## Introduction

In recent times, society operates in such a way that a large amount of information becomes available to users. Due to this information overload, it becomes increasingly difficult for users to gain access to information that catches their interest. This explains why many companies have developed recommender systems to connect users and information accurately. Kholia (2020) explained that a recommender system is a simple algorithm that supplies the most important information to users by finding out dataset patterns. Recommendation systems are effective and efficient tools for interaction with large and complex information systems. The system is often used for large data to find certain information, sort out and give forms to a large amount of information. (Ma 2016; Felfering & Burke; 2008). Various techniques are also used in a recommendation, but the most prominent method used is the collaborative technique. (Schafer et. Al 2007). This method is widely used and accepted because of its simplicity.

In collaborative recommendation, computing utility issues are changed into the problem of extracting missing values in the rating matrix. It is often encouraged for its simplicity, justifiability, efficiency, and stability among others (Kabore, 2012). Among the different types of collaborative filtering, item-based collaborative filtering identifies similar items based on users’ previous ratings, hence, its application for movie recommendation. Movie recommendation is important in increasing the commercial values and acceptability of various movies by consumers. A study by (Ma 2016) suggested that Netflix declared that 60% of their users search for and locate movies of their interest by the recommender system. This could also find applicability in anime recommendations. The study aims to provide information on the recommendation of the best anime for users and assist in further recommendations on anime that users can watch. This study seeks to use the collaborative filter-based recommendation system to suggest anime titles for users.

## Approach

The analysis was done using the pyspark application on python. The requirements for using the data were satisfied and the files were imported before building a model to extract useful recommendations. Implementation was done in google colab notebook that utilizes Apache Spark and Pandas data frame. The original data for anime and rating data was gotten from Kaggle and myanimelist.net.

“from google.colab import drive

drive.mount('/content/drive/')”

ApacheSpark and Pandas were first imported as seen below:



## It then became possible to import, read, clean data and analyse the files

## anime = spark.read.csv("/content/drive/MyDrive/Project Rector/anime.csv", header = True, inferSchema = True)

## rating = spark.read.csv("/content/drive/MyDrive/Project Rector/rating.csv", header = True, inferSchema = True)

## 

## The steps were highlighted below:

## Removal of all ratings with -1 as values where -1 stands for user haven watched the movie but not rated it. It is assumed that animes with a -1 rating give no useful information as this added no value to the result.

## Renaming the “name” column in anime data to anime\_title before merging and also renaming “rating” in rating data to user\_rating as both anime and rating data have the rating column.

## Merging both data using inner join on the primary key (anime\_id) which is unique to both anime and rating data. This was done by using the join() function.

## Many symbols were found in anime\_title. Those symbols were removed using a function.

## All the rows with one or more NaN values were dropped as they do not aid the process of building a recommendation system.

## The merged data were split randomly into training and test sets to get the RMSE score to measure the model's performance. The data was split into 80% training and 20% test sets.

## ALGORITHM

## The recommendation system is built by using the “Alternating Least Squares (ALS)” on the training data. ALS is a matrix factorization algorithm operating in a parallel manner by itself. The operationalization of ALS is in Apache Spark ML. It is developed for large-scale collaborative filtering problems. ALS works well by solving scalability and sparseness of the Rating data coupled with simplicity and good scalability for a dataset that are large. The cold-start strategy was set to drop to ensure that NaN evaluation criteria are not gotten. The training data fit for the ALS model while the test data was used to generate predictions. The model was then evaluated using the root mean square regression evaluation.

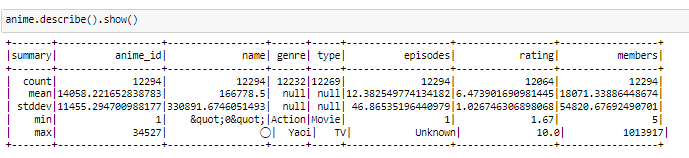
**RESULTS**

***Q1. Description and Identification of the number of columns in the two dataset files.***

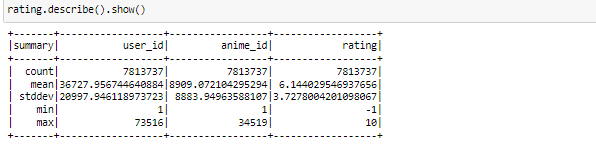
**The rating and anime dataset were identified using the pyspark command anime.printSchema() & rating.printSchema()**

The description for dataset anime was seen below

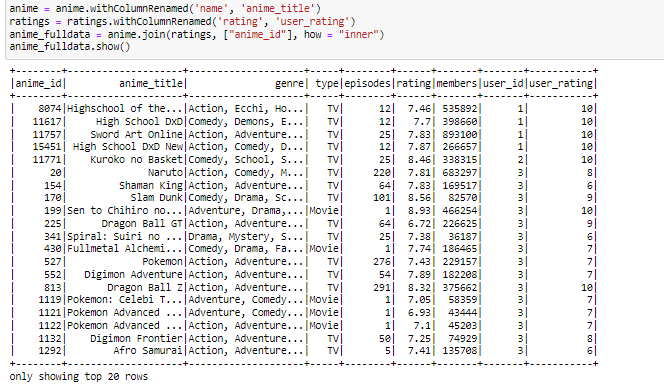
FOR ANIME DATA



FOR RATING DATA

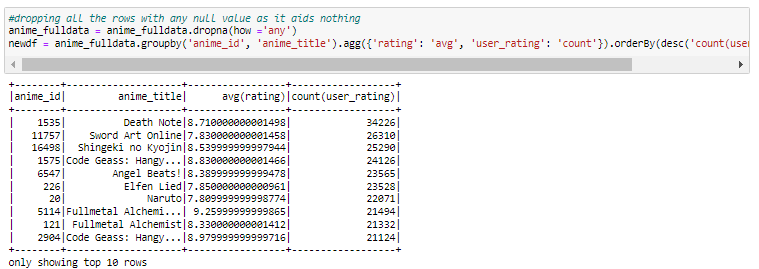


**Q2**.  Merging the two datasets and identifying key common columns being performed?

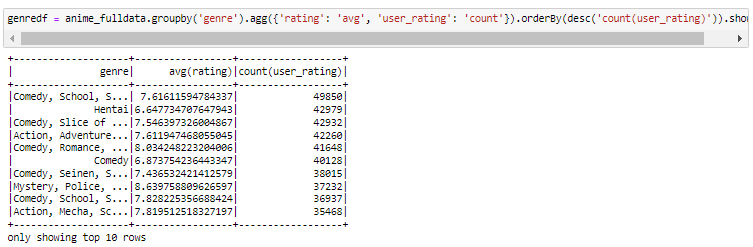


**Q3. Use** graph tofind the top 10 anime based on user rating.

All the rows that contain any null values were dropped. The user rating was then used to find the top 10 animes from the merged data. The output of the result is presented below



**Q4.** Presentation of the top 10 genres based on user rating.



Q5. Designing a collaborative filter-based recommendation system.

Before building the recommendation system, users who rated animes only once were not considered even if they have rated it 5 as that was not deemed valuable enough for the recommendation. A minimum of 200 ratings was considered as the threshold value for each unique user.



The root means square error value based on the model evaluation on the test set is 1.2402.

Q6. Example of best three anime recommendations for a given number of users.

To get this, the analysis was run with the code below:

**user\_x = test.filter(test['user\_id'] == x).select(['user\_id', 'anime\_id', 'anime\_title'])**

**recx = model.transform(user\_x)**

**recx.sort(desc("prediction")).show(3)**

**where x is the user Id**

**x = 5, 139, 210, 233, 250, 271, 308, 593, 572 &497**

## References

Felfering A. and Burke, R., 2008. Constraint-based recommender systems: technologies and research issues. In Proceedings of 10th International Conference on Electronic Commerce) IECE’08) ACM. New York, NY Article3.10pages

Kabore, S. C. 2012. Design and Implementation of a recommender system as a module for Liferay portal. Master’s Thesis at Barcelona School of Computing University Polytechnic of Catalunya

Kholia T. 2020. Movie Recommendation System, Data Analytics and Machine Learning. Project Report

Ma, K. 2016. Content-based Recommender System for Movie Website, Skolan for Information Master’s Thesis at VionLabs

Schafer, J., Frankowski, D., Herocker, J. and Sen, S. 2007. Collaborative filtering Recommender Systems. In Brusilovsky.Kobsa and NejdL(eds.) The Adaptive Web. Pages 291-324 Springer