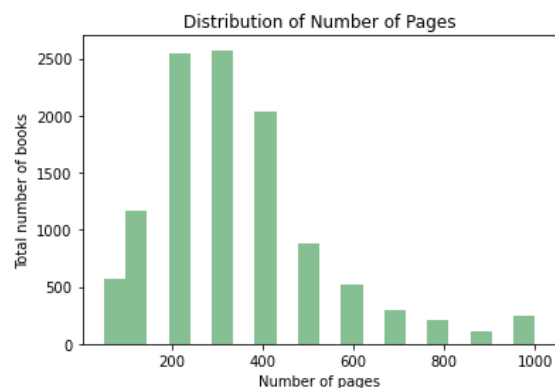
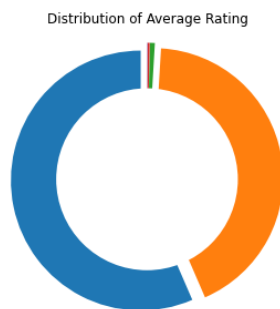
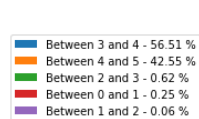
I initially did some preliminary data analysis to understand the variables and their distributions. Then I did bivariate and multivariate analyses to understand the relationship between different variables. I then used K-Means clustering to find groups between **average rating** and **ratings counts**. Using this I was able to classify the whole data into 5 clusters, which were then used to build a recommendation system.

RESULTS:

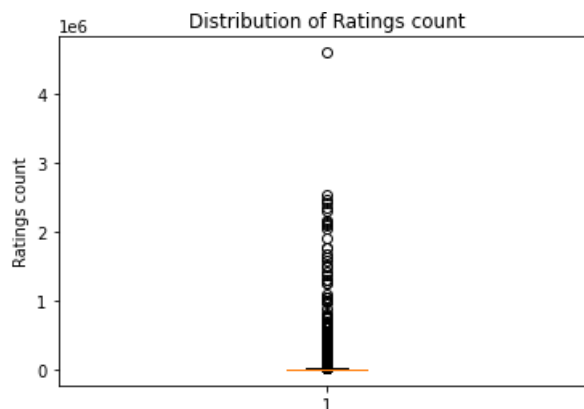
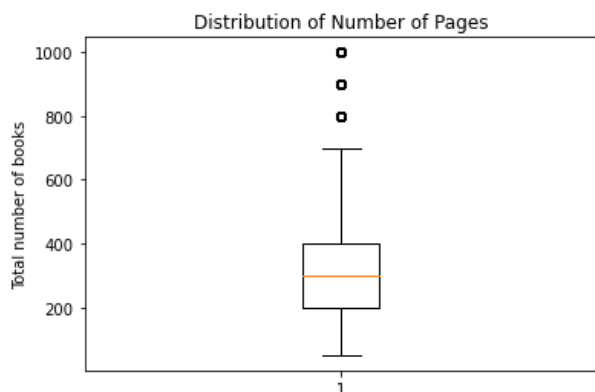
I have done Exploratory Data Analysis to obtain the following visualizations and interpretations:



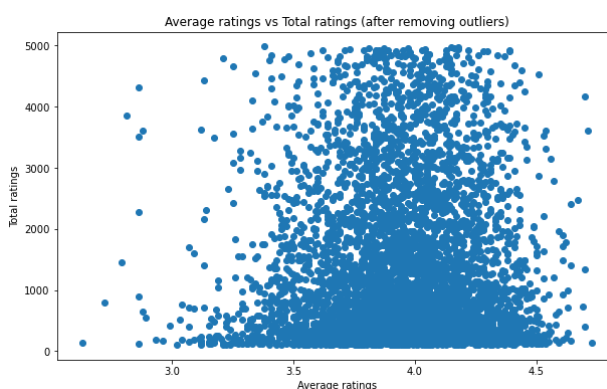
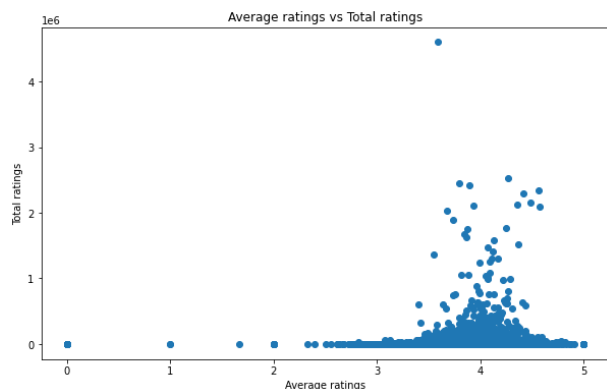
1. We can see that "The Iliad" and "The Brothers Karamazov" have the greatest number of concurrences with the same name in the data. Various editions of these books have been found repeatedly in this database.
2. We can that Rumiko Takahashi has the most number of books in the list. Several of them might be just different editions of the same book, considering that his work has been around for a long time, spanning decades. The names on the list again indicate that most of the authors have either written for decades.



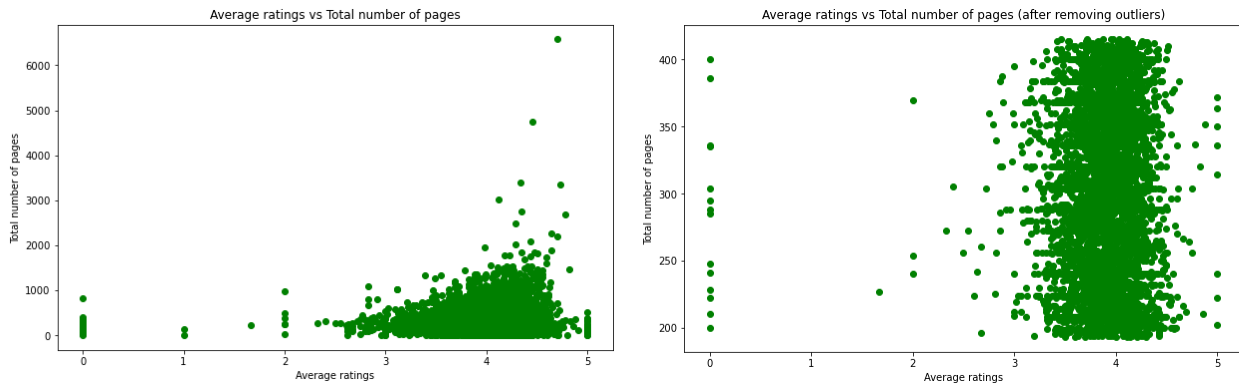
- From the plot, we can infer that the majority of the ratings are between 3 and 4 and there are very few books with a rating of 5.
- As expected, we can observe from the above right skewed histogram that there are a significant number of data points (perhaps outliers) that are greater than the mode.



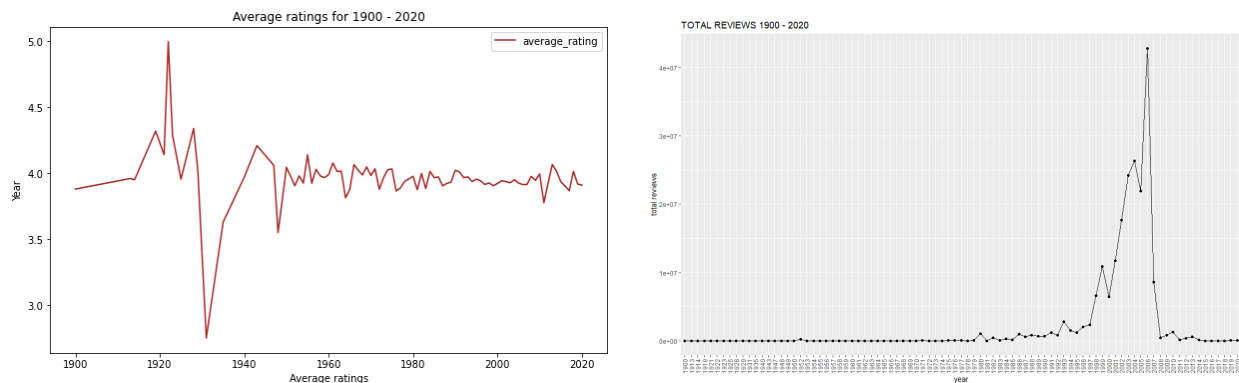
- We can observe a significant number of outliers between 750 and 1000 pages, which suggests that these outliers may be too important to remove from the data set before data analysis.
- We can observe that due to the large number of massive outliers, it will be necessary to include outlier analysis when exploring questions related to ratings counts.



7. We observe that there may be a potential relationship between the average rating and ratings count. As the number of ratings increase, the rating for the book seems to move towards 4. The average rating seems to become sparse while the ratings count keeps on decreasing.



8. We observe that books with the page number range of 200 – 400 have the highest rating, peaking near 250. It may be due to the fact that people seem to prefer books with a moderate number of pages.

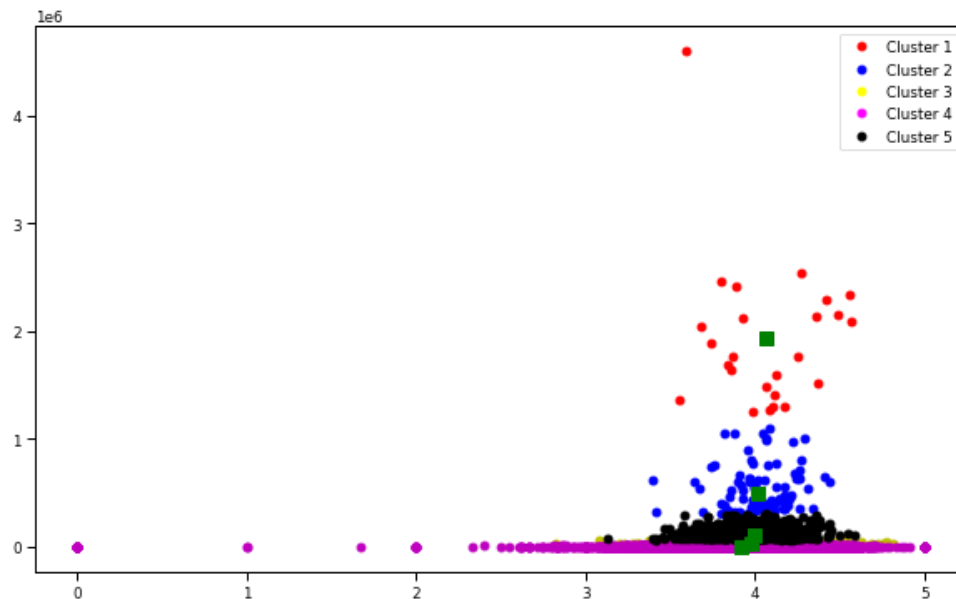


9. I plotted a general distribution of average ratings and there was a sharp dip in what is seen as early 1930s. This may be attributed to the Great Depression with lesser people with the ability to spend and review books.
10. We notice that after 2000, the number of ratings were increasing and there's a very sharp dip during 2008, maybe due to the recession.

Recommendation system:

I implemented K-Means clustering algorithm to identify groups between average rating and ratings count. By using the elbow curve method, I found that the optimal number of clusters is 5. In this way I was able to cluster all the books into 5 clusters. I observed that as the count increases, the rating would end up near the clusters given above. The green squares represent the

centroids of the clusters. As the number of ratings decreases, the average rating appears to become sparser, with higher volatility and less accuracy.



K Nearest Neighbors is the algorithm used in the recommendation system. In response to a book entered by the user, the nearest neighbors would be classified as the books the user may like. KNN can be used for both classification and regression. To predict the label of an instance in classification problems, we first find k closest instances to it based on the distance metric and then we predict the label based on either a majority voting scheme or weighted majority voting scheme.

Finding similar books - recommendations

```
In [95]: print_similar_books("The Catcher in the Rye")
```

```
Angels & Demons (Robert Langdon #1)
Animal Farm
Lord of the Flies
Romeo and Juliet
Of Mice and Men
```

```
In [96]: get_id_from_partial_name("Harry Potter and the ")
```

```
Harry Potter and the Half-Blood Prince (Harry Potter #6) 0
Harry Potter and the Order of the Phoenix (Harry Potter #5) 1
Harry Potter and the Chamber of Secrets (Harry Potter #2) 2
Harry Potter and the Prisoner of Azkaban (Harry Potter #3) 3
Harry Potter and the Half-Blood Prince (Harry Potter #6) 0
Harry Potter and the Prisoner of Azkaban (Harry Potter #3) 3
Harry Potter and the Chamber of Secrets (Harry Potter #2) 2
Harry Potter and the Sorcerer's Stone (Harry Potter #1) 8875
Harry Potter and the Philosopher's Stone (Harry Potter #1) 10677
Harry Potter and the Goblet of Fire (Harry Potter #4) 10678
```

```
In [97]: print_similar_books(id = 1) #ID for the Book 5
```

```
Harry Potter and the Half-Blood Prince (Harry Potter #6)
The Fellowship of the Ring (The Lord of the Rings #1)
Harry Potter and the Chamber of Secrets (Harry Potter #2)
Harry Potter and the Prisoner of Azkaban (Harry Potter #3)
The Hobbit or There and Back Again
```

DISCUSSION:

Interpretation of the results:

Below is the interpretation of the results obtained in the previous section:

- I observed that there is a relationship between the average rating and ratings count, as number of ratings increase, rating of books seems to move towards 4 and as number decreases, the rating becomes sparse.
- I observed that readers prefer books with moderate range of pages (around 200 – 400) as highest number of ratings are given in that range.
- I also discovered that authors whose books are highly rated are Stephen King, P.G. Wodehouse, Rumiko Takahashi, Orson Scott Card, Agatha Christie, etc.
- I found that the most highly rated publishers are so due to having a limited number of highly satisfied readers.
- I also observe that the authors with highest ratings have a smaller number of total reviews, a smaller number of books published in the years and vice versa.
- The quantity of reader ratings has very less effect on the number of text reviews posted on the Goodreads website. Even if the number of reviews for a book is high, the ratings for that book may be poor.
- I was able to provide 5 recommendations for a given book (The Catcher in the Rye) based on the ratings distribution using the recommendation system.

Implementation of skills from the course:

Implementing this project helped me strengthen my cloud technology skills like **Google Cloud Platform** and how to create **clusters** in it using **Dataprocc**. I also got a good grasp of **PySpark** which I have used to load the data and do quality assurance checks. Using **Hadoop** clusters, I was able to leverage its multi-processing power and understood the importance of parallel processing. The modules **Distributed Computing and File Systems** and **Processing and Analytics** have been very helpful for me to get a good understanding of all of these concepts. I also learnt how to create **virtual machine** instances and run **jupyter notebook** in it. All these concepts I believe are very important and will help me immensely in any future projects I take up.

Issues encountered:

One of the major issues I faced during the course of this project was when I tried to create a cluster using Dataprocc. I was unable to do it initially as I faced an issue with the primary network connection. I hadn't created one before creating the cluster. After going through the Qwicklabs tutorials, I was able to figure out that creating a Virtual Private Cloud (VPC) network as my default network will resolve this issue. I also faced some issues with few dependencies in Jupyter

Notebook. I was able to resolve this by downloading and installing the libraries from PySpark and Python official documentations.

CONCLUSION:

I always felt a recommendation system is of major importance for any e-commerce website to retain customers and improve customer experience. I always wanted to implement one and for this project, I was able to leverage many big data tools and technologies to implement a Book Recommendation System. I was able to create an end-to-end pipeline, right from data collection, storage, analysis and recommendation system. I was able to strengthen my skills in various aspects like virtualization, parallel processing, data visualization and machine learning modelling. Despite a few issues, I was able to resolve them and reach the end goal of finishing the project without losing focus on learning the above-mentioned tools and technologies. Taking this course further helped me to explore the real-world data and deal with a large amount of data.

REFERENCES:

<https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks>

<https://cloud.google.com/dataproc/docs/tutorials/jupyter-notebook>

<https://cloud.google.com/dataproc/docs/concepts/components/jupyter#console>

<https://cloud.google.com/vpc/docs/create-modify-vpc-networks>

[Book Recommendation System | Build A Book Recommendation System \(analyticsvidhya.com\)](#)

<https://www.kaggle.com/code/hoshi7/goodreads-analysis-and-recommending-books#Background-checkup>.