**MINI PROJECT REPORT**

**DATA SCIENCE & BIG DTA ANALYTICS**

**SUBMITTED BY**

**[RA1711003010172 – SUPARNO BHATTA]**

**UNDER THE GUIDANCE OF**

**MRS.D. VIJI**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KARRANKULATHUR - 603203**

**DECLARATION**

I, Suparno Bhatta, RA 1711003010172, studying III year B.Tech in Computer Science and Engineering at SRM Institute of Science and Technology, Kattankulathur, Chennai, hereby declare that this Mini project is an original work of mine and I have not verbatim copied / duplicated any material from sources like internet or from print media, excepting some vital company information / statistics and data that is provided by the Technical organisations itself.

Signature of the Student

Date: 14 October, 2019

Place: SRM IST, KATTANKULATHUR

**ACKNOWLEDGEMENT**

It is a matter of great pleasure and privilege for me to present this report of as a project for the partial fulfilment of the course: Data Science and Big Data Analytics. Through this report, I would like to thank numerous people whose consistent support and guidance has been the standing pillar in architecture of this report.

I would like to thank my Faculty for Data Science and Big Data Analytics, Mrs. D. Viji who gave me this opportunity to go out and gain the very valuable experiences. Also thanks to HOD CSE ma’am, Dr. B. Amutha, who continuously supported me in every possible way.

SUPARNO BHATTA

**ABSTRACT**

**Goodreads E-book Prediction System**

The basic idea behind analysing the Goodreads dataset is to get a fair idea about the relationships between the multiple attributes a book might have, such as the aggregate rating of each book, the trend of the authors over the years and books with numerous languages. With over a hundred thousand ratings, there are books which just tend to become popular as each day seems to pass. We have always considered the magical persona books seem to hold, and with this notebook, we step out on a journey to see what kind of books really drives people to read in this era of modern smart devices.

With such a vast, overwhelming number of factors, we will go over such demographics:

-Does any relationship lie between ratings and the total ratings given?

-Where do majority of the books lie, in terms of ratings - Does reading a book really bring

forth bias for the ratings?

-Do authors tend to perform same over time, with all their newer books? Or do they just

fizzle out.

-Do number of pages make an impact on reading styles, ratings and popularity?

-Can books be recommended based on ratings? Is that a factor which can work? - by

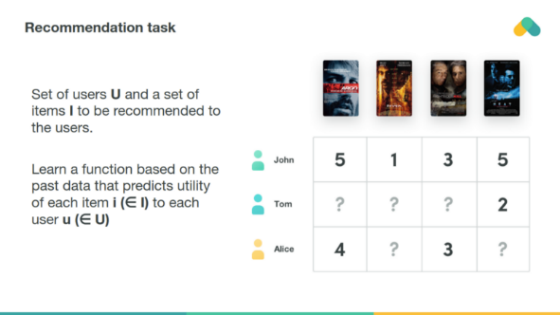
Kmeans

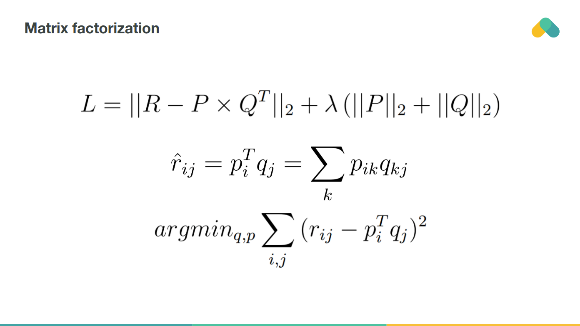
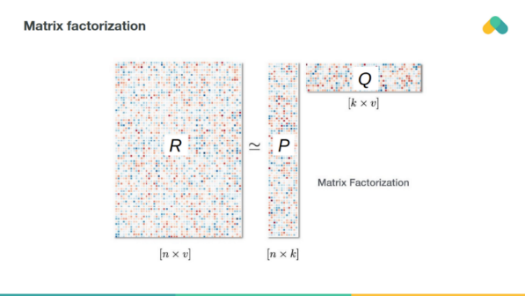
**TABLE OF CONTENTS**

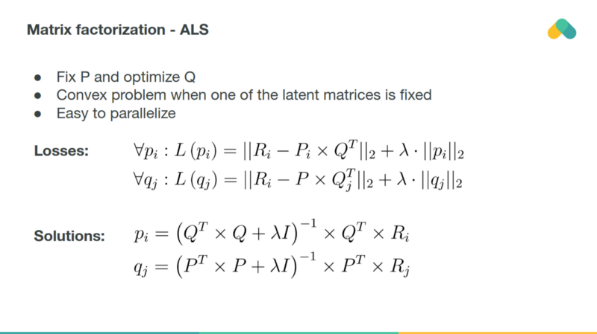
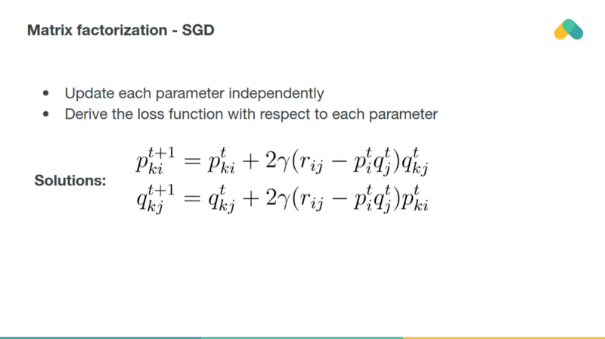
|  |  |  |
| --- | --- | --- |
| Serial Number | Topic | Page Number |
| 1. | INTRODUCTION | 1-3 |
| 2. | REVIEW OF LITERATURE | 4 |
| 3. | RESEARCH METHODOLOGY | 5 |
| 4. | DATA ANALYSIS | 6-13 |
| 5. | INTERPRETATION & FINDINGS | 14.15 |
| 6. | SUGGESTION & CONCLUSIONS | 16 |
| 7. | LIST OF TABLES | 17 |
| 8. | LIST OF FIGURES / GRAPHS | 18-22 |
| 9. | LIST OF ANNEXURE | 23-29 |
| 10. | BIBLIOGRAPHY | 30 |

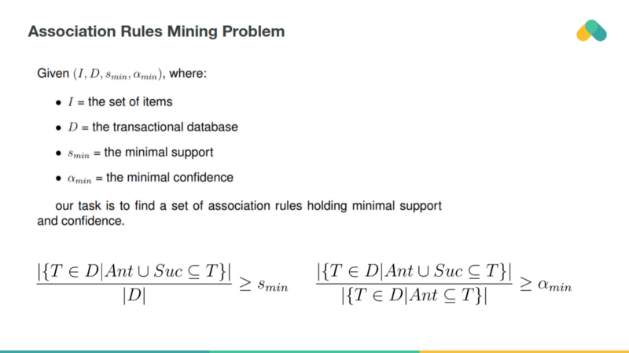
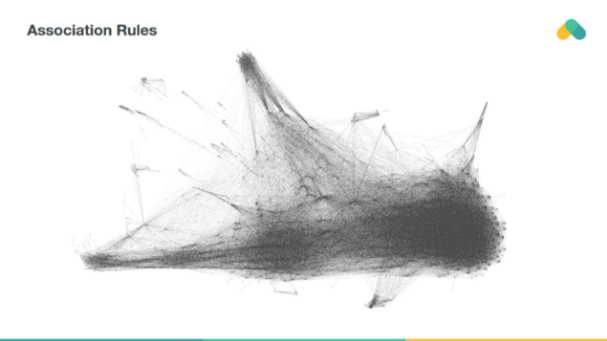
**INTRODUCTION**

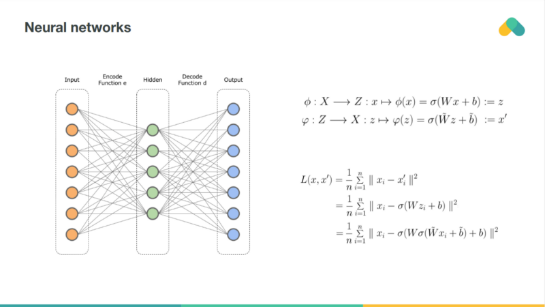
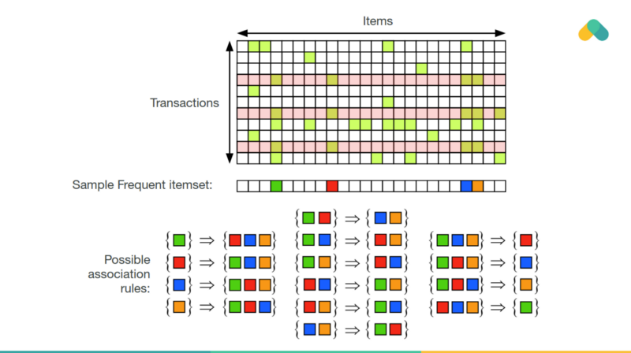
Recommender systems are one of the most successful and widespread application of machine learning technologies in business. There were many people on waiting list that could not attend our[MLMU talk](https://www.meetup.com/Prague-Machine-Learning/events/250915214/) so I am sharing slides and comments here. We can find large scale recommender systems in [retail](http://rejoiner.com/resources/amazon-recommendations-secret-selling-online/),[video on demand](https://research.netflix.com/research-area/recommendations), or[music streaming](https://www.theverge.com/tldr/2018/2/5/16974194/spotify-recommendation-algorithm-playlist-hack-nelson). In order to develop and maintain such systems, a company typically needs a group of expensive data scientist and engineers. That is why even large corporates such as BBC decided to [outsource](https://www.broadbandtvnews.com/2018/01/04/thinkanalytics-wins-bbc-personalisation-contract/) its recommendation services. Machine learning algorithms in recommender systems are typically classified into two categories — content based and collaborative filtering methods although modern recommenders combine both approaches. Content based methods are based on similarity of item attributes and collaborative methods calculate similarity from interactions. Below we discuss mostly collaborative methods enabling users to discover new content dissimilar to items viewed in the past.

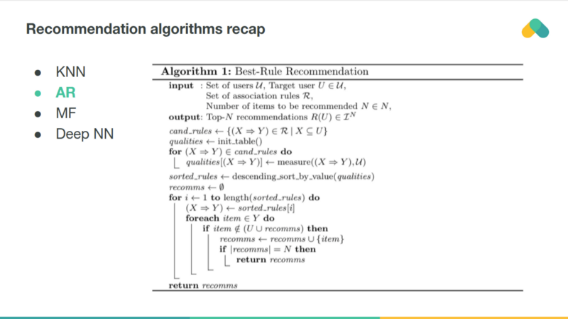
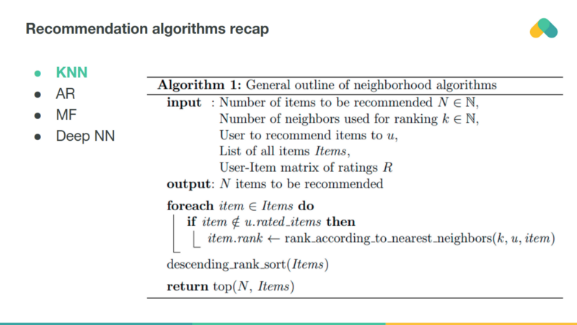


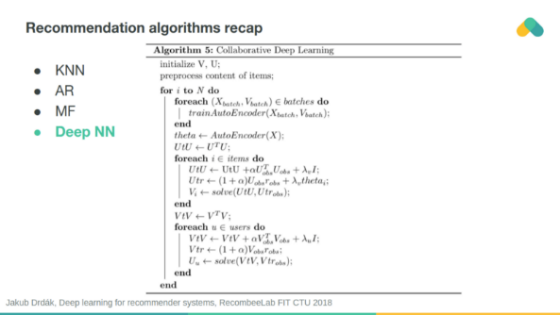
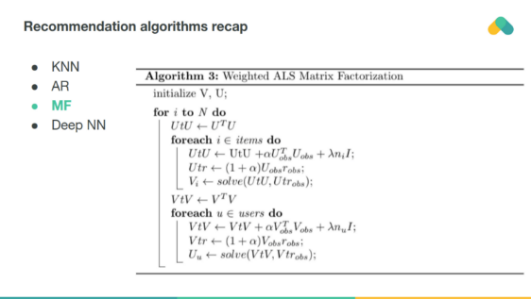


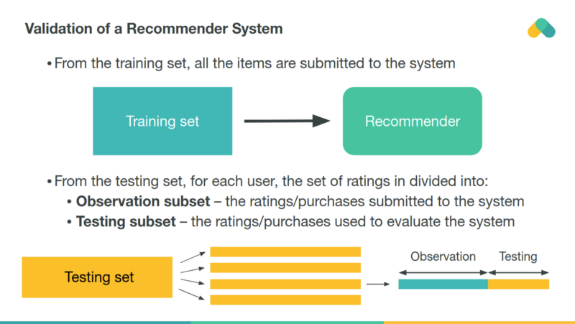
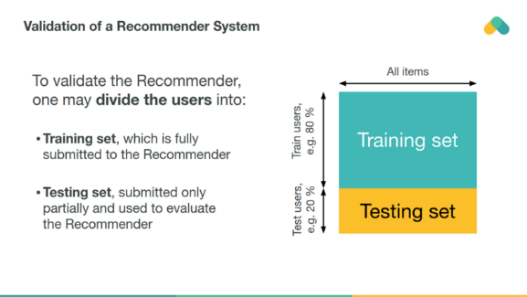


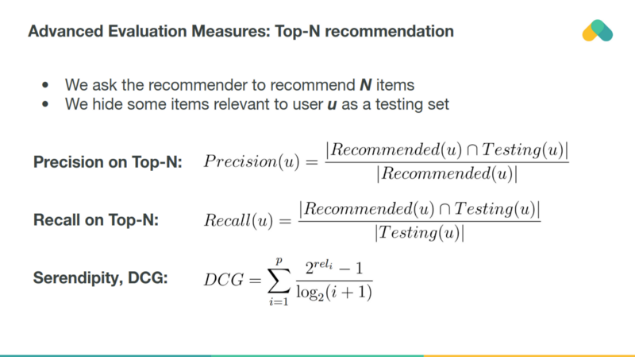
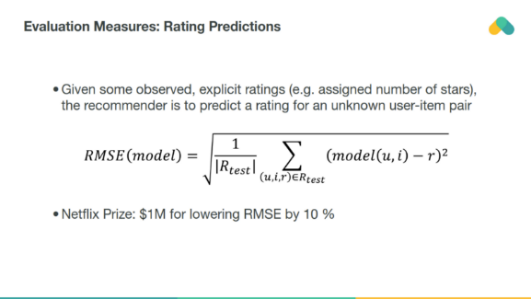


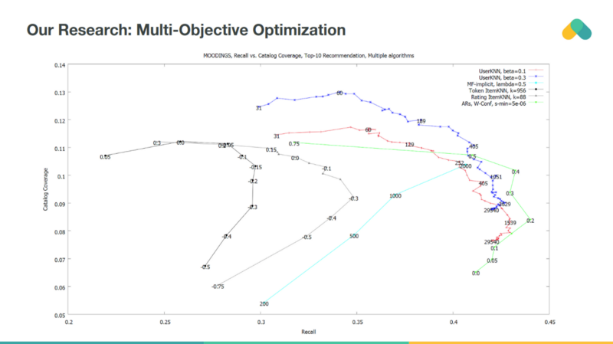
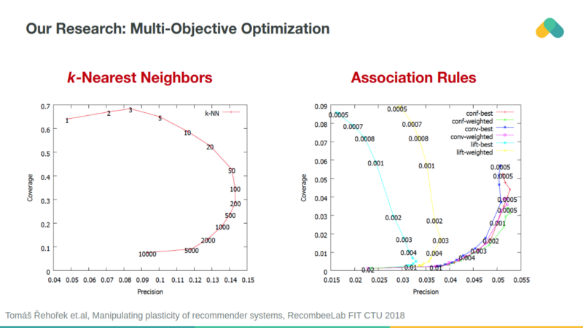


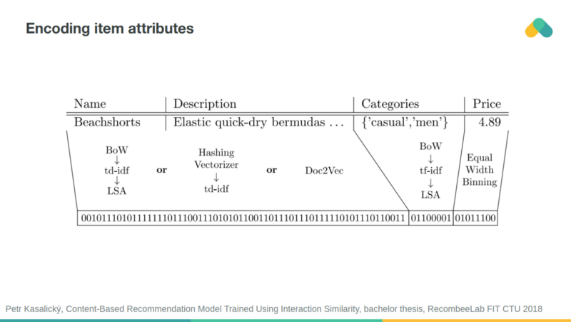
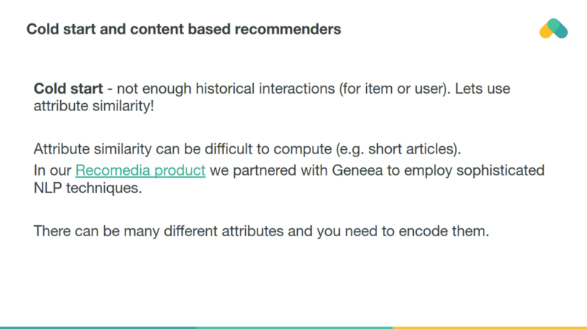


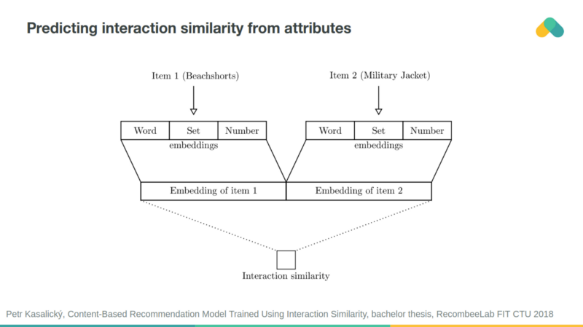












**REVIEW OF LITERATURE**

**Machine Learning Model** The aim of this project is to come up with a model for predicting the book ratings. It used linear regression to build a model that predicts book ratings. Linear regression algorithm is a basic predictive analytics technique. There are two kinds of variables in a linear regression model:

1. The **input** or **predictor variable** is the variable(s) that help predict the value of the output variable. It is commonly referred to as **X**.
2. The **output variable** is the variable that we want to predict. It is commonly referred to as **Y**.

It divided the data into attributes and labels.

Attributes are the independent variables whilst labels are dependent variables whose values are to be predicted. Then they split 80% of the data to the training set and 20% of the data to test set. test\_size variable is where we actually specify the proportion of the test set. Now to train our algorithm, they imported LinearRegression class instantiate it, and call the fit() method along with the training data.

In the next step they to use the test data to check accurately our algorithm predicts the percentage score. Now it was compared the actual output values for X\_test with the predicted values. Then they visualized the comparison result.

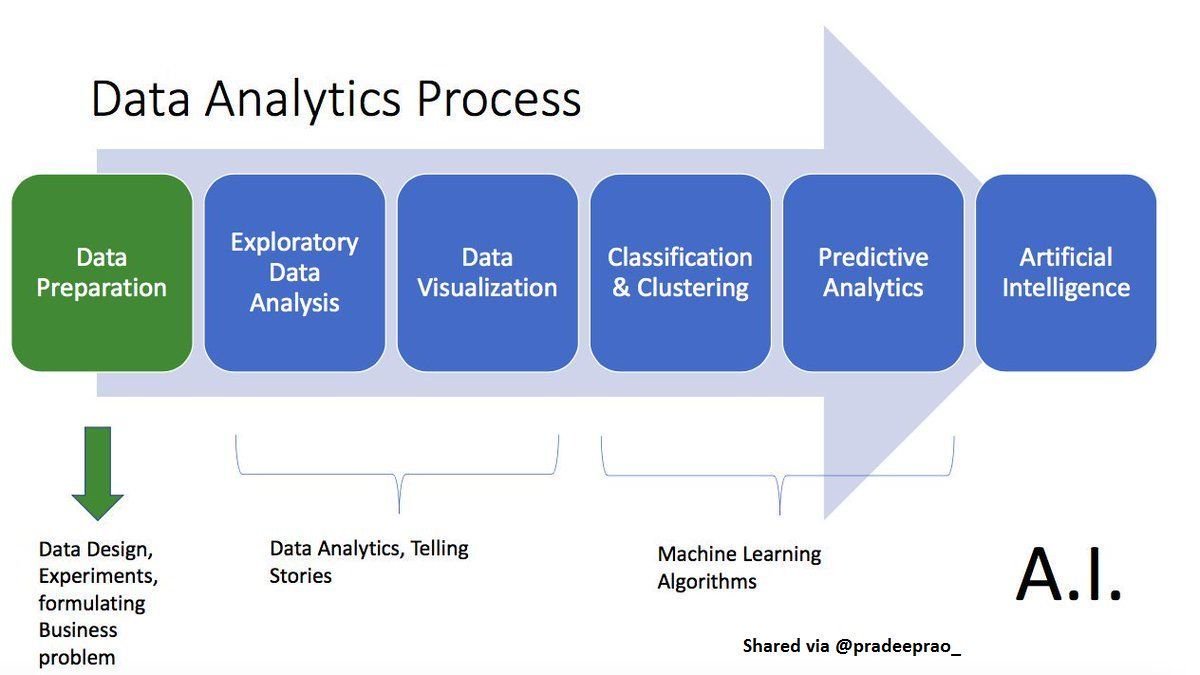
Though the model is not very precise, the predicted percentages are close to the actual ones. Then they evaluated the performance of the algorithm using MAE, MSE & RMSE.

MAE: 0.22617011110589422

MSE: 0.10979819213369937

RMSE: 0.33135810256231757

**RESEARCH METHODOLOGY**



**DATA ANALYSIS**

1. Installation of libraries

*!pip install isbnlib*

*!pip install newspaper3k*

*!pip install goodreads\_api\_client*

*import numpy as np*

*import pandas as pd*

*import os*

*import seaborn as sns*

*import isbnlib*

*from newspaper import Article*

*import matplotlib.pyplot as plt*

*plt.style.use('ggplot')*

*from tqdm import tqdm*

*from progressbar import ProgressBar*

*import re*

*from scipy.cluster.vq import kmeans, vq*

*from pylab import plot, show*

*from matplotlib.lines import Line2D*

*import matplotlib.colors as mcolors*

*import goodreads\_api\_client as gr*

*from sklearn.cluster import KMeans*

*from sklearn import neighbors*

*from sklearn.model\_selection import train\_test\_split*

*from sklearn.preprocessing import MinMaxScaler*

*import warnings*

*warnings.filterwarnings("ignore")*

1. Fetching data

*df = pd.read\_csv("/books.csv")*

*df.index = df['bookID']*

*#Finding Number of rows and columns*

*print("Dataset contains {} rows and {} columns".format(df.shape[0], df.shape[1]))*

*df.head()*

Columns Description: **bookID** Contains the unique ID for each book/series **title** contains the titles of the books **authors** contains the author of the particular book **average**\_**rating** the average rating of the books, as decided by the users **ISBN** ISBN(10) number, tells the information about a book - such as edition and publisher **ISBN** **13.** The new format for ISBN, implemented in 2007. 13 digits **language\_code** Tells the language for the books **Num\_pages** Contains the number of pages for the book **Ratings\_count** Contains the number of ratings given for the book **text\_reviews\_count** Has the count of reviews left by users.

**3.** **Exploratory Data Analysis**

**3.1.** Which are the books with most occurances in the list?

*#Taking the first 20:*

*sns.set\_context('poster')*

*plt.figure(figsize=(10,10))*

*books = df['title'].value\_counts()[:20]*

*rating = df.average\_rating[:20]*

*sns.barplot(x = books, y = books.index, palette='deep')*

*plt.title("Most Occurring Books")*

*plt.xlabel("Number of occurances")*

*plt.ylabel("Books")*

*plt.show()*

We can see that One Hundred Years Of Solitude and Salem's List have the most number of occurrances with the same name in the data.

These books have come up in this database over and over again, with various publication editions. From the list, we can see that most of the books from the given chart are either old, steadfast classics or books which are usually assigned to schools. Seems like some books do age well, and these have just braved the flow of time.

**3.2.** What is the distribution of books for all languages?

*sns.set\_context('paper')*

*plt.figure(figsize=(20,5))*

*ax = df.groupby('language\_code')['title'].count().plot.bar()*

*plt.title('Language Code')*

*plt.xticks(fontsize = 20)*

*for p in ax.patches:*

*ax.annotate(str(p.get\_height()), (p.get\_x()-0.3, p.get\_height()+100))*

From the given graph, we can infer that in the given data, majority of the books are in english languages, with some further categorised into English-US, english-UK and english-CA.

**3.3.** Which are the top 10 most rated books?

*most\_rated = df.sort\_values('ratings\_count', ascending = False).head(10).set\_index('title')*

*plt.figure(figsize=(15,5))*

*sns.barplot(most\_rated['ratings\_count'], most\_rated.index, palette='rocket')*

1. We can see that the beginning books of the series usually have most of the ratings, i.e, Harry Potter and the Sorcerer's stone, Twilight #1, The Hobbit, Angels and demons #1.
2. Harry potter's first book dominates the section by having more than 5000000 ratings. Infact, apart from a few, such as Catcher in the Rye and Animal Farm, all of the books seem to be from a series of books, getting the notion into our head that once people begin, most of them seem to dive in with the notion of completing it.

Yet, when we glance at the first and fifth book of Harry Potter, we can also notice that there has been a ridiculously huge margin in the number of readers/ratings for the books, signifying that there were people who did not pick up the next book in the series and/or only found the first book to touch their hearts up to an extent to drop a vote.

**3.4.** Which are the authors with most books?

*sns.set\_context('talk')*

*most\_books = df.groupby('authors')['title'].count().reset\_index().sort\_values('title', ascending=False).head(10).set\_index('authors')*

*plt.figure(figsize=(15,5))*

*ax = sns.barplot(most\_books['title'], most\_books.index, palette='icefire\_r')*

*ax.set\_title("Top 10 authors with most books")*

*ax.set\_xlabel("Total number of books")*

*for i in ax.patches:*

*ax.text(i.get\_width()+.3, i.get\_y()+0.5, str(round(i.get\_width())), fontsize = 10, color = 'k')*

We can see from the above plot that Agatha Christie has the most number of books in the list - although a lot of them might be just various publications for the same book, considering the fact that her work has been here for quite a while, spanning decades.

From the names in the list, we can again gather that most of the authors have either been writing for decades, churning numerous books from time to time, or are authors who are regaled as the 'classics' in our history. It seems, hype does play a role in this.

**3.5.** Which are the top 10 highly rated authors?

*high\_rated\_author = df[df['average\_rating']>=4.3]*

*high\_rated\_author = high\_rated\_author.groupby('authors')['title'].count().reset\_index().sort\_values('title', ascending = False).head(10).set\_index('authors')*

*plt.figure(figsize=(15,5))*

*ax = sns.barplot(high\_rated\_author['title'], high\_rated\_author.index, palette='Set2')*

*ax.set\_xlabel("Number of Books")*

*ax.set\_ylabel("Authors")*

*for i in ax.patches:*

*ax.text(i.get\_width()+.3, i.get\_y()+0.5, str(round(i.get\_width())), fontsize = 10, color = 'k')*

We can see that JJR Tolkien, Bill Waterson and JK Rowling are the highest rated authors.

**3.6.** What is the rating distribution for the books?

*def segregation(data):*

*values = []*

*for val in data.average\_rating:*

*if val>=0 and val<=1:*

*values.append("Between 0 and 1")*

*elif val>1 and val<=2:*

*values.append("Between 1 and 2")*

*elif val>2 and val<=3:*

*values.append("Between 2 and 3")*

*elif val>3 and val<=4:*

*values.append("Between 3 and 4")*

*elif val>4 and val<=5:*

*values.append("Between 4 and 5")*

*else:*

*values.append("NaN")*

*print(len(values))*

*return values*

*df.average\_rating.isnull().value\_counts()*

*df.dropna(0, inplace=True) #Removing Any null values*

*plt.figure(figsize=(10,3))*

*rating= df.average\_rating.astype(float)*

*sns.distplot(rating, bins=20)*

Majority of the ratings lie near 3.7 to 4.3, approximately. Books having scores near 5 are extremely rare.

**3.7.** Is there any relationship between ratings and review counts?

*#Checking for any relation between them.*

*plt.figure(figsize=(20,10))*

*df.dropna(0, inplace=True)*

*sns.set\_context('paper')*

*ax =sns.jointplot(x="average\_rating",y='text\_reviews\_count', kind='scatter', data= df[['text\_reviews\_count', 'average\_rating']])*

*ax.set\_axis\_labels("Average Rating", "Text Review Count")*

*plt.show()*

We can infer from the plot that most of the ratings for the books seems to lie near 3-4, with a heavy amount of reviews lying barely near 5000*.*

***3.8.***Is there a relationship between number of pages and ratings?

*plt.figure(figsize=(15,10))*

*sns.set\_context('paper')*

*ax = sns.jointplot(x="average\_rating", y="# num\_pages", data = df, color = 'crimson')*

*ax.set\_axis\_labels("Average Rating", "Number of Pages")*

This plot doesn't give that much of an accurate inference due to the massive presence of outliers for books above 1000 pages, for the maximum density is between 0-1000 pages.

**3.9**. Is there a relationship between ratings and ratings count?

*trial = df[~(df.ratings\_count>2000000)]*

*sns.set\_context('paper')*

*ax = sns.jointplot(x="average\_rating", y="ratings\_count", data = trial, color = 'brown')*

*ax.set\_axis\_labels("Average Rating", "Ratings Count")*

From the graph, we can see that there can be a potential relationship between the average rating and ratings count. As the number of ratings increase, the rating for the book seems to taper towards 4. The average rating seems to become sparse while the number keeps on decreasing.

**3.10.** Which are the books with the highest reviews?

*most\_text = df.sort\_values('text\_reviews\_count', ascending = False).head(10).set\_index('title')*

*plt.figure(figsize=(20,10))*

*sns.set\_context('poster')*

*ax = sns.barplot(most\_text['text\_reviews\_count'], most\_text.index, palette='magma')*

*for i in ax.patches:*

*ax.text(i.get\_width()+2, i.get\_y()+0.5,str(round(i.get\_width())), fontsize=15,color='black')*

*plt.show()*

From all the above inferences, we can fundamentally decide that although the reviews matter, there can't be any specific relation between them and the ranking for all the books.

**3.10.** Topic Modelling

KMeans Clustering without outliers

KMeans clustering is a type of unsupervised learning which groups unlabelled data. The goal is to find groups in data.

With this, I attepmt to find a relationship or groups between the rating count and average rating value.

*trial = df[['average\_rating', 'ratings\_count']]*

*data = np.asarray([np.asarray(trial['average\_rating']), np.asarray(trial['ratings\_count'])]).T*

I'll use the Elbow Curve method for the best way of finding the number of clusters for the data.

*X = data*

*distortions = []*

*for k in range(2,30):*

*k\_means = KMeans(n\_clusters = k)*

*k\_means.fit(X)*

*distortions.append(k\_means.inertia\_)*

*fig = plt.figure(figsize=(15,5))*

*plt.plot(range(2,30), distortions, 'bx-')*

*plt.title("Elbow Curve")*

From the above plot, we can see that the elbow lies around the value K=5, so that's what we will attempt it with.

*#Computing K means with K = 5, thus, taking it as 5 clusters*

*centroids, \_ = kmeans(data, 5)*

*#assigning each sample to a cluster*

*#Vector Quantisation:*

*idx, \_ = vq(data, centroids)*

*# some plotting using numpy's logical indexing*

*sns.set\_context('paper')*

*plt.figure(figsize=(15,10))*

*plt.plot(data[idx==0,0],data[idx==0,1],'or',#red circles*

*data[idx==1,0],data[idx==1,1],'ob',#blue circles*

*data[idx==2,0],data[idx==2,1],'oy', #yellow circles*

*data[idx==3,0],data[idx==3,1],'om', #magenta circles*

*data[idx==4,0],data[idx==4,1],'ok',#black circles*

*)*

*plt.plot(centroids[:,0],centroids[:,1],'sg',markersize=8, )*

*circle1 = Line2D(range(1), range(1), color = 'red', linewidth = 0, marker= 'o', markerfacecolor='red')*

*circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker= 'o', markerfacecolor='blue')*

*circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0, marker= 'o', markerfacecolor='yellow')*

*circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker= 'o', markerfacecolor='magenta')*

*circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker= 'o', markerfacecolor='black')*

*plt.legend((circle1, circle2, circle3, circle4, circle5)*

*, ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5'), numpoints = 1, loc = 0, )*

*plt.show()*

We can see from the above plot, that because of two outliers, the whole clustering algortihm is skewed.

KMeans with optimisation Finding the outliers and then removing them-

*trial.idxmax()*

*trial.drop(3, inplace = True)*

*trial.drop(41865, inplace = True)*

*data = np.asarray([np.asarray(trial['average\_rating']), np.asarray(trial['ratings\_count'])]).T*

*#Computing K means with K = 8, thus, taking it as 8 clusters*

*centroids, \_ = kmeans(data, 5)*

*#assigning each sample to a cluster*

*#Vector Quantisation:*

*idx, \_ = vq(data, centroids)*

*# some plotting using numpy's logical indexing*

*sns.set\_context('paper')*

*plt.figure(figsize=(15,10))*

*plt.plot(data[idx==0,0],data[idx==0,1],'or',#red circles*

*data[idx==1,0],data[idx==1,1],'ob',#blue circles*

*data[idx==2,0],data[idx==2,1],'oy', #yellow circles*

*data[idx==3,0],data[idx==3,1],'om', #magenta circles*

*data[idx==4,0],data[idx==4,1],'ok',#black circles*

*)*

*plt.plot(centroids[:,0],centroids[:,1],'sg',markersize=8, )*

*circle1 = Line2D(range(1), range(1), color = 'red', linewidth = 0, marker= 'o', markerfacecolor='red')*

*circle2 = Line2D(range(1), range(1), color = 'blue', linewidth = 0,marker= 'o', markerfacecolor='blue')*

*circle3 = Line2D(range(1), range(1), color = 'yellow',linewidth=0, marker= 'o', markerfacecolor='yellow')*

*circle4 = Line2D(range(1), range(1), color = 'magenta', linewidth=0,marker= 'o', markerfacecolor='magenta')*

*circle5 = Line2D(range(1), range(1), color = 'black', linewidth = 0,marker= 'o', markerfacecolor='black')*

*plt.legend((circle1, circle2, circle3, circle4, circle5)*

*, ('Cluster 1','Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5'), numpoints = 1, loc = 0, )*

*plt.show()*

From the above plot, now we can see that once the whole system can be classified into clusters. As the count increases, the rating would end up near the cluster given above. The green squares are the centroids for the given clusters.

As the rating count seems to decrease, the average rating seems to become sparser, with higher volatility and less accuracy.

**3.11.** Recommendation Engine

Having seen the clustering, we can infer that there can be some recommendations which can happen with the relation between Average Rating and Ratings Count.

Taking the Ratings\_Distribution (A self created classifying trend), the recommendation system works with the algortihm of K Nearest Neighbors.

Based on a book entered by the user, the nearest neighbours to it would be classified as the books which the user might like.

KNN is used for both classification and regression problems. In classification problems to predict the label of a instance we first find k closest instances to the given one based on the distance metric and based on the majority voting scheme or weighted majority voting(neighbors which are closer are weighted higher) we predict the labels.

*books\_features = pd.concat([df['Ratings\_Dist'].str.get\_dummies(sep=","), df['average\_rating'], df['ratings\_count']], axis=1)*

*min\_max\_scaler = MinMaxScaler()*

*books\_features = min\_max\_scaler.fit\_transform(books\_features)*

*model = neighbors.NearestNeighbors(n\_neighbors=6, algorithm='ball\_tree')*

*model.fit(books\_features)*

*distance, indices = model.kneighbors(books\_features)*

*def get\_index\_from\_name(name):*

*return df[df["title"]==name].index.tolist()[0]*

*all\_books\_names = list(df.title.values)*

*def get\_id\_from\_partial\_name(partial):*

*for name in all\_books\_names:*

*if partial in name:*

*print(name,all\_books\_names.index(name))*

*def print\_similar\_books(query=None,id=None):*

*if id:*

*for id in indices[id][1:]:*

*print(df.iloc[id]["title"])*

*if query:*

*found\_id = get\_index\_from\_name(query)*

*for id in indices[found\_id][1:]:*

*print(df.iloc[id]["title"])*

In a setting such as this, the unsupervised learning takes place, with the similar neighbors being recommended. For the given list, if I ask recommendations for "The Catcher in the Rye", five books related to it would appear.

Creating a book features table, based on the Ratings Distribution, which classifies the books into ratings scale such as:

Between 0 and 1 Between 1 and 2 Between 2 and 3 Between 3 and 4 Between 4 and 5 Broadly, the recommendations then consider the average ratings and ratings cout for the query entered.

**INTERPRETATION & FINDINGS**

1. We can see that One Hundred Years Of Solitude and Salem's List have the most number of occurances with the same name in the data.
2. We can see that most of the books from the given chart are either old, steadfast classics or books which are usually assigned to schools. Seems like some books do age well, and these have just braved the flow of time.
3. We can infer that in the given data, majority of the books are in English languages, with some further categorised into English-US, English-UK and English-CA.
4. We can see that the beginning books of the series usually have most of the ratings, that is Harry Potter and the Sorcerer's stone, Twilight #1, The Hobbit, Angels and demons #1.
5. Harry potter's first book dominates the section by having more than 5000000 ratings. In fact, apart from a few, such as Catcher in the Rye and Animal Farm, all of the books seem to be from a series of books, getting the notion into our head that once people begin, most of them seem to dive in with the notion of completing it.
6. We glance at the first and fifth book of Harry Potter, we can also notice that there has been a ridiculously huge margin in the number of readers/ratings for the books, signifying that there were people who did not pick up the next book in the series and/or only found the first book to touch their hearts up to an extent to drop a vote.
7. We can see from the above plot that Agatha Christie has the most number of books in the list - although a lot of them might be just various publications for the same book, considering the fact that her work has been here for quite a while, spanning decades.
8. We can again gather that most of the authors have either been writing for decades, churning numerous books from time to time, or are authors who are regaled as the 'classics' in our history. It seems, hype does play a role in this.
9. We can see that JJR Tolkien, Bill Waterson and JK Rowling are the highest rated authors.
10. Majority of the ratings lie near 3.7 to 4.3, approximately. Books having scores near 5 are extremely rare.
11. We can infer from the plot that most of the ratings for the books seems to lie near 3-4, with a heavy amount of reviews lying barely near 5000*.*
12. We can see that there can be a potential relationship between the average rating and ratings count. As the number of ratings increase, the rating for the book seems to taper towards 4. The average rating seems to become sparse while the number keeps on decreasing.
13. We can fundamentally decide that although the reviews matter, there can't be any specific relation between them and the ranking for all the books.
14. We can see from the above plot, that because of two outliers, the whole clustering algortihm is skewed.
15. Having seen the clustering, we can infer that there can be some recommendations which can happen with the relation between Average Rating and Ratings Count.
16. Taking the Ratings\_Distribution (A self created classifying trend), the recommendation system works with the algortihm of K Nearest Neighbors.
17. Based on a book entered by the user, the nearest neighbours to it would be classified as the books which the user might like.
18. KNN is used for both classification and regression problems. In classification problems to predict the label of a instance we first find k closest instances to the given one based on the distance metric and based on the majority voting scheme or weighted majority voting (neighbors which are closer are weighted higher) we predict the labels.
19. In a setting such as this, the unsupervised learning takes place, with the similar neighbors being recommended. For the given list, if I ask recommendations for "The Catcher in the Rye", five books related to it would appear.
20. Creating a book features table, based on the Ratings Distribution, which classifies the books into ratings scale such as: Between 0 and 1 Between 1 and 2 Between 2 and 3 Between 3 and 4 Between 4 and 5 Broadly, the recommendations then consider the average ratings and ratings cout for the query entered.

**SUGGESTION & CONCLUSIONS**

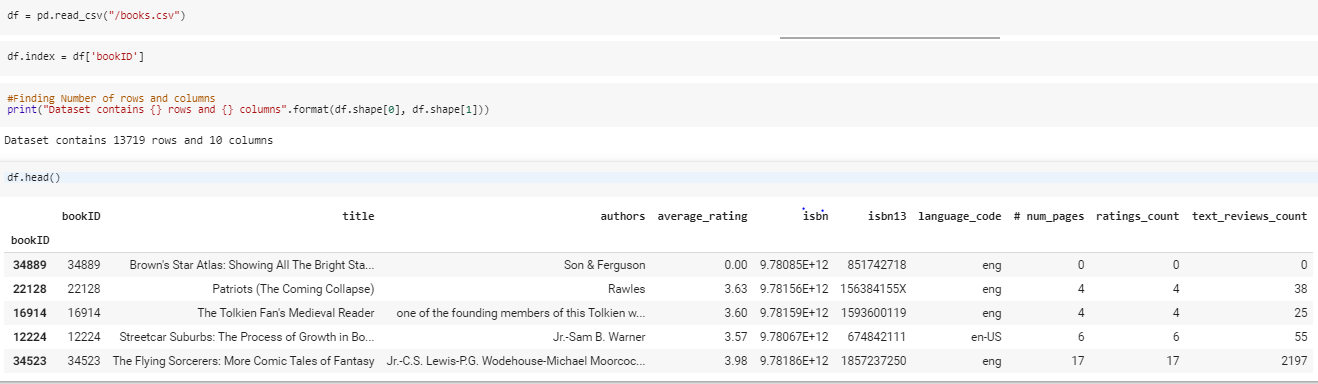
Overall, this project incorporated in the course of Data Science and Big Data Analytics has helped me enhance and develop my skills, abilities, and knowledge. I really recommend this course to my fellow friends. It was a wonderful experience working with about 13200 book data over 12 attributes.

The K-Means algorithm I used in this prediction system will help anyone to get a precise suggestion on the books they would love to read in future by the last book or the preferred type of book they have read lately.

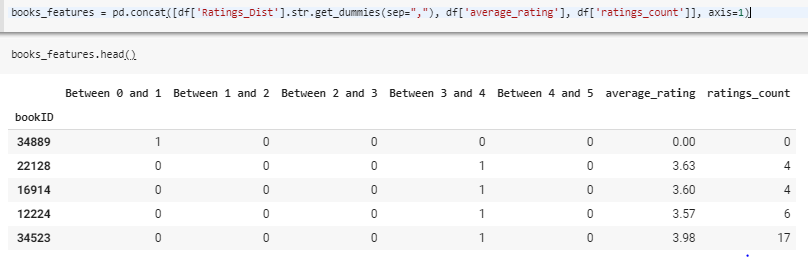
This project is dedicated to all the book readers as it helps them in its business case scenario.

“I do believe something magical can happen when you read a good book” – J.K. Rowling

**LIST OF TABLES**



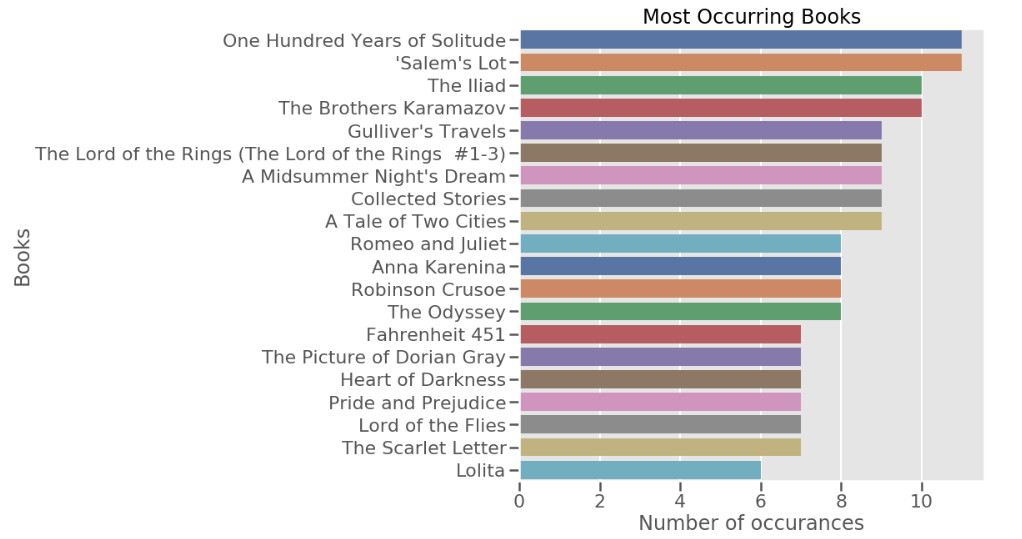
Dataset



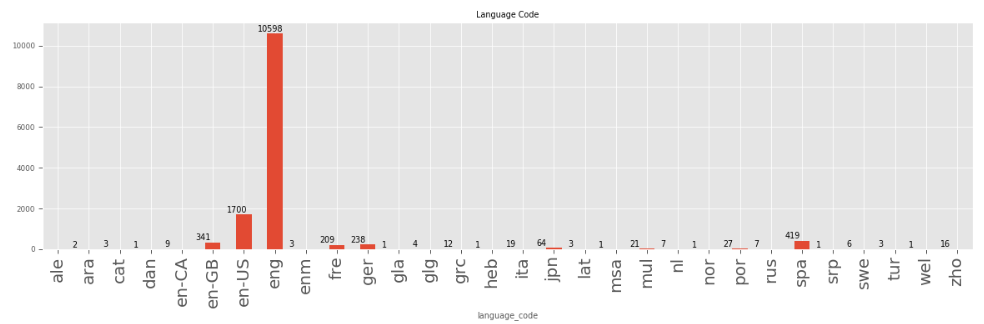
Books Features Table

**LIST OF FIGURES / GRAPHS**

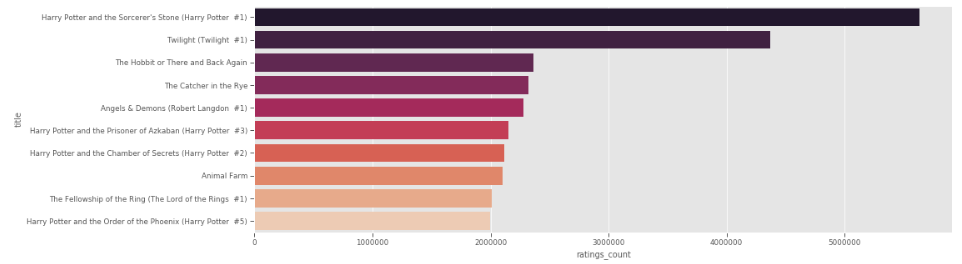
**1.**



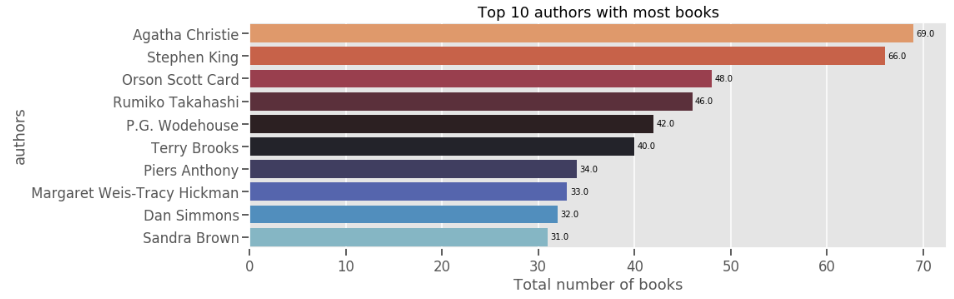
**2.**



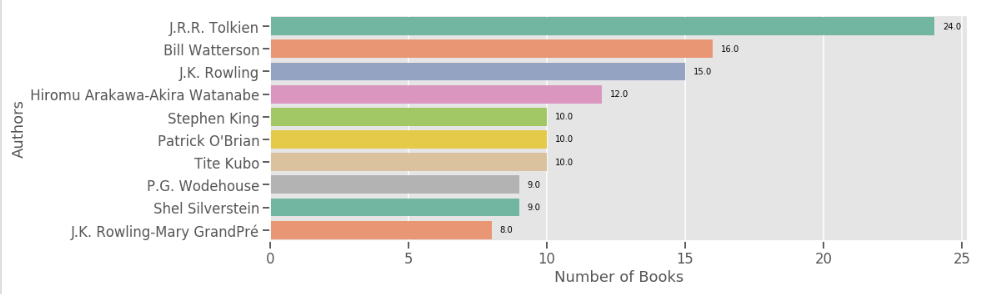
**3.**



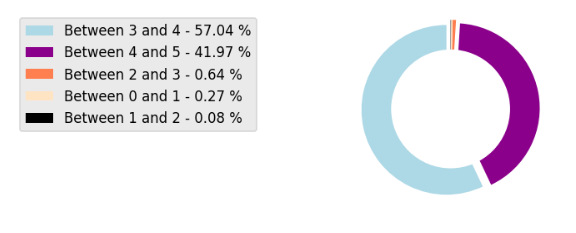
**4.**



**5.**

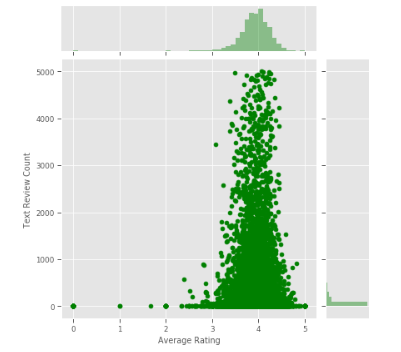


**6.**

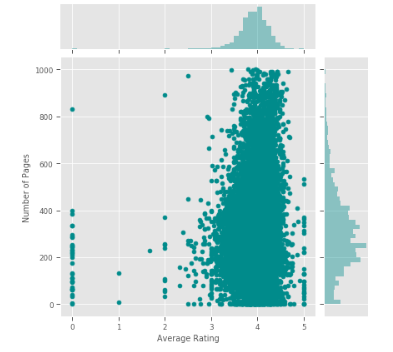


Rating %

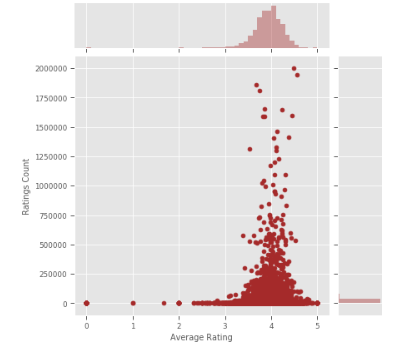
**7.**



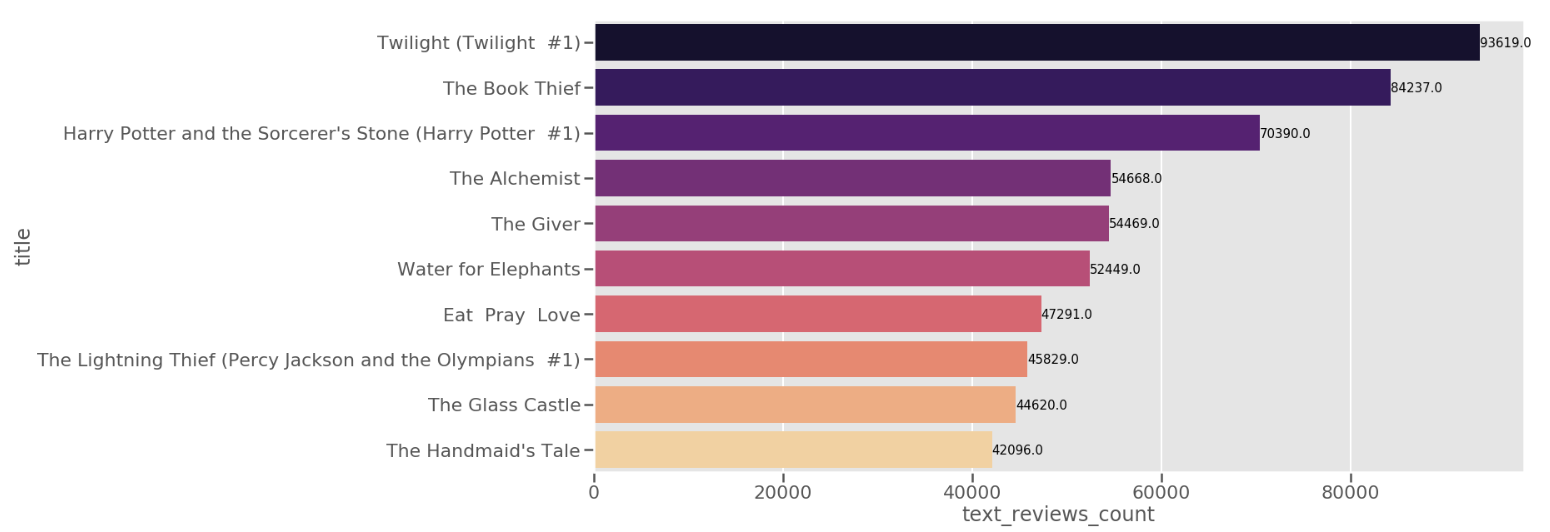
**8.**



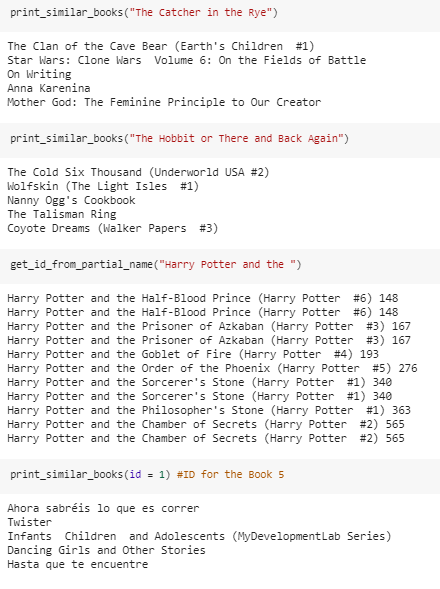
**9.**



**10.**

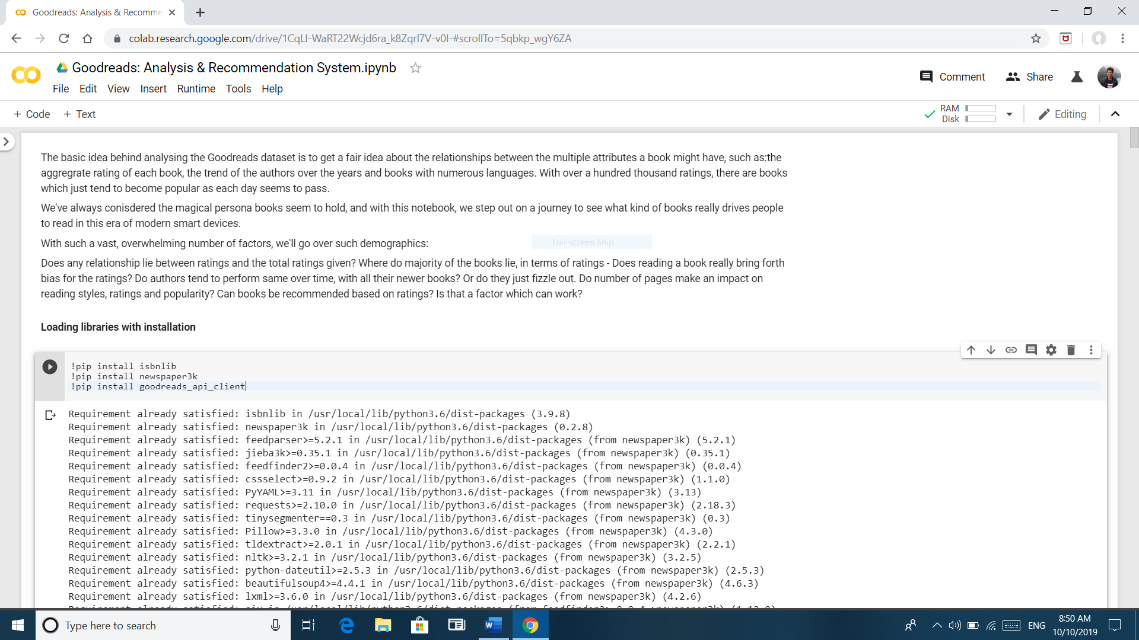


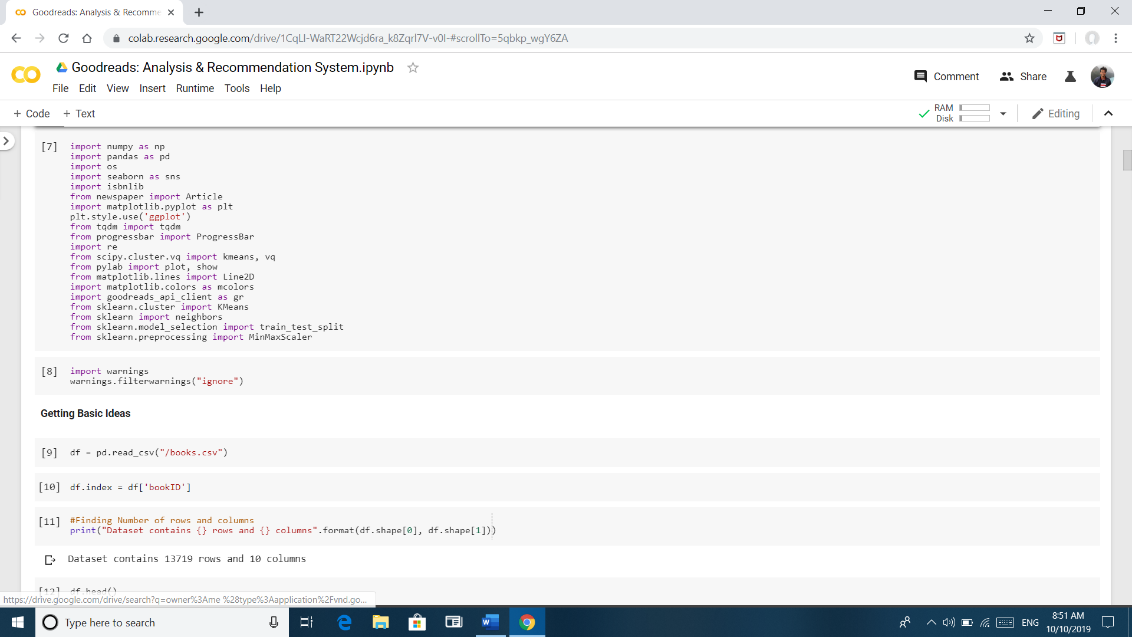
**11.**



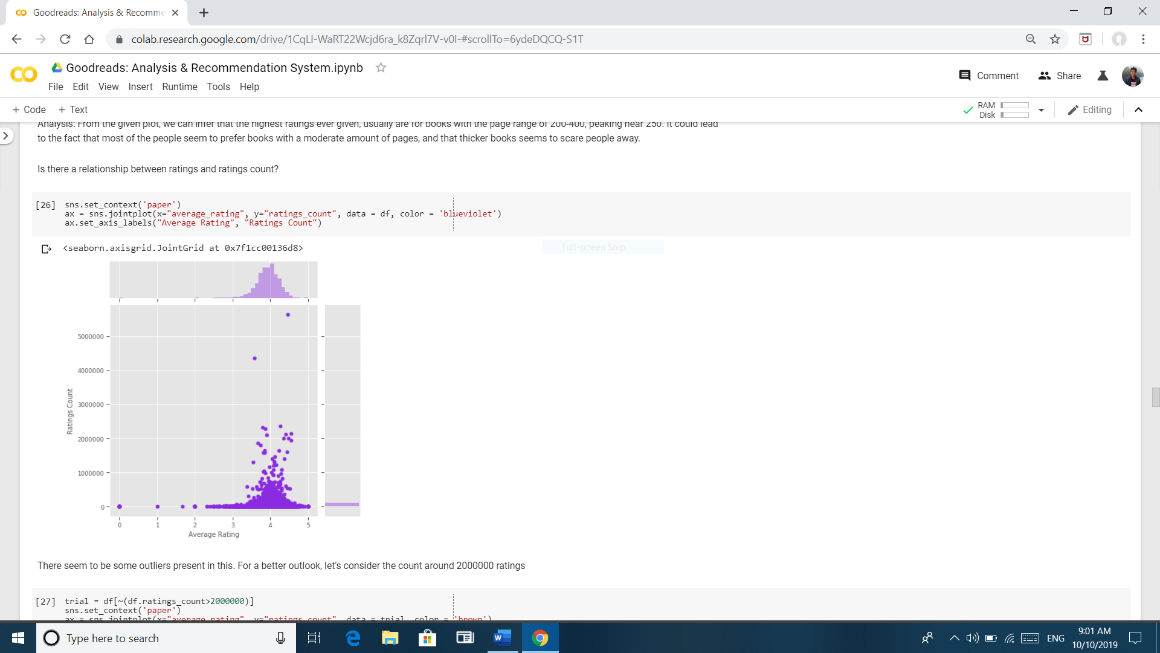
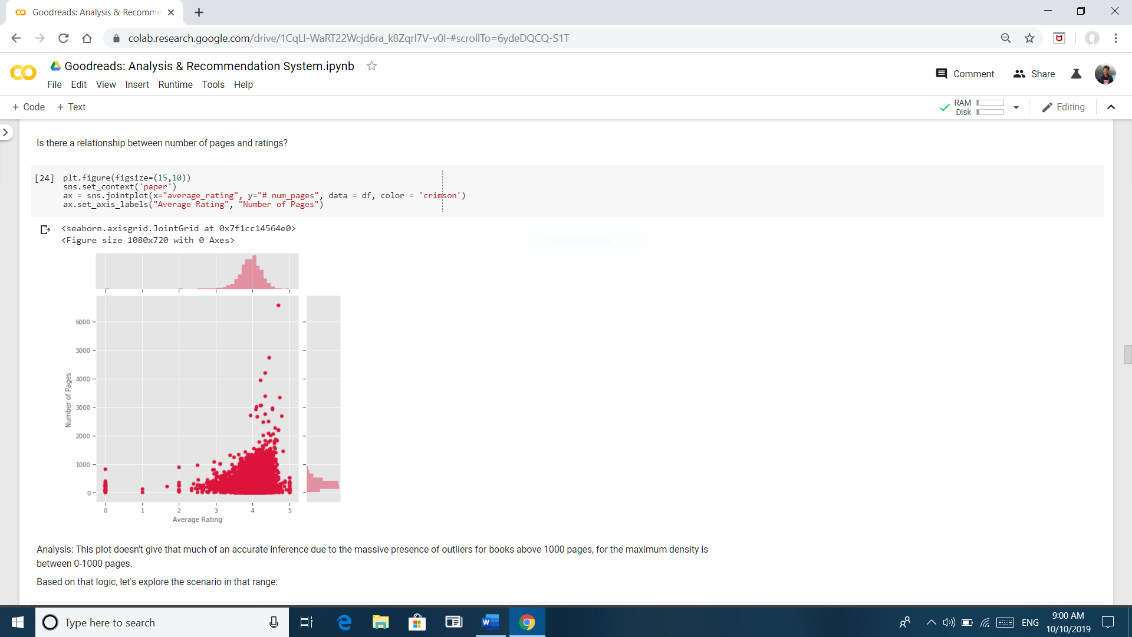
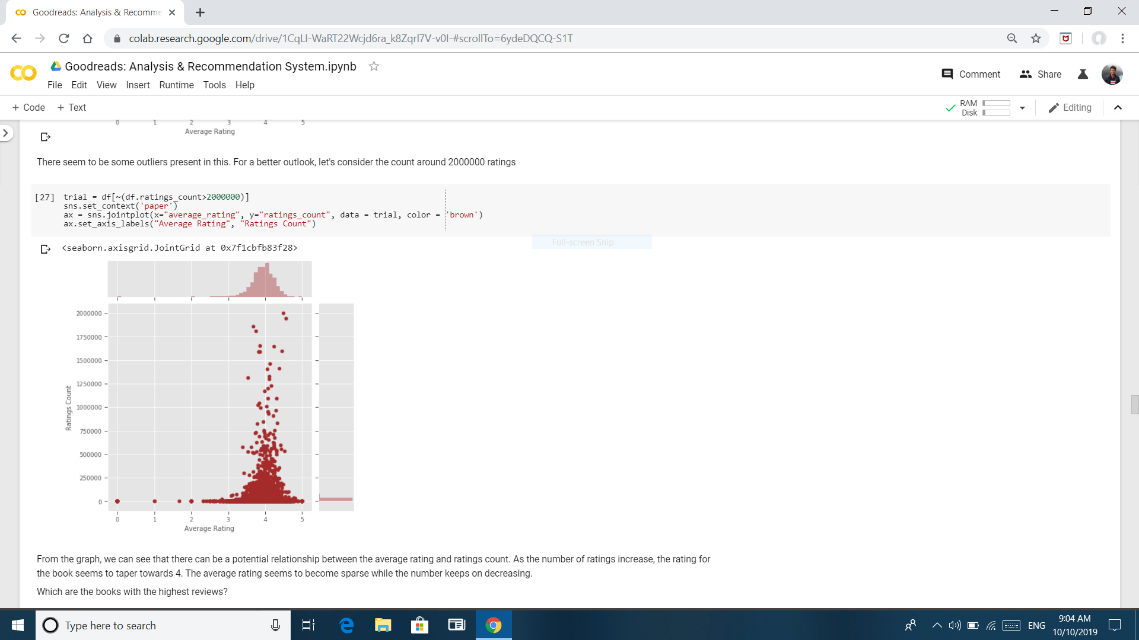
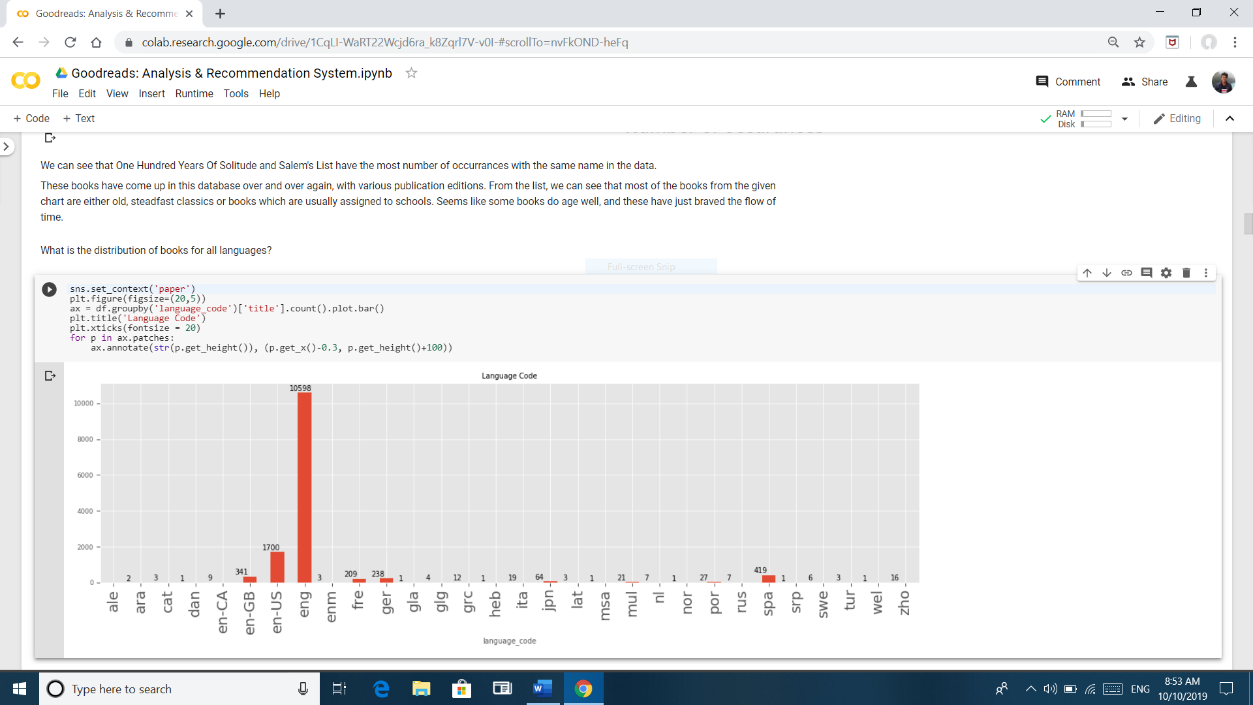
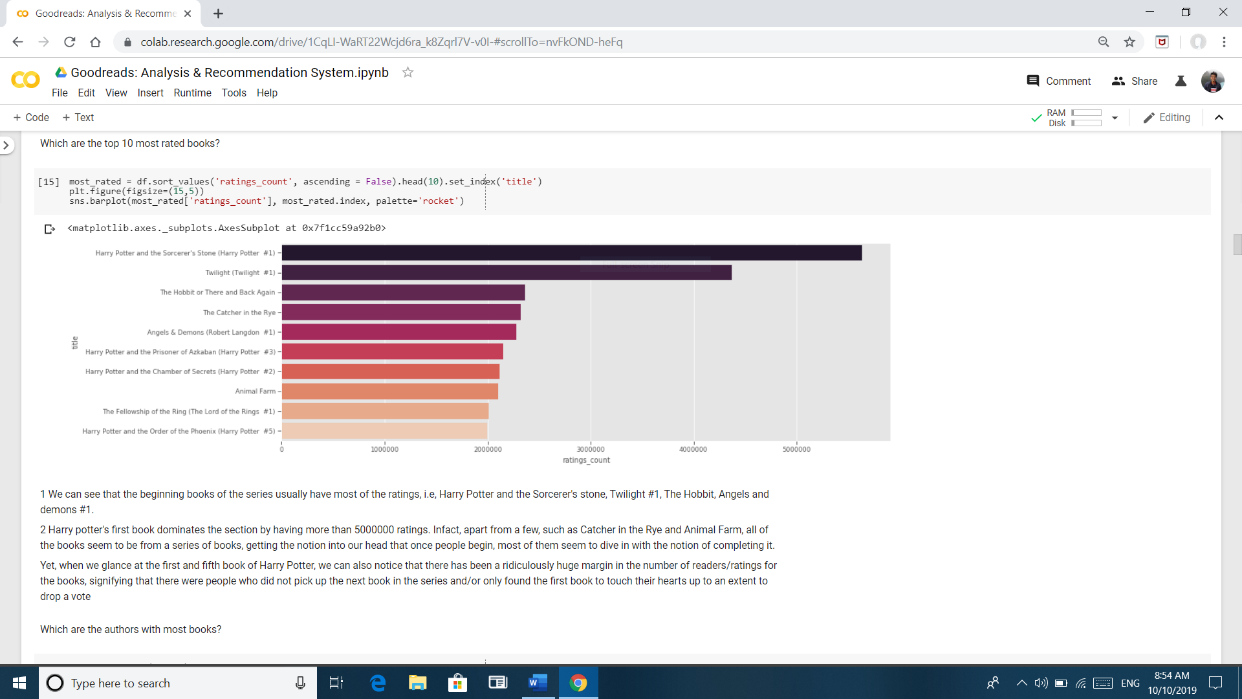
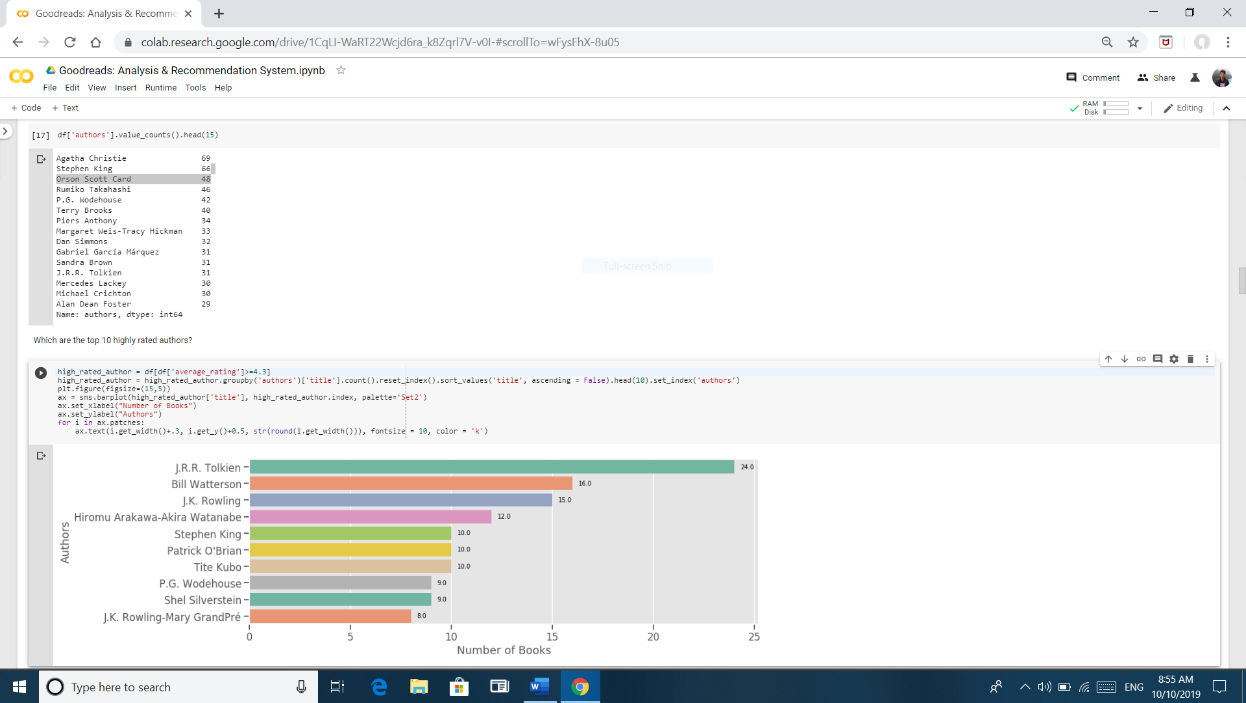
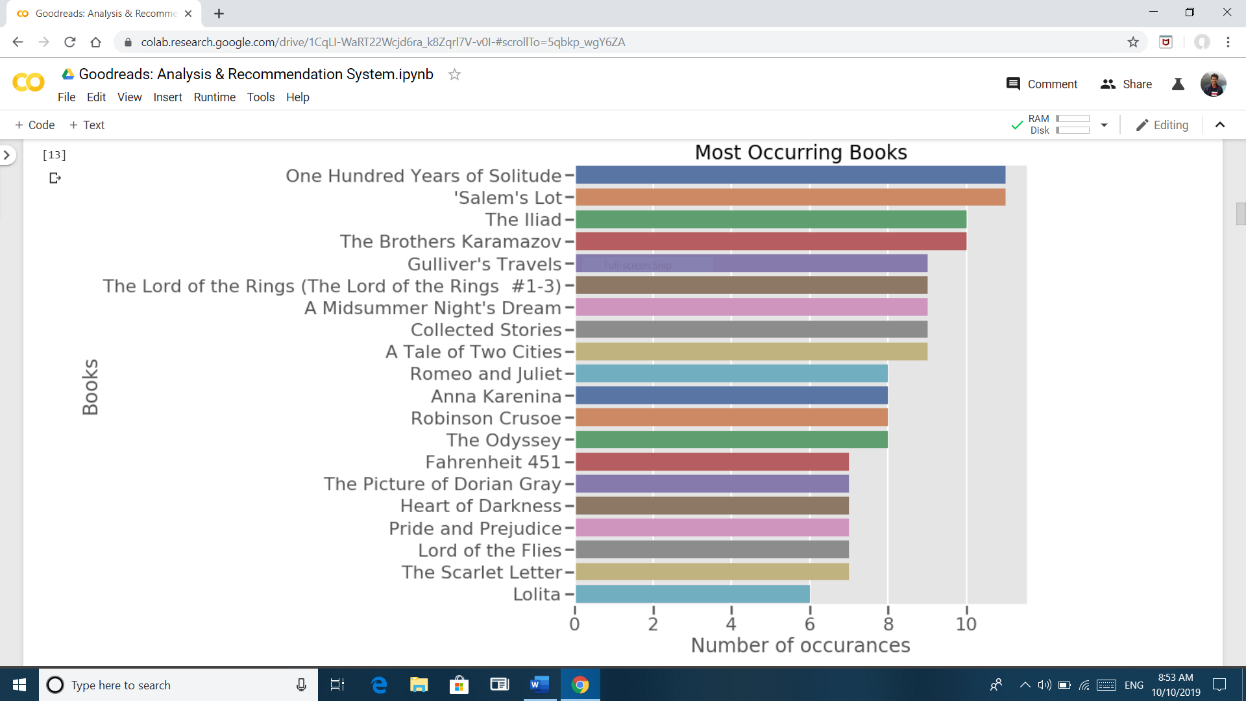
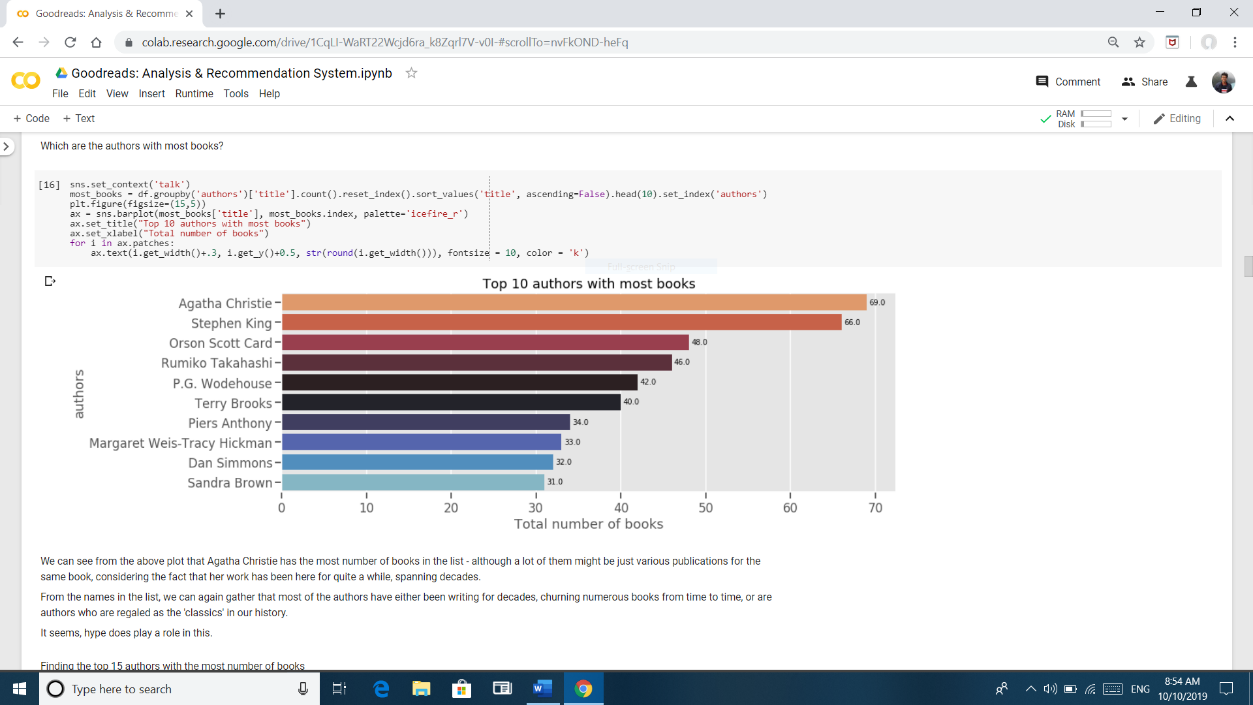
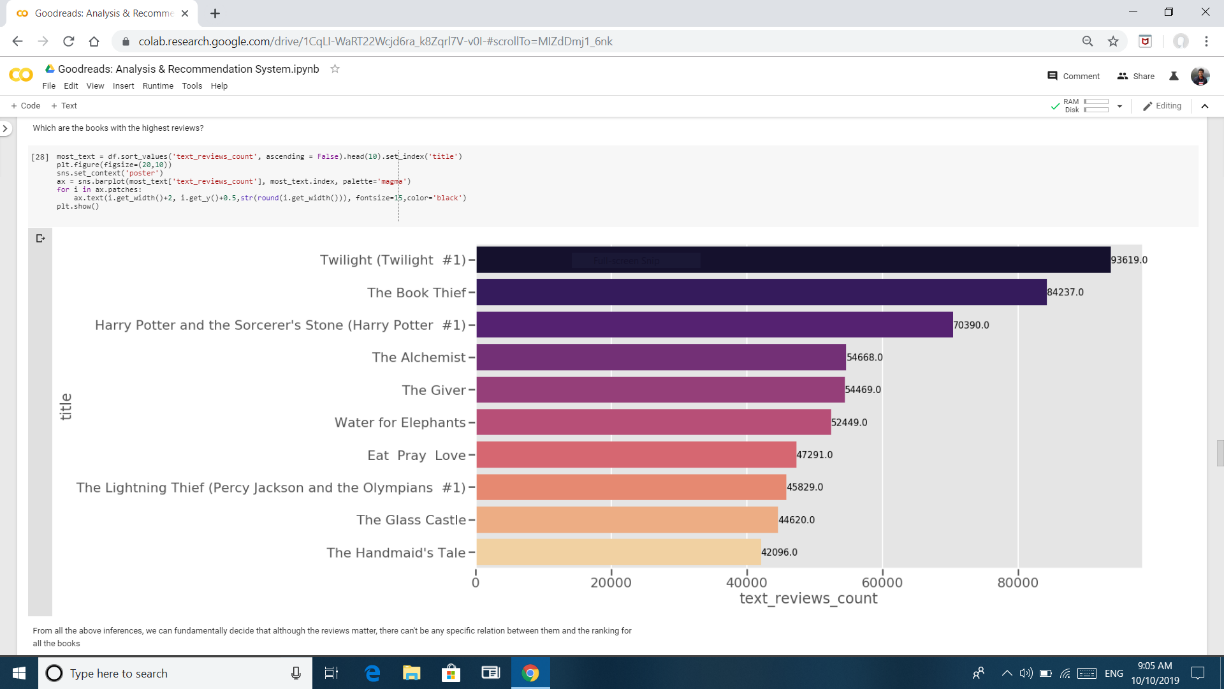
Demo Results

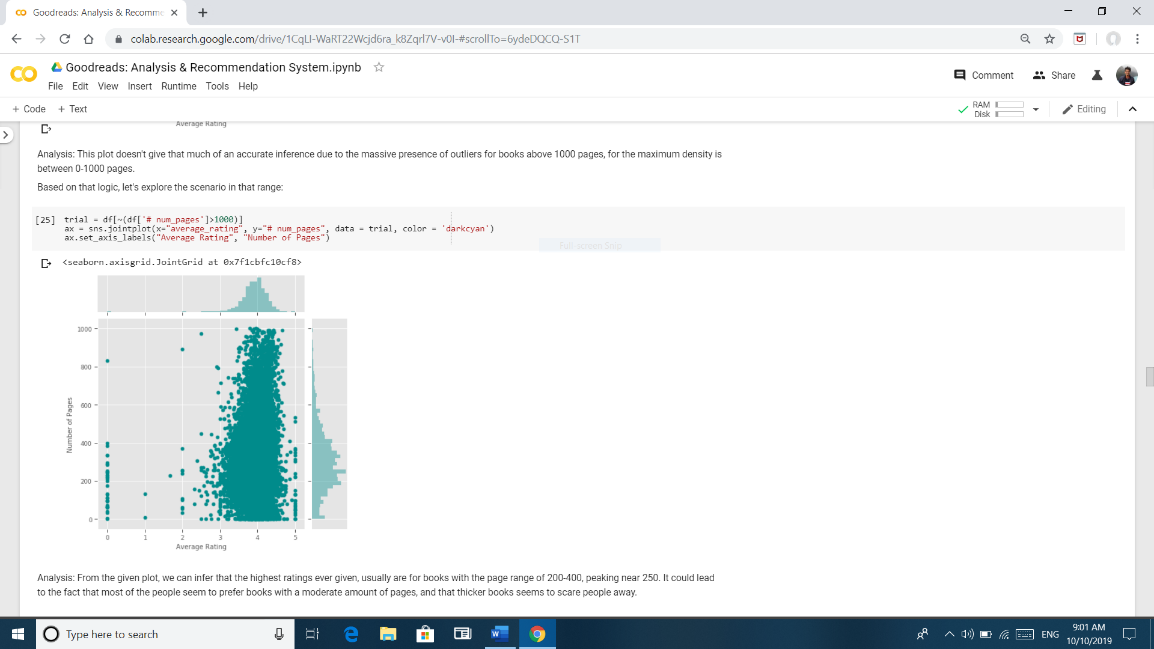
**LIST OF ANNEXURE**

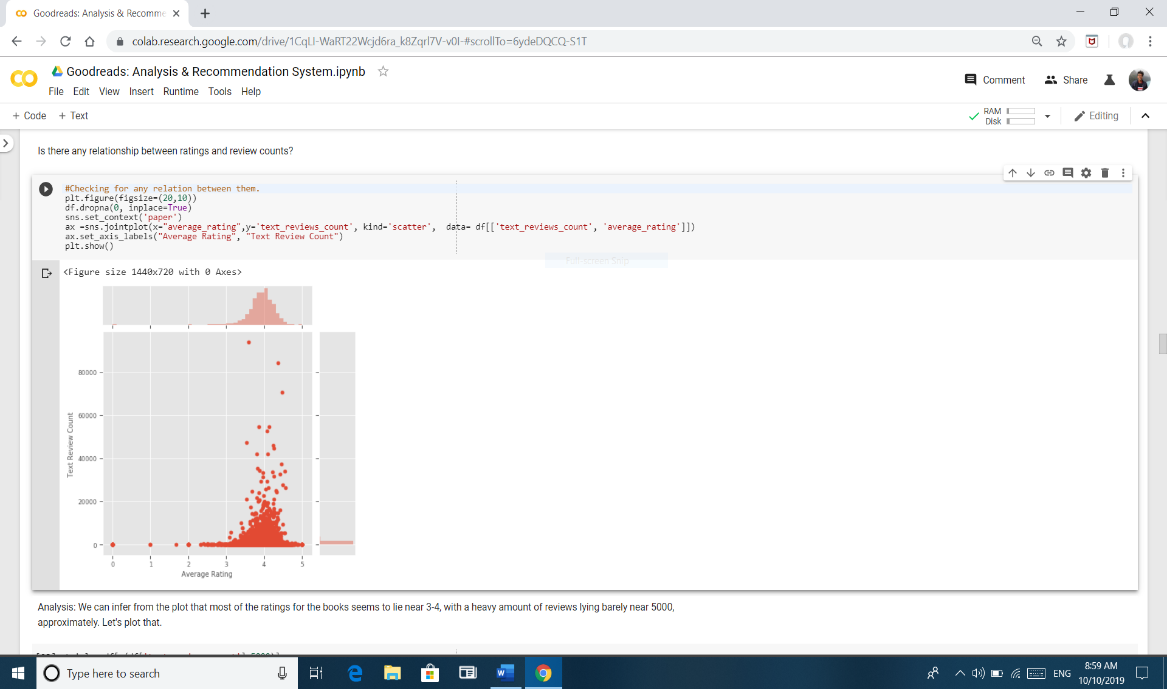
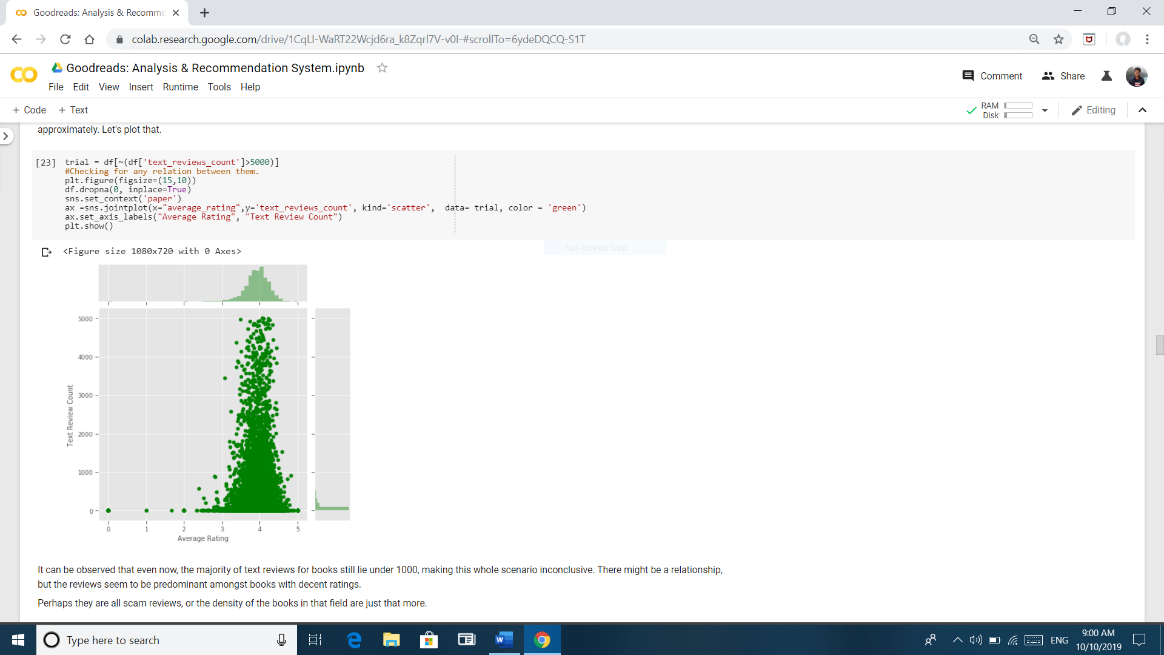


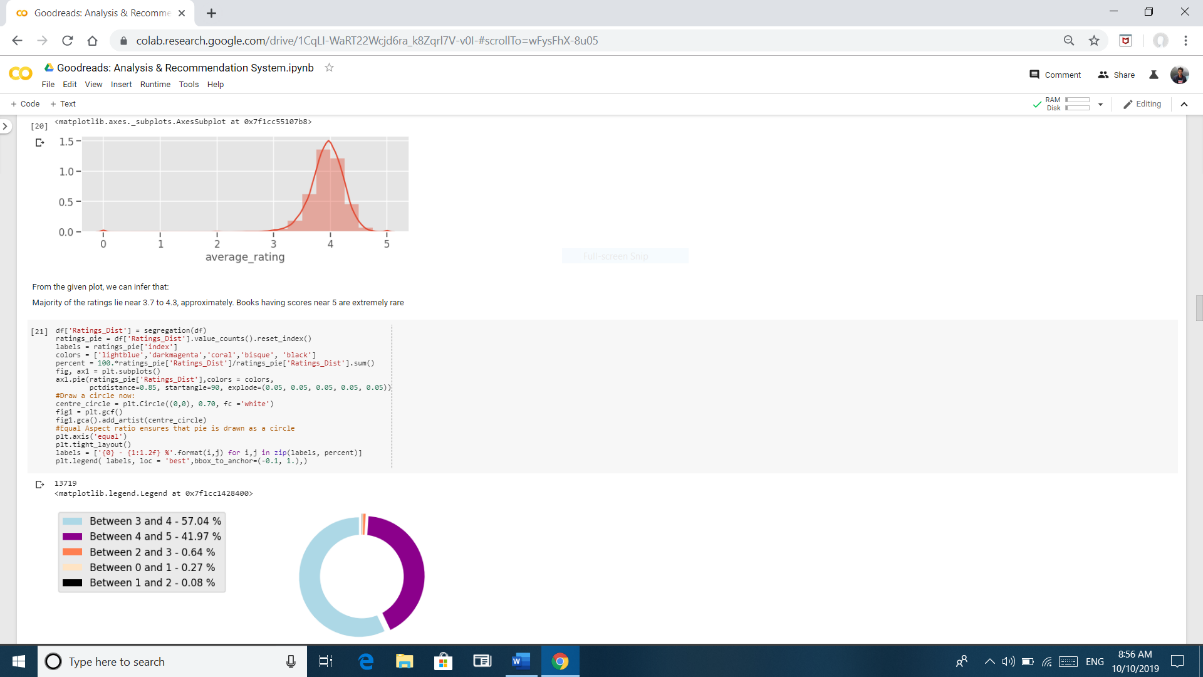


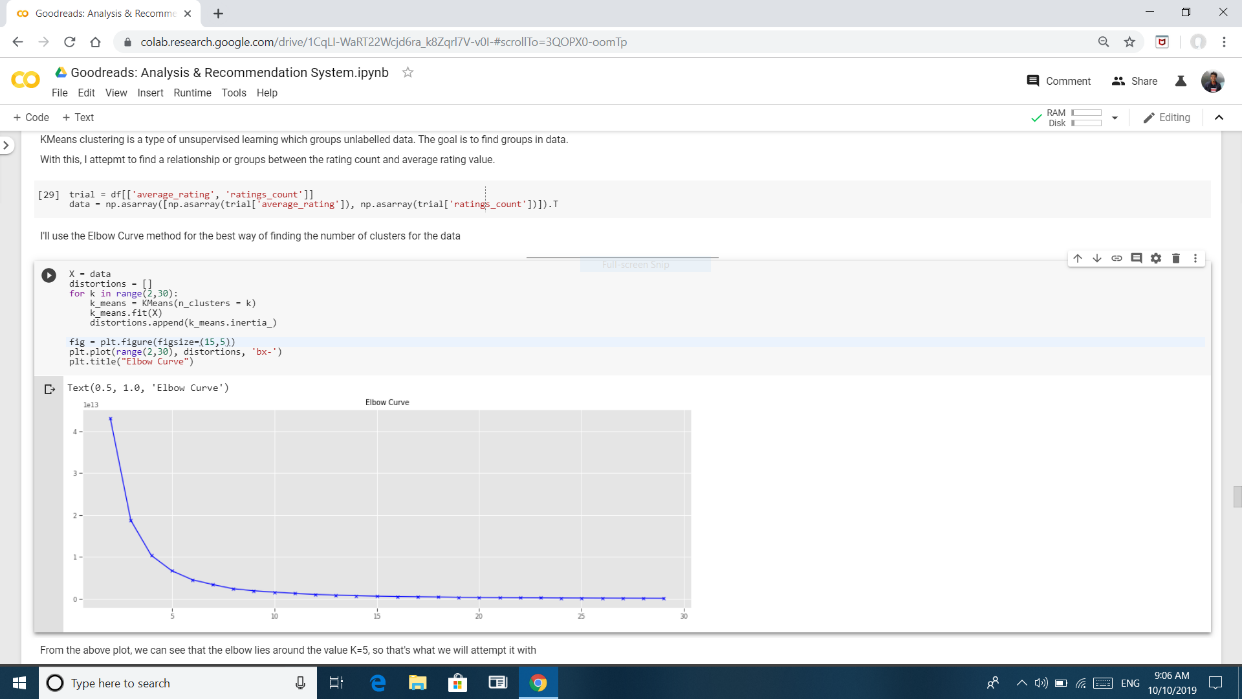


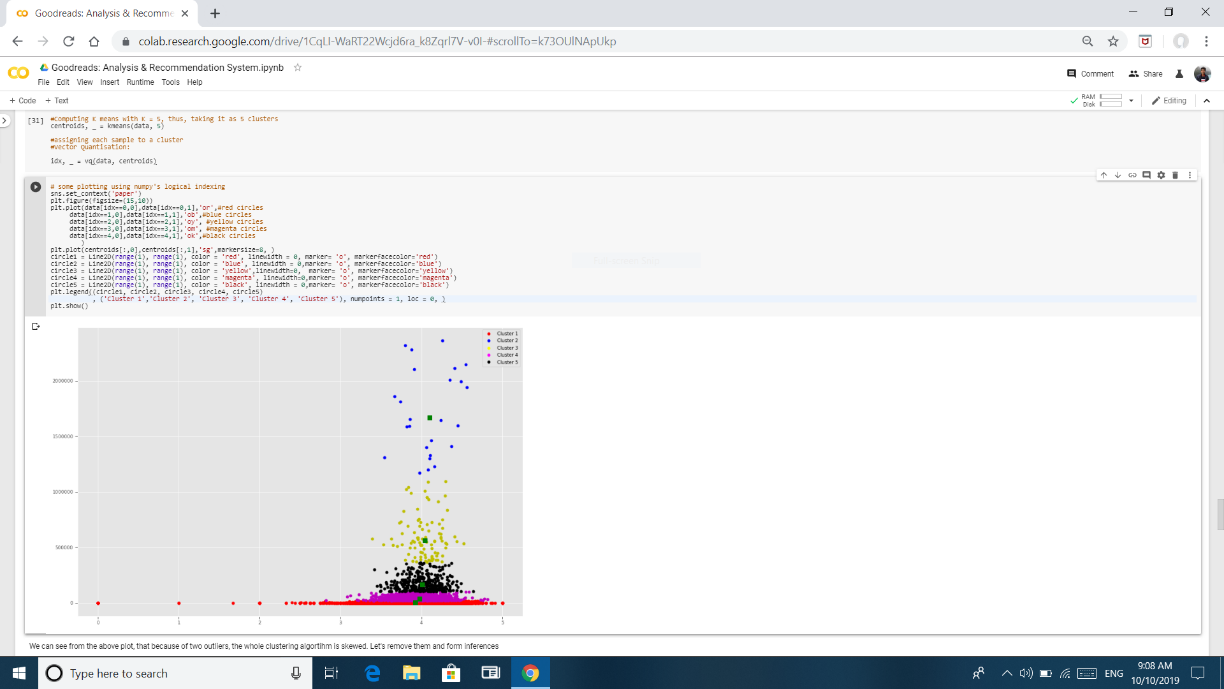


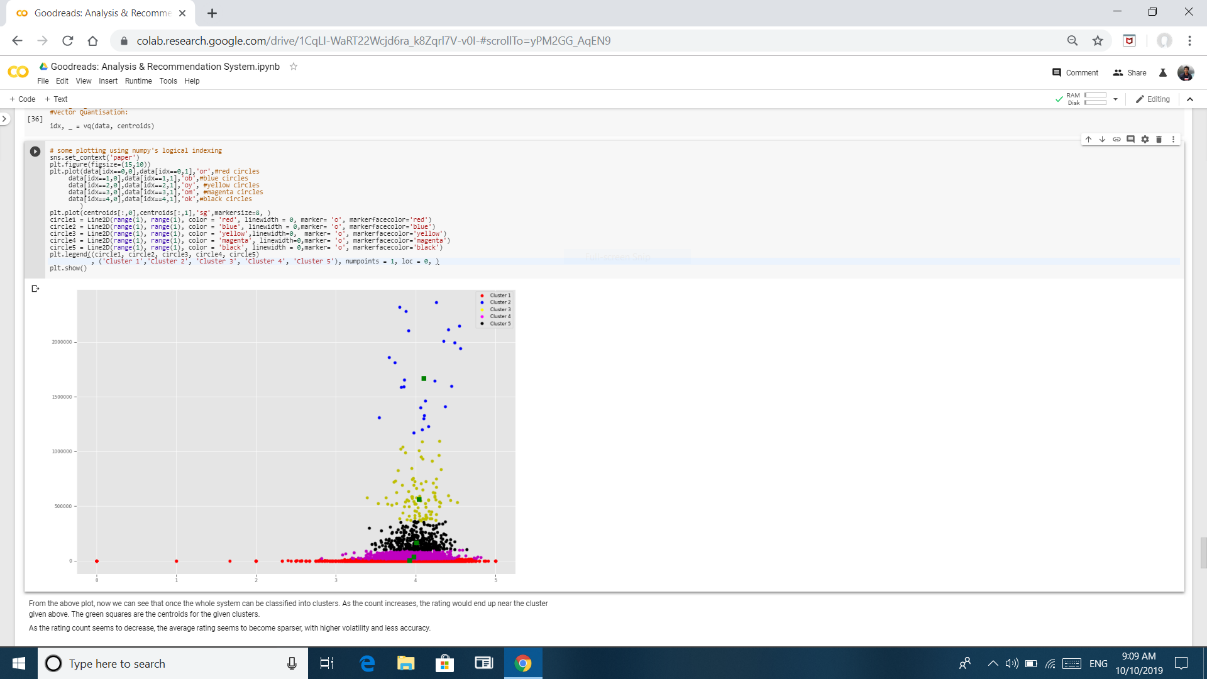


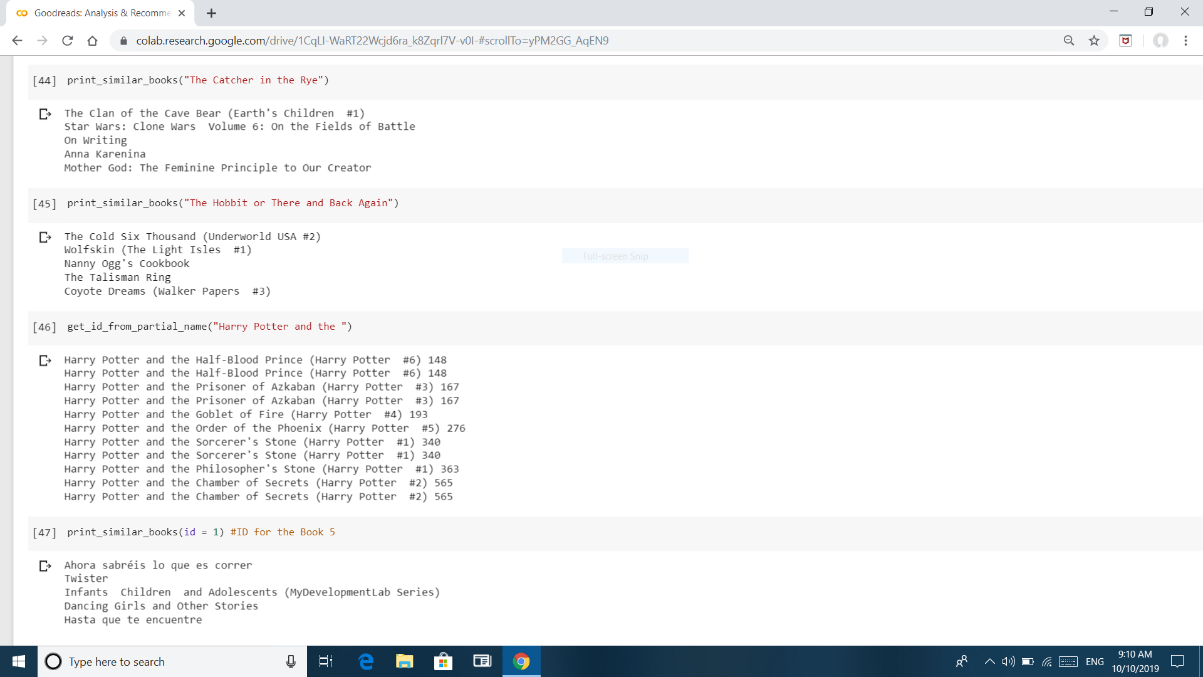












**BIBLIOGRAPHY**

**1.** <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>

**2.** <https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a>

**3.** <https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/>

**4.** <https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed>

**5.** <http://cseweb.ucsd.edu/~jmcauley/datasets.html>

**6.** <https://towardsdatascience.com/beginners-guide-to-machine-learning-with-python-b9ff35bc9c51>