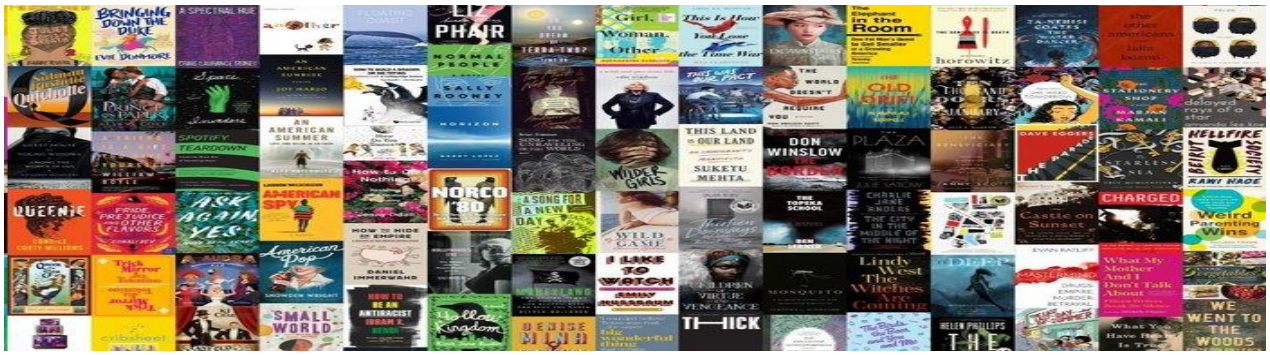
Book Recommendation System



Contents:

1 Introduction

We have gone through a great journey throughout this course where we got the opportunity to study the data lifecycle in depth with hands-on its different subject areas. Through the medium of this project, I aim to use "Book-Crossing Dataset" and build a recommendation system through distributed computation by using Apache Spark and Hadoop. To build this system we will be using various packages provided by ML Lib Apache Spark for applying collaborative filtering technique to get recommendations for our users .We will be going through every step of data pipeline to get a good look as every phase in depth and deal with the possible challenges.

2 Background

Nowadays, recommendation systems have become an integral part of a huge user base. There are a lot of companies like Netflix, Amazon, Facebook, LinkedIn which use sophisticated technology to build a strong recommendation system to increase its customer base. There are three major techniques which are frequently used to build a recommendation system Collaborative filtering, Content- based filtering and Hybrid technique. In this project, we have employed "Alternating least square matrix factorization" method, which is an optimization technique for collaborative filtering algorithm.

Collaborative Filtering: The basic idea behind collaborative filtering is that you have a database full of user preferences and the algorithm predicts the missing user-item interactions according to user's past preferences which he has not used yet. The two types of dataset which can be used for such a setup are:

Explicit: Dataset having rating according to user.

Implicit: Dataset with information like click, view purchases.

I have used explicit data for my implementation, as we can easily see whether a user enjoyed a book based on the rating provided.

Dimensionality Reduction: As we aspire to build a recommendation system, we must work on a huge data repository to get as accurate prediction as possible. In case the of the user-item matrix where all users are cross joined with all the items, if the matrix is mostly empty then reducing its dimensionality can considerably increase performance in terms of both time and space. Thus, matrix factorization comes handy in this situation. Hence, we have used alternating least square matrix factorization for implementing collaborative filtering to get recommendations.

Alternating least square: Uses matrix factorization algorithm where you decompose your large user-item matrix into lower dimensional user factors and item factors and then estimate the user rating by mathematical computation and optimizations.

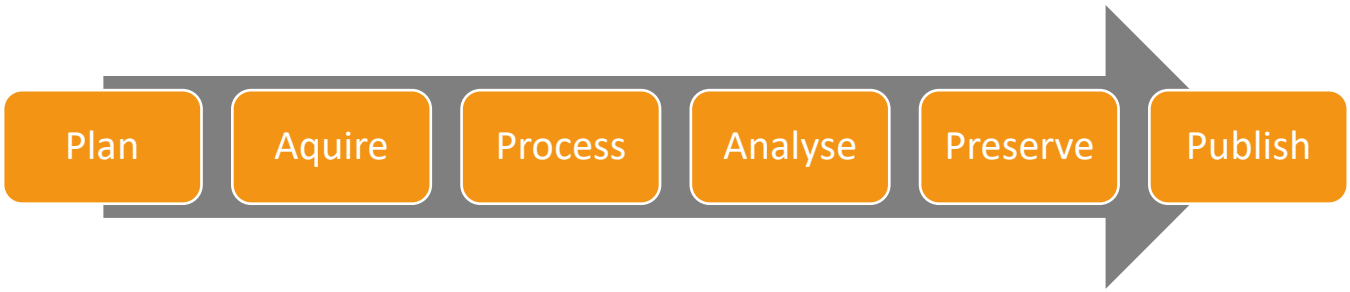
Let's assume we have an original ratings matrix R of size MxN, where M is the number of users and N is the number of items. This matrix is quite sparse since most users only interact with a few items each. We can factorize this matrix into two separate smaller matrices: one with

dimensions MxK which will be our latent user feature vectors for each user (U) and a second with dimensions KxN, which will have our latent item feature vectors for each item (V). Multiplying these two feature matrices together approximates the original matrix, but now we have two matrices that are dense including a number of latent features K for each of our items and users. We can use SVD (Singular vector decomposition) to carry out the factorization but it will involve inverting large matrix and thus will be computationally expensive. With ALS, we can just solve one feature vector at a time, thus it can run in parallel making it an ideal choice to be implemented on Spark.[7]

3 Methodology

The project has been designed while keeping in mind the different phases encountered in a data lifecycle so that we can consider all the resources which we will be needing at all stages. I have followed the USGS (US geological survey) Data Life Cycle model which aims to develop the best data management strategy. The model consists of six components and three cross cutting activities.

Data Lifecyle



The cross-cutting activities involve:

- ***Describe:*** We aim to describe the metadata and documentation.
- ***Manage Quality***
- ***Manage Backup & Rest of Data Corruptions***

3.1 Plan

At this stage, we create a plan of the data activity and decide the resources which we are going to use during this project. Since we aim to create a novel recommendation system, first we need to have access of the catalogue of books along with its purchase and feedback history. I have used the "Book-Crossing Dataset" which contains 278,858 users and 271,379 books information along with user ratings.

Dataset: The books-crossing dataset consists of three tables:

- **BX-Users:** It contains user information with fields like "Location" and" Age" along with User-ID.
- **BX-Books:** Books are identified by ISBN numbers which are the primary key of this table. The other attributes of books involve fields like (`Book-Title`, `Book-Author`, `Year-Of-

Publication`, `Publisher`). The table also contains the book's cover page image url in three different sizes large, medium and small. All this information has been collected from Amazon web services.

- **BX-Book-Ratings:** Contains the book rating given by different users from scale 1-10 where higher value depict higher appreciation where 0 represents implicit rating.[12]

Tools Used:

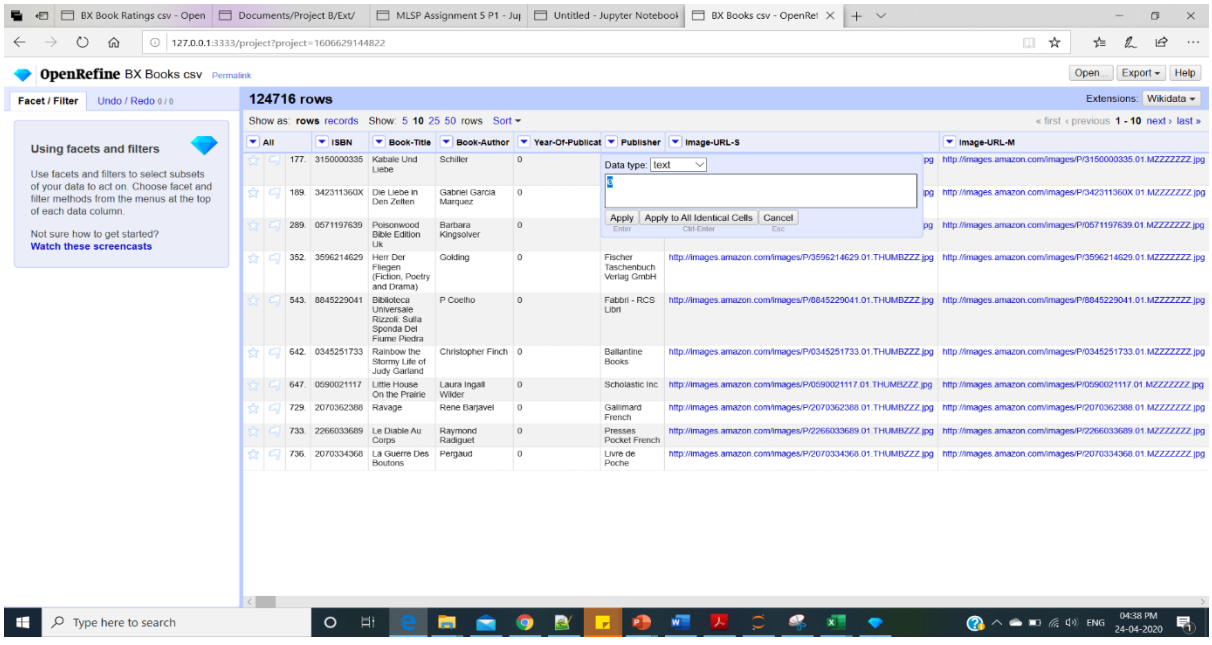
- **Open Refine:** standalone open source desktop application for data clean-up and transformation to other formats, the activity known as data wrangling. [Wikipedia]
- **Apache Spark:** Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. [Wikipedia]
- **Hadoop:** It provides a software framework for distributed storage and processing of big data.
- **PySpark:** PySpark is the Python API for Spark
- **Jetstream VM:** Virtual Machine environment developed by Indiana University Bloomington.
- **Jupyter Notebook:** In this project used for carrying out visualization tasks.

3.2 Acquire

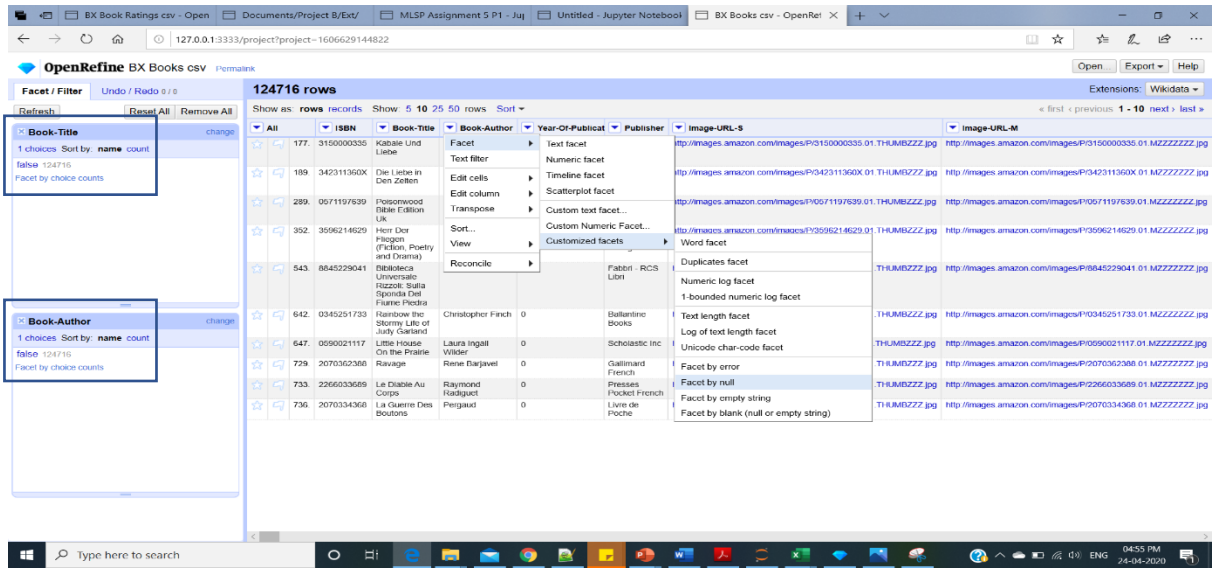
In this phase, we focus on retrieving/ generating relevant data. This has to be done while maintaining the best practice to ensure data integrity. In order to acquire the data, I retrieved the CSV dump from the official site of this dataset http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

To ensure the data quality I used the tool Open refine to deal with the nuances of the data. The following characteristics of data were kept in mind while performing the cleaning.

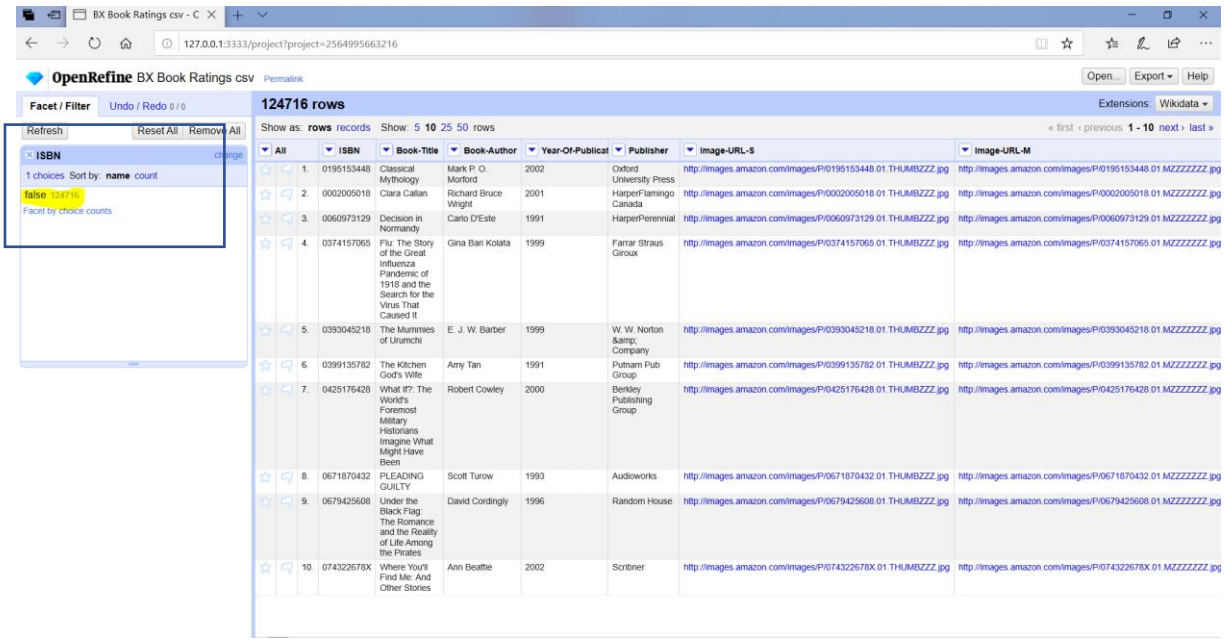
- **Accuracy:** Verified the correctness of the data by checking the grains of the data. Found Year='0' which is an invalid number for Year, so replaced it with NULL by using Open Refine filter.



- **Completeness:** Checked if all the ID's had corresponding attributes and there were no NULL values by using null facet functionality of Open Refine.



- **Uniqueness:** Checked if there were duplicate primary keys in the table by using Duplicate Facet which gives the total number of records which are duplicate against TRUE and unique records against FALSE.



- **Uniformity:** Checked if similar information was presented in a similar way across the system. This was done by eyeballing the data and checking errors during data ingestion.

Future Data Cleaning Proposal:

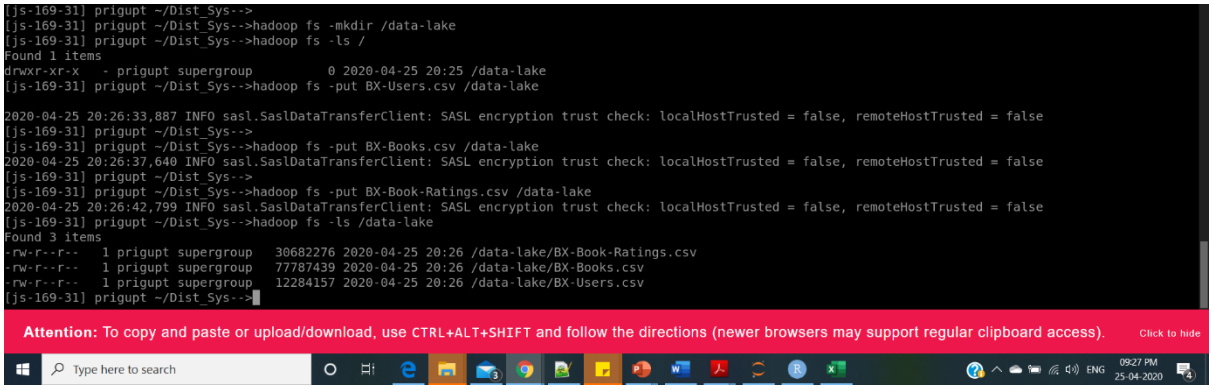
- Removal of all alphanumeric symbols like *&%$# from the data. Although I have cleaned these characters at my level using a filter a finer look at the data is necessary.

- The format of location can further be bifurcated into City, Area and Country for better readability.

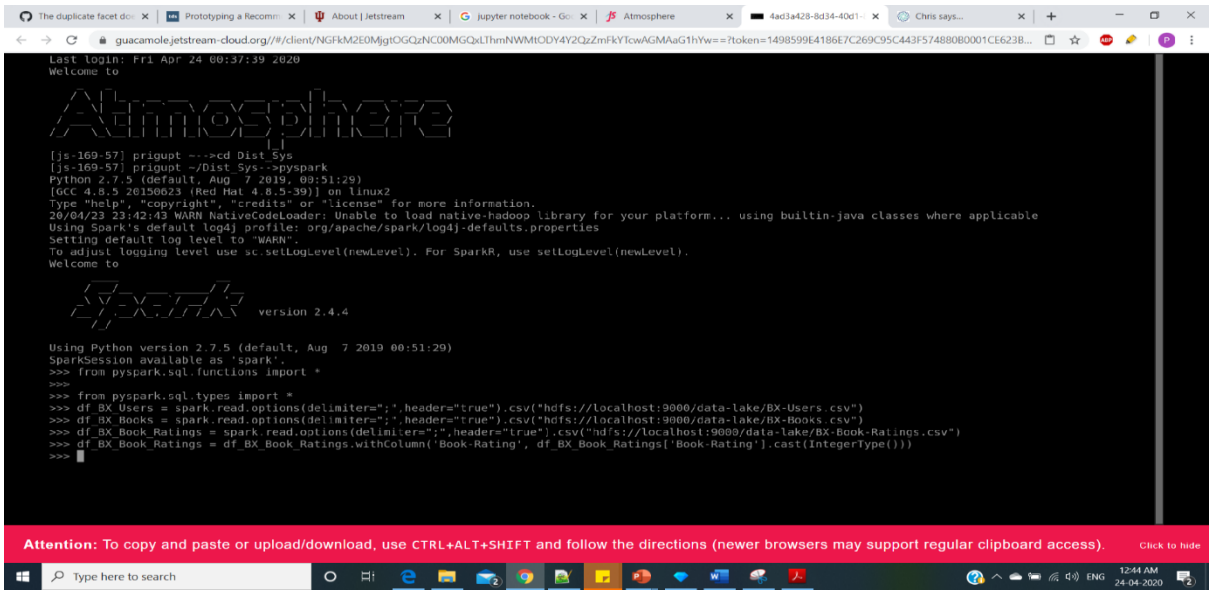
3.3 Process

This phase accounts for the preparation of data for analysis. Extract, Transform and Load operations are the integral part of this stage and the expected outcome is the readiness of data for further analysis.

Data Storage: Since the dataset is comparable to big data, I chose to use **Hadoop Distributed File System (HDFS)** as a primary data storage system. It implements distributed file system that provides high-performance access to data across highly scalable Hadoop clusters. This system provides a reliable means for managing big data pools and also supports analytics. In order to ensure long term preservation of data, I created a data lake on HDFS and placed all the needed dataset on it.



Extract: I installed Hadoop and Spark on Jetstream virtual machine which was the platform used to perform ETL operations on the data. The comma separated file format was put on HDFS which was further accessed by Apache Spark by storing the data in a PySpark dataframe to perform further operations on it.



Transform: Spark gives us objects that represent a dataset and you can call methods on that it to perform operations on it. With Spark, we can work directly with HDFS data and operate on it. Certain transformations were required to be performed on data for implementation of the recommendation system were:

- Transforming Rating from text to integer for doing analysis like average rating.
- Joining two data frames to get book ratings along with user and book information to get rating of a book corresponding to a user.
- Removing the Nulls produced after the join as it gave exception while training the model.
- Since the ALS algorithm in ML Lib only accepts user-id and book-id in integer format, the needed transformation had to be made on PySpark.
-

```
>>
>> df_join = df_BX_Books.join(df_BX_Book_Ratings, on=['ISBN'], how='inner')
>> ratings=df_join.select('ISBN','User-ID','Book-Rating')
>>
>> ratings = ratings.withColumn('ISBN', ratings['ISBN'].cast(IntegerType()))
>>
>> ratings = ratings.withColumn('User-ID', ratings['User-ID'].cast(IntegerType()))
>>
>> ratings=ratings.filter(ratings["User-ID"].isNotNull())
>>
>> ratings=ratings.filter(ratings["ISBN"].isNotNull())
>>
>>
```

Load: To create visualization graphs, I extracted the CSV from these joined dataframe with appropriate filter using Spark API to perform exploratory data analysis on Jupyter Notebook using python. A few extract loading procedures are shown below:

```
KeyboardInterrupt
>>> df_BX_Books_filtered=df_BX_Books.filter(df_BX_Books["Publisher"].contains('Zebra Books'))
>>> df_BX_Books_filtered.write.format('com.databricks.spark.csv').save('prigupt/book_join_pub.csv',header = 'true')
```

```
>>> df_BX_Books_filtered=df_BX_Books.filter(df_BX_Books["Book-Author"].contains('Agatha Christie'))
>>>
>>> df_BX_Books_filtered.write.format('com.databricks.spark.csv').save('prigupt/book_join_auth.csv',header = 'true')
>>>
```

3.4 Analysis

For this phase I proceeded with conducting two types of analytics to explore the data I am working with:

1. **Descriptive Analytics:**

With these analytics, I intended to focus on basic understanding of what is happening with my data. I conducted the analytics using Seaborn and Pandas packages in Python to summarize the data and analyse its distribution by plotting the data.

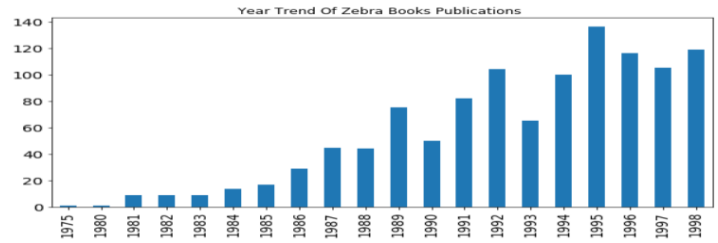
➢ Analysing publishing trend of the publisher "Zebra Books" over the years.

Code: df_1=pd.read_csv("Ext/Book_join_pub.csv")

df_1["Year-Of-Publication"].value_counts().sort_index().head(20).plot.bar(figsize=(10,5),fontsize=14, x='Year',y="Frequency",title="Year Trend Of Zebra Books Publications ")

```
# Books published by publisher in years
df_1=pd.read_csv("Ext/Book_join_pub.csv")
df_1.head()
df_1["Year-Of-Publication"].value_counts().sort_index().head(20).plot.bar(figsize=(10,5),fontsize=14, x='Year',y="Frequency",tit
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x27c4d9de860>
```



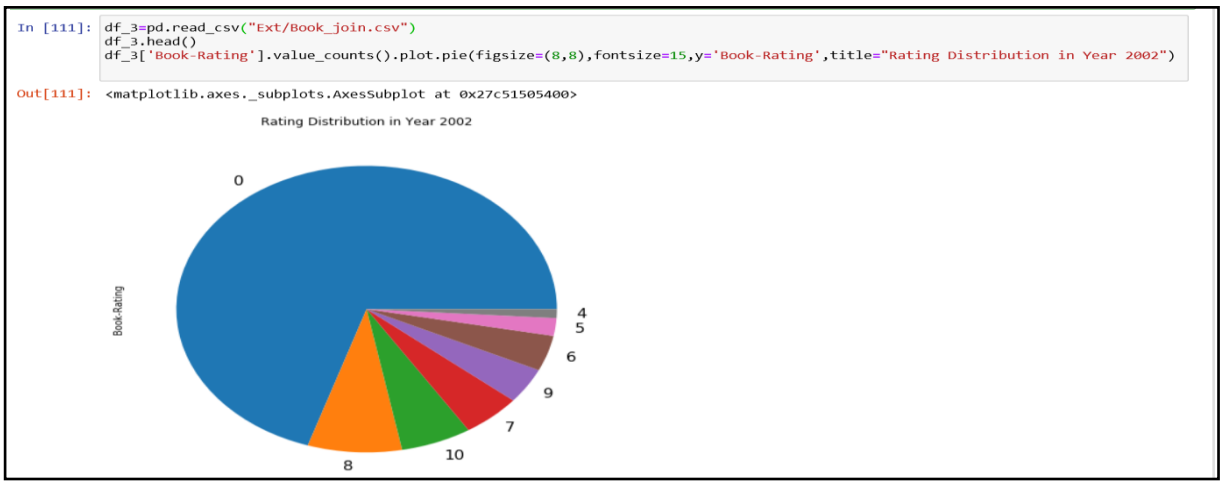
Interpretation: We can see here that zebra books publication has published more books from 1995 to 1998.

- Book rating distribution over the year 2002

Code: df_3=pd.read_csv("Ext/Book_join.csv")
df_3['Book-Rating'].value_counts().plot.pie(figsize=(8,8),fontsize=15,y='Book-Rating',title="Rating Distribution in Year 2002")

```
In [111]: df_3=pd.read_csv("Ext/Book_join.csv")
          df_3.head()
          df_3['Book-Rating'].value_counts().plot.pie(figsize=(8,8),fontsize=15,y='Book-Rating',title="Rating Distribution in Year 2002")
```

```
Out[111]: <matplotlib.axes._subplots.AxesSubplot at 0x27c51505400>
```



Interpretation: This shows that most of the books in 2002 have been rated 0 after that we a lot of books rated 8 and 10.

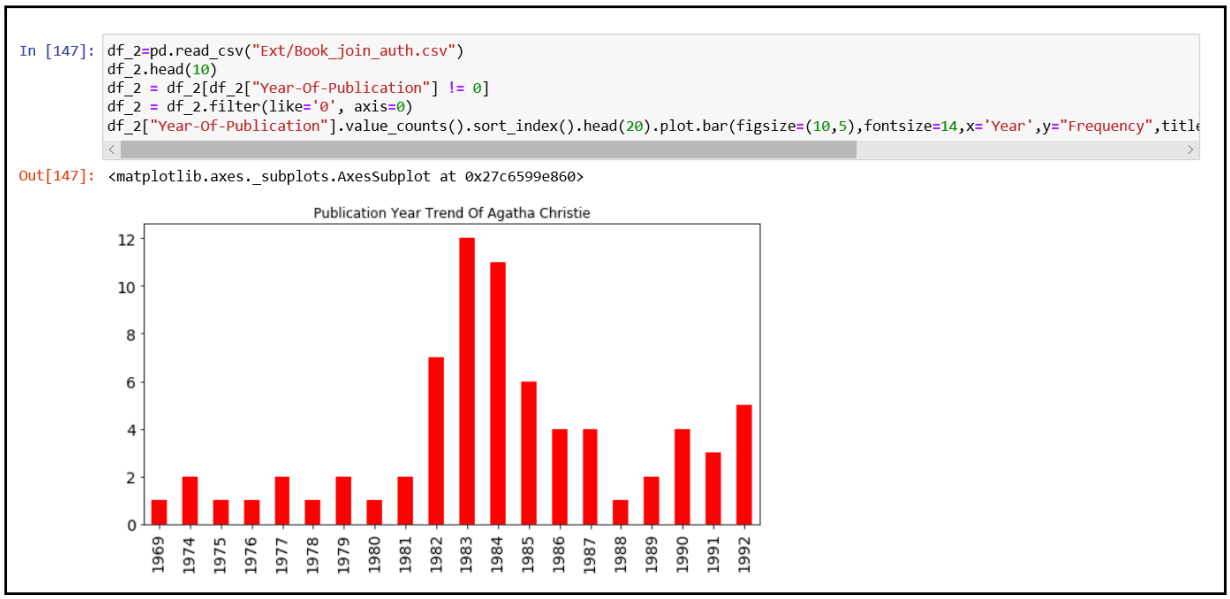
- Book publishing trend of the writer "Agatha Christie"

Code: df_2=pd.read_csv("Ext/Book_join_auth.csv")

df_2 = df_2[df_2["Year-Of-Publication"] != 0]

df_2 = df_2.filter(like='0', axis=0)

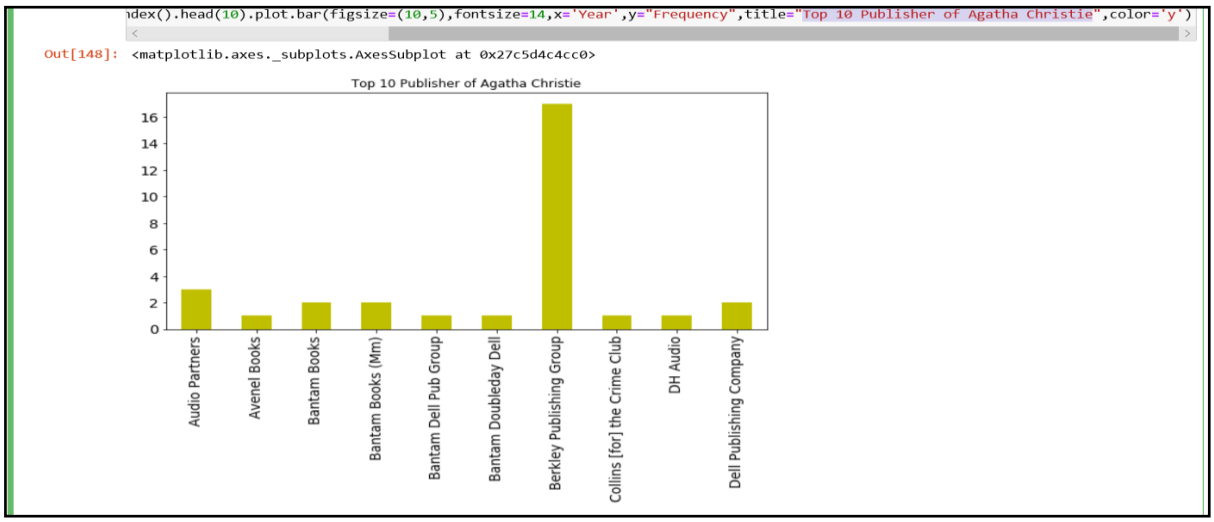
df_2["Year-Of-Publication"].value_counts().sort_index().head(20).plot.bar(figsize=(10,5),fontsize=14,x='Year',y="Frequency",title="Publication Year Trend Of Agatha Christie",color='r')



Interpretation: This shows that most of the books by Agatha Christie have been published in the year 1983-1984.

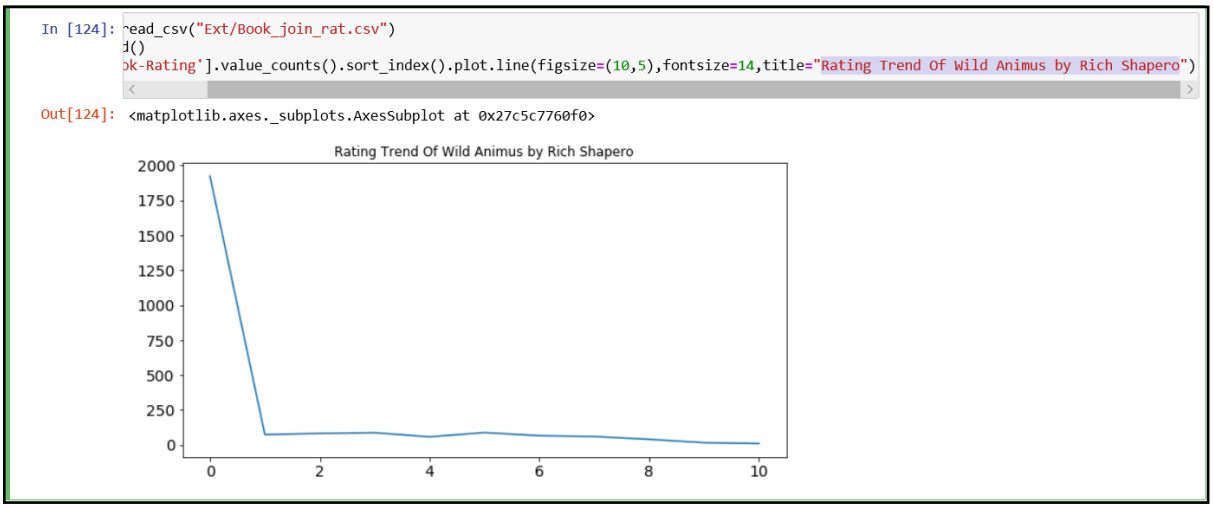
- Top 10 Publisher of Agatha Christie.

Code:df_2["Publisher"].value_counts().sort_index().head(10).plot.bar(figsize=(10,5),fontsize=14,x='Year',y="Frequency",title="Top 10 Publisher of Agatha Christie",color='y')



Interpretation: Most of the books by Agatha Christie have been published by Berkley Publishing Group.

- Rating count Of Wild Animus by Rich Shapero.

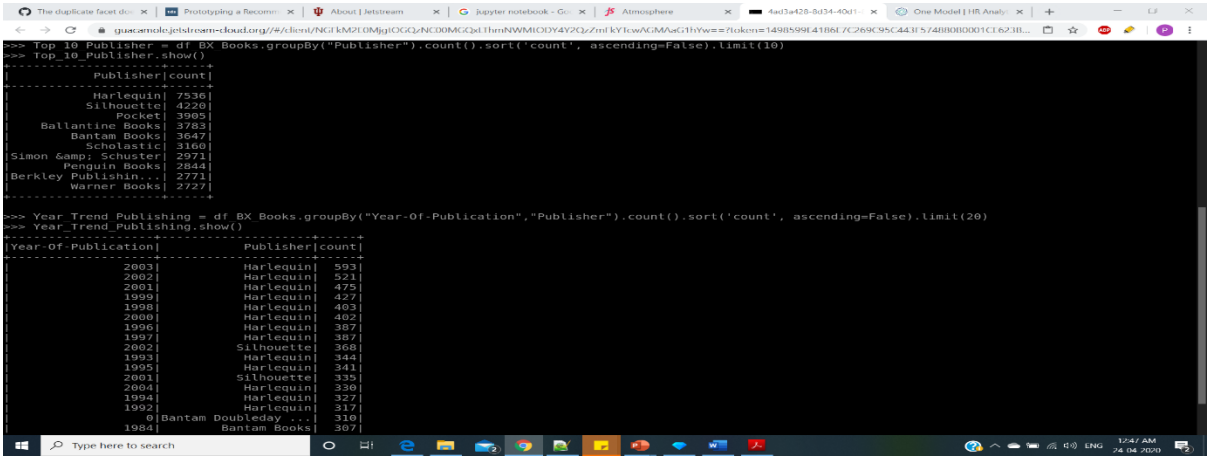


Interpretation: Most of the ratings of the book Wild Animus by Rich Shapero is ranged between 0 and 4.

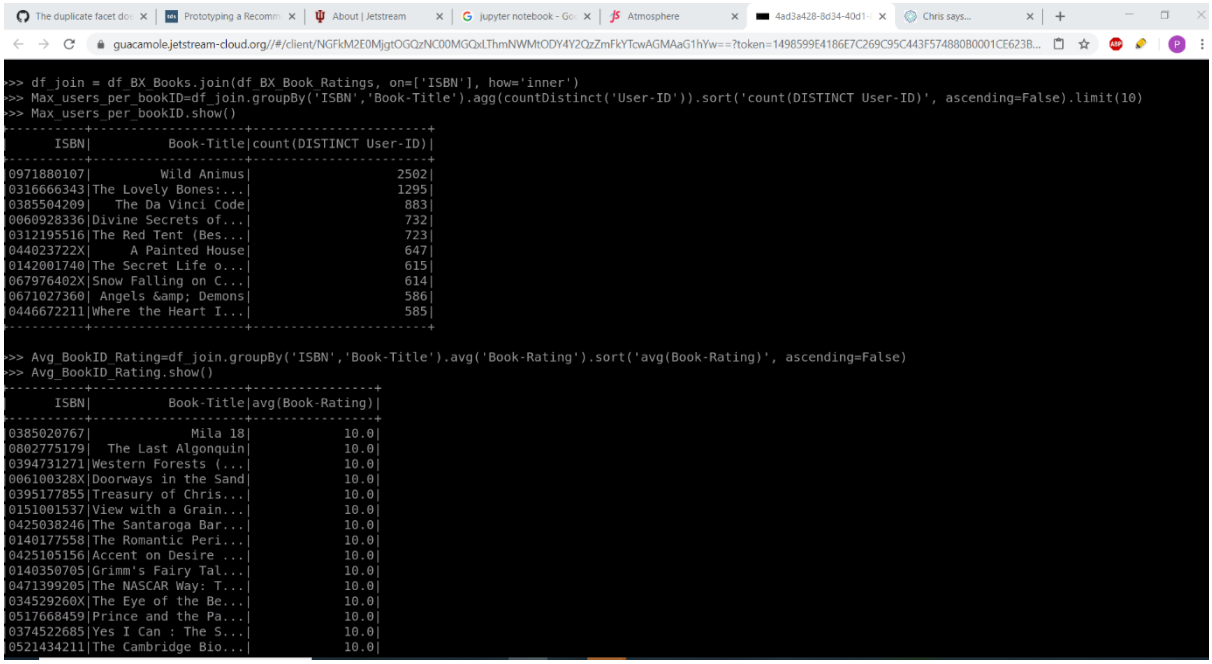
2. Diagnostic Analytics:

This analysis allowed me to examine data by drilling down into groups and getting the counts and averages for them. I performed this type of analysis by using the various dataframe operations offered by PySpark.

- Following screen shots gives the name of the Top 10 Publisher who have published maximum books according to the dataset. It also gives the year wise trends of publishers with maximum books published.



- Following screen shots gives the top 10 books with maximum users. It also gives the average rating of the books in sorted order.



```
>>> df_join = df_BX_Books.join(df_BX_Book_Ratings, on=['ISBN'], how='inner')
>>> Max_users_per_bookID=df_join.groupBy('ISBN','Book-Title').agg(countDistinct('User-ID')).sort('count(DISTINCT User-ID)', ascending=False).limit(10)
>>> Max_users_per_bookID.show()
+----------+--------------------+----------------------+
|      ISBN|          Book-Title|count(DISTINCT User-ID)|
+----------+--------------------+----------------------+
|0971880107|         Wild Animus|                  2502|
|0316666343|The Lovely Bones:...|                  1295|
|0385504209|   The Da Vinci Code|                   883|
|0060928336|Divine Secrets of...|                   732|
|0312195516|The Red Tent (Bes...|                   723|
|044023722X|     A Painted House|                   647|
|0142001740|The Secret Life o...|                   615|
|067976402X|Snow Falling on C...|                   614|
|0671027360| Angels &amp;amp; Demons|                   586|
|0446672211|Where the Heart I...|                   585|
+----------+--------------------+----------------------+

>>> Avg_BookID_Rating=df_join.groupBy('ISBN','Book-Title').avg('Book-Rating').sort('avg(Book-Rating)', ascending=False)
>>> Avg_BookID_Rating.show()
+----------+--------------------+---------------+
|      ISBN|          Book-Title|avg(Book-Rating)|
+----------+--------------------+---------------+
|0385020767|            Mila 18|           10.0|
|0802775179|  The Last Algonquin|           10.0|
|0394731271|Western Forests (...|           10.0|
|006100328X|Doorways in the Sand|           10.0|
|0395177855|Treasury of Chris...|           10.0|
|0151001537|View with a Grain...|           10.0|
|0425038246|The Santaroga Bar...|           10.0|
|0140177558|The Romantic Peri...|           10.0|
|0425105156|Accent on Desire ...|           10.0|
|0140350705|Grimm's Fairy Tal...|           10.0|
|0471399205|The NASCAR Way: T...|           10.0|
|034529260X|The Eye of the Be...|           10.0|
|0517668459|Prince and the Pa...|           10.0|
|0374522685|Yes I Can : The S...|           10.0|
|0521434211|The Cambridge Bio...|           10.0|
```

- Join the two together (We now have Book Id (ISBN), avg(rating), and count columns)

```
>>> AvgCount = counts.join(Avg_BookID_Rating, 'ISBN').sort('avg(Book-Rating)', ascending=False).limit(10)
>>>
>>> AvgCount.show()
+----------+-----+----------------+
|      ISBN|count|avg(Book-Rating)|
+----------+-----+----------------+
|0140294309|    1|            10.0|
|034529260X|    1|            10.0|
|0140350705|    1|            10.0|
|0060530839|    1|            10.0|
|0140547452|    1|            10.0|
|0140177558|    1|            10.0|
|0142180114|    1|            10.0|
|0155226606|    1|            10.0|
|0194229718|    1|            10.0|
|0094770506|    1|            10.0|
+----------+-----+----------------+
```

- Following are the Top 10 books names.

```
>>> Top_BookNames=AvgCount.join(df_BX_Books, on=['ISBN'], how='inner').limit(10).select("Book-Title").limit(10)
>>> Top_BookNames.show()
+--------------------+
|          Book-Title|
+--------------------+
|Grimm's Fairy Tal...|
|The Eye of the Be...|
|Darwin in 90 Minu...|
|Sharpes Eagle (Sh...|
|Dracula: Level 2:...|
|Deep in the Fores...|
|English Romantic ...|
|The Lion, the Wit...|
|The Black Prince ...|
|The Romantic Peri...|
+--------------------+

>>>
```

3. **Predictive Analytics:** This analytics technique helps to identify what might happen in future. As mentioned earlier, we will be using collaborative filtering method optimized by ALS algorithm. For this procedure first we need to create the training set and the testing set.

- ***Train set:*** Joined the dataframes to get the User-ID, Book-id (ISBN) and rating in one dataframe. Since the ALS training procedure needs integer inputs, I had to typecast all the fields into an integer as part of the transformation procedure. The transformed data frame was then taken as the training dataset.

```
>>> train=ratings.sort('ISBN', ascending=False)
```

```
>> train.show()
+----------+-------+-----------+
      ISBN|User-ID|Book-Rating|
+----------+-------+-----------+
2130530508|  14232|          7|
2130530508|  33943|          0|
2130530508|  56039|          0|
2130530508|  82142|          0|
2130530508| 106954|          0|
2130530508| 131441|          5|
2130530508| 183287|          0|
2130507395| 166534|          0|
2130505325| 216157|         10|
2130494838| 187805|          5|
2130490921| 139480|          6|
2130483410| 188084|          0|
2130467784| 255846|          0|
2130457622| 170158|          0|
2130457193| 166363|          6|
2130453066| 159218|         10|
2130451322| 166363|          8|
2130451233|  14232|          8|
2130451039| 189536|          0|
2130431526| 251782|          6|
+----------+-------+-----------+
only showing top 20 rows

>>
```

- ***Test Set:*** Created test set for a user already existing in training set by entering every book which was rated more than 10 times. Now we had a list of books corresponding to a user so that its rating could be predicted according to the training set ALS model.

```
...
>>> ratingCounts = train.groupBy("ISBN").count().filter("count > 10")
>>>
>>> # Construct a "test" dataframe for user 0 with every movie rated more than 100 times
...
>>> test = ratingCounts.select("ISBN").withColumn('User-ID', lit(14232))
```

```
>> test.join(df_BX_Books, on=['ISBN'], how='inner').select("ISBN","Book-Title","User-ID").show()
+----------+--------------------+-------+
      ISBN|          Book-Title|User-ID|
+----------+--------------------+-------+
2070419878|Novecento : Pianiste|  14232|
2070419657|               Soie|  14232|
2070725804|      Comme un roman|  14232|
2070757625|Balzac et la peti...|  14232|
2070416801|Balzac Et LA Peti...|  14232|
2070518426|Harry Potter a l'...|  14232|
2070744833|LA Premiere Gorge...|  14232|
2070417948|La Jeune Fille a ...|  14232|
2070513289|Le Petit Prince (...|  14232|
2070528189|Harry Potter et l...|  14232|
+----------+--------------------+-------+

>>
```


- ***ALS Model:*** Created an ALS collaborative filtering model from the complete dataset and fit it on the training dataframe. After fitting the model, we transform it on the test dataframe to get the recommendations. I then transformed the recommendation information into a data frame to display the results on screen.

- ***Parameter Tuning:***
  We can tune some parameters in this function to ensure good performance of the model:

  - ***Rank*:** It is the number of latent factors in the model
  - **maxIter:** It is the maximum number of iterations to run (defaults to 10).

- **regParam:** It specifies the regularization parameter in ALS (defaults to 1.0).[8]

```
>>>
>>> from pyspark.ml.recommendation import ALS
>>>
>>> als = ALS(rank=25, maxIter=5, regParam=0.09, userCol="User-ID", itemCol="ISBN", ratingCol="Book-Rating",nonnegative=True)
>>>
>>> model = als.fit(train)
20/04/24 00:31:35 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
20/04/24 00:31:35 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
>>> recommendations = model.transform(test)
>>>
>>> topRecommendations = recommendations.sort(recommendations.prediction.desc()).take(20)
>>>
>>> topRecommendations=spark.createDataFrame(topRecommendations)
>>>
```

3.5 Preserve

Data preservation is an important part of whole data lifecycle as it ensures that we have a way to store huge amount of data, manage it , make it accessible, process it in new ways and deliver the results in production [2]. In this project, I have created a data lake repository in the Hadoop distributed file system environment (HDFS) to ensure that the data is preserved in the native format so that it could be used for exploratory or analysis purposes whenever needed. Hadoop is one of the preferred data platforms due to its scalability, low price and security, however there are a lot of other platforms that could be used for this purpose. The security measures in the data lake may be assigned in a way that grants access to certain information to users of the data lake that do not have access to the original content source [3]. Through this functionality, data will be at your disposal if you have the correct access which enables to keep the data in storage so it can be repurposed repeatedly as new business requirements emerge for the lake's data.

3.6 Publish

Data preserved in data lakes as no meaning if it cannot be queried or if the user can't get a hold of it for further visualization. Spark has a lot of connectors to data visualization formats such as Tableau. Spark comes with a lot of pre-packaged connectors within the Spark SQL API [5]. We have used the "write" connector which makes it very easy to write a dataframe to CSV using a single line of code. The CSV extracts obtained from this command were further used for visualization. The following screenshot is the example in which I am filtering the data of interest and then extracting it in CSV format with the help of Spark API.

```
>>> df_BX_Books_filtered=df_BX_Books.filter(df_BX_Books["Book-Author"].contains('Agatha Christie'))
>>>
>>> df_BX_Books_filtered.write.format('com.databricks.spark.csv').save('prigupt/book_join_auth.csv',header = 'true')
>>>
```

4. Results

The key functionality of a recommendation system is to provide an appropriate recommendation to a particular user according to his likes and dislikes collected in the form of data. As part of the predictive analysis, until now we have used ML Lib's ALS function to train a model and come up with the predictions for the user in the test set. The following are the top book recommendations for the user:

```
>>> topRecommendations.join(df_BX_Books, on=['ISBN'], how='inner').sort('prediction', ascending=False).limit(10).select("ISBN","Book-Title","prediction").show
()
+----------+--------------------+--------------------+
|      ISBN|          Book-Title|          prediction|
+----------+--------------------+--------------------+
|2070518426|Harry Potter a l'...|   6.048744201660156|
|2070528189|Harry Potter et l...|  2.9110639095306396|
|2070416801|Balzac Et LA Peti...|   2.747732400894165|
|2070744833|LA Premiere Gorge...|  2.5517282485961914|
|2070725804|      Comme un roman|  2.4498910903930664|
|2070419657|                Soie|  2.2631402015686035|
|2070417948|La Jeune Fille a ...|   1.642556071281433|
|2070419878|Novecento : Pianiste|  0.3557487428188324|
|2070513289|Le Petit Prince (...| 0.04037068039178848|
|2070757625|Balzac et la peti...|3.931049141101539E-4|
+----------+--------------------+--------------------+
```

The book title gives us the name of the recommended book and prediction column gives us the predicted rating of the book from the perspective of the user. So, after implementing this algorithm we get the predicted rating for every book present in the testing set and we can make the recommendations according to the top-rated books. Some recommendation systems choose to provide new users with a set of default ratings (e.g., an average value across all ratings), while others choose to provide no ratings for new users. Spark's ALS algorithm yields a NaN (Not a Number) value when asked to provide a rating for a new user.[6]

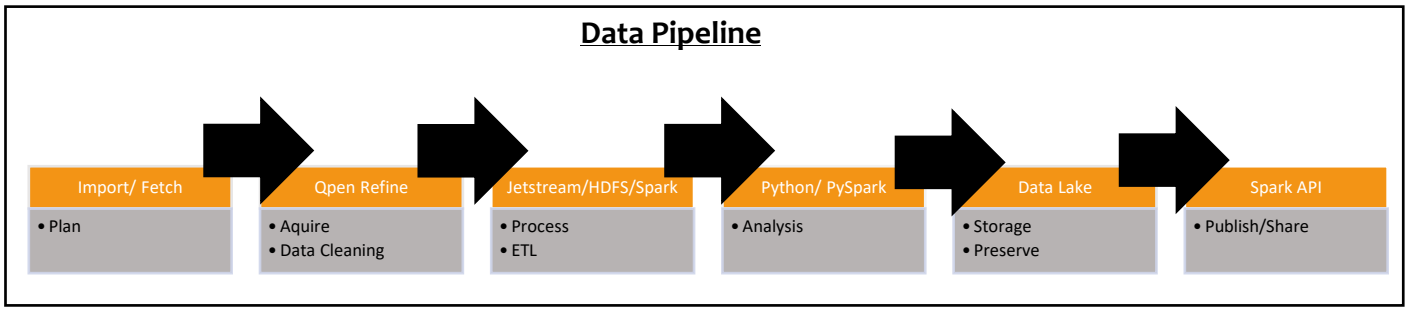
Evaluation Metrics: Collaborative filtering is an unsupervised learning algorithm hence; we do not have any labels to compare our predicted ratings to the expected ratings. There are ways to evaluate your prediction by the method of cross-validation by comparing the root mean square error. Using the ML Pipeline's CrossValidator with ALS is problematic because cross-validation involves dividing the training data into a set of folds and then using those folds for testing and evaluating the parameters during the parameter grid search process. Some of the folds will likely contain users that are not in the other folds, and, as a result, ALS produces NaN values for those new users. When the CrossValidator uses the Evaluator (RMSE) to compute an error metric, the RMSE algorithm will return NaN. [9]
There are workarounds to this problem by RMSE dropping NaN values, thus this may help to evaluate our model to some extent.

5. Discussion

Interpretation of Results: A recommendation system aims to find out how users and items are related to one another. This is something which we have achieved by using matrix factorization Essentially, we take a large matrix of user/item interactions and figure out the latent (or hidden) features that relate them to each other in a much smaller matrix of user features and item features. To approximate this factorization ALS is used for which we only need to solve one feature vector at a time, which means it can be run in parallel. This is one of the advantages because of which this exclusive algorithm has been chosen for Spark ML Lib. The dot product of the factors acquired by this process then results in the predicted rating which is for a specific user-item interaction even if there is no prior interaction.[7]
You can see this in the results obtained by my recommendation system. The test data was created with a user-id already present in the training set along with the random books picked from the dataset to which the user had no prior interaction. On the transformation of test on the trained model, we were able to get the predicted ratings for the users. According to our results, the user is most like to read “Harry Potter” from the pool of books that were given during testing. According to this strategy we can recommend the “Harry Potter” book to this user in future which can promise increase reader base.

Data Pipeline:

As we have seen so far, to ensure the integrity of data we need to go through a series of operations like collect, enrich and process data, analyse etc. Till now I have explained the whole data life-cycle, now I would like to conclude the data pipeline that I have followed in this project.



Another approach of doing this project could have been simply using Pandas to the extract the data and then using Jupyter Notebook to implement the recommendation algorithm but I preferred to use what I have learnt in this course as the main attraction of my project. Using the Hadoop distributed file system ensures that we handle Volume, Variety, Velocity & Value of data. With Hadoop, I can extract the data and can carry out distributed processing of large data set on a cluster. This gives the capacity to my project to become scalable in future as large data sets across hundreds of inexpensive servers that operate in parallel [8]. It does not only ensure fast processing of data but also gives it the power to be resilient to failures as data sent to a node is also replicated to other nodes of a cluster.

Using Apache Spark in this project made it possible for me to work with interactive queries and iterative algorithm efficiently. The inbuilt machine learning library helped me to implement the algorithm with ease and gave me good analytics experience. There are other functionalities in Spark which could be further explored, like the use of resilient distributed dataset which gives fault tolerance capabilities to the system because of immutable primary abstraction. Graphx libraries provided by Spark gives a good graphical analytics experience.

Challenges Encountered:

While the planning phase, collecting the data and decision of the resources to be used at each stage was driven towards handling big data by using a distributed system, however, I faced some unanticipated issues through this project due to which some approaches had to be changed from real world scenario.

- **Slow data cleaning process through Open Refine**

  Since the data I was working with was large, I encountered the problem of its slow processing while using Open refine for cleaning. I looked up for resources to deal with this issue, and found that by assigning more memory to Open Refine, it can work better. After assigning the memory it worked better but yet I could only make common cleaning of the data. I found alphanumeric entries in some fields of the data but refining it took a lot of time hence I could make limited changes.

- **Limited Space in Jetstream Instance**

  Since during this course, I got well acquainted by the usage of Jetstream VM, I continued to use it during this project. However, I encountered problems while training the ALS model as the dataset had more than 2 million records and the space allocated to my instance was limited. It was this time when I started exploring the **Hortonworks framework for Hadoop and Apache Spark**, although it was pretty similar to Jetstream VM, but due to my limited memory in my local machine, the VM faced some unresolvable issue. Therefore, I decided to train the model on filtered data on Jetstream VM for a smooth workflow.

- **Spark Configuration Setting change**

  It was during the creation of the test dataset that I encountered some issues for which I had to make changes to the Spark configurations. For example, joining the tables gave an exception, to resolve it I had to debug the issue and change the spark configuration by setting "spark.conf.set("spark.sql.crossJoin.enabled", "true")".

6. Conclusion

I have illustrated how to build a scalable collaborative filtering book recommendation system on Apache Spark and Hadoop. I have presented a step-wise understanding of every technology used while building this system. I have introduced the core concept of using ALS on Spark ML Lib and have provided implementation for the same. In the end we were able to get books recommendation for a particular user by getting predicted rating of books according to his taste. This process can be done for entire customer base.

7. References

[1]https://searchdatamanagement.techtarget.com/definition/Hadoop-Distributed-File-System-HDFS

[2]https://earlyadopter.com/2018/03/09/saving-your-data-lake/

[3]https://www.searchtechnologies.com/blog/search-data-lake-with-big-data

[4]https://tdwi.org/Articles/2017/10/16/ARCH-ALL-Data-Lake-Manifesto-10-Best-Practices.aspx?Page=2

[5] https://medium.com/sicara/publish-data-outside-datalake-spark-connector-f1684a736422

[6] https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2799933550853697/2823893187441173/2202577924924539/latest.html

[7] https://jessesw.com/Rec-System/

[8] https://spark.apache.org/docs/2.2.0/ml-collaborative-filtering.html

[9] https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2799933550853697/2823893187441173/2202577924924539/latest.html

[10] https://www.youtube.com/watch?v=l8jMvfOU3vQ&list=PL6cactdCCnTJ2XZYIQYIwLperpbKB86jv

[11] https://medium.com/@patelneha1495/recommendation-system-in-python-using-als-algorithm-and-apache-spark-27aca08eaab3

[12] http://www2.informatik.uni-freiburg.de/~cziegler/BX/ (Dataset)