Movie Analytics for Effective Recommendation

Anusha Mettu
Computer Science
UMKC
Kansas City, Missouri, US
am2q5@umsystem.edu

ABSTRACT

Due to the exponential growth of video streaming services, the volume of video content has significantly increased, posing a challenge for end consumers to find the films they want. To address this issue, a hybrid approach utilizing collaborative filtering and Apache Spark & ML frameworks was employed to create a recommendation system that is both accurate and trustworthy. The ALS paradigm was used to develop a collaborative filtering technique for filtering films or movie content based on the highest ratings. In order to provide users with a combination of the highest rated suggestions, our proposed method takes into account a user's most recent search information about a movie category. The study addresses sparseness and scalability issues through the use of a matrix factorization model that employs the ALS filtering technique and relies on content.

KEYWORDS

Video streaming services, recommendation systems, Apache Spark, machine learning, filtering-based, highest-rated, movie category, ALS filtering technique, collaborative filtering technique.

1. INTRODUCTION

The accessibility of the Web is continuously increasing, leading to a daily growth in the number of web users. Consequently, there is an increase in the volume of information transmitted via the Web. This overload of data has resulted in users being overwhelmed with knowledge. The rapid growth of data has given rise to a new era of knowledge, leading to the creation of innovative, efficient, and effective systems. Among these systems is the recommender system, which utilizes a filtering-based approach to search queries based on a user's search history. Forecasting techniques are utilized to predict the demands of an audience or the rating they would give to a particular important product. These recommender systems can be found on various websites, where they collect and extract data from disparate and diverse sources.

In film or any recommendation system (RS), the primary approach is sequential, where viewers first rate different products and the algorithm then predicts ratings and reviews for an item that has not yet been evaluated. These updated estimates are based on the ratings of the most active users and the current ratings. Collaborative filtering and content-based algorithms are commonly used to accomplish this. With the collaborative filtering (CF) method, customer applications can replicate customized recommendations more quickly and effectively.

Big data is a driving force behind recommendation engines as it enhances their efficiency. As the volume of data increases, it becomes more challenging to aggregate information from multiple sources. To address this, we created a recommendation engine using an ALS model and matrix factorization-based approach. We extensively utilized the MLlib APIs for Spark-based machine learning. This comprehensive machine learning package supports various methods, such as clustering, classification, pattern recognition, Spark ML, and recommendations.

2. RELATED WORK

Content-based recommendation systems can be prone to being fooled, which is why a hybrid machine learning program was developed using Apache Spark. This program is designed to improve the efficiency of the recommendation system through parallel processing. This approach utilized both structured and unstructured data, while reducing the usage of superfluous data

through hybrid and customer cooperation. Major issues, including flexibility, delayed start, meagerness, and unaccountability were addressed, and the technique for massive data can aid in scaling. To address the freezing start issue and provide a potential recommendation system, multiple permutations of collaborative filtering algorithms were utilized. The author of this work explains the consumer and product modes of Spark operation with coupled techniques. Two additional computational strategies were used to overcome the constraints of insufficiency, sparseness, and flexibility. As the directions for recommender systems are rapidly evolving, the development of frameworks (R and Spark) will be explored in the future. The global integration of Spark and R formats will result in the creation of a Spark that offers newly gathered data (Khadse, Basha, Iyengar, & Caytiles, 2018).

The K-means and ALS techniques, along with the Lambda architecture, are the most commonly used architectures discussed in this work. The study reveals that while Spark's machine learning tools are powerful, the drawback is that the dataset requires linear evaluation. To reduce the dataset size, MSE and WCSS can be utilized, and by evaluating these two approaches, good results can be achieved. If cluster growth occurs, more adjustments may be required. The algorithms were tested and evaluated, along with the SARF technique built on Spark, which connects queries to clients with accurate replies and higher potential (Chen, Yuan, Yang, & Zahir, 2021).

The study showcased the application of numerous Spark machine learning libraries with the ALS model, which generated the desired recommendations via the recommender system, and presented an impressive approach for the era of big data (Spark and ML lib Related research or a methodology CF was just used as a filter). The ALS model was specifically utilized in this research to predict movies. This study demonstrated an effective technique for obtaining top recommendations in the big data era using recommender systems. By utilizing numerous Spark machine learning libraries coupled with the ALS model (Spark-ML), the model outperformed the previous model by reducing the RMSE by approximately 0.97613 (Khalid & Zebaree, 2021). Moreover, it achieved a 97% accuracy score, which surpassed the state-of-the-art accuracy of 91%. The movie lens data was used for performance analysis and evaluation, and the study showed that the ALS model and collaborative filtering analytics effectively resolved the frequent sparseness issue.

The researchers utilized Naive Bayes and ML algorithms in Spark MLlib to predict Amazon food, Black Friday sales, and stock market reviews in 2021. They achieved approximately 90% efficiency with a decent time frame. The study developed a system that demonstrated how important data could be extracted from large datasets by using various datasets. The experimental results indicated that precise models could be created to generate accurate outcomes with vast datasets. The study investigated four different recommendation system models that combine user similarity and balanced transmission. However, both the accuracy of singular value decomposition and the backpropagation efficiency fell below 70%, and the only issue was the high value of the ideal rating recommendation. In another study, scalability issues were examined and resolved using Hive, which detects similarities and differences. The problem of scalability was eventually resolved by utilizing the collaborative approach, which provided faster execution speeds than the sequential approach due to the prevalence of non-parallelism in this scenario. The results showed that the multi-criteria methodology using Spark had significantly faster performance than the non-parallel approach. Execution time, speedup, and system extensibility were measured for each node. The only obstacle to improving network communication efficiency was the knowledge gap that needed to be addressed (Alam, 2018).

Our proposed system utilizes Spark's big data architecture to perform filtering based on customer and product categorization. Despite handling a large amount of data, we applied the ALS technique to categorize ratings, focusing on the product and customer matrix, as well as latent attributes. By multiplying the subdivided matrices and merging them, we created a new matrix that could potentially encounter a frozen beginning issue, as it may lack an item reflecting the movies and forecasts. To address this, the system utilizes user and product characteristics to fill in any missing records. According to research observations, the proposed methodology yields productive and effective outputs, offering quick forecasts of the best movies. Future research could delve deeper into mining-related techniques to gain further insights into customer perspectives.

3. PROPOSED TECHNIQUES AND TOOLS

3.1.1 Machine learning

Machine learning, a field of computer science and artificial intelligence, employs various techniques and data to simulate human learning and improve system accuracy over time. Recent developments in processing speed and storage capacity have enabled the creation of machine learning-based services such as self-driving cars and recommender systems.

3.1.2 Pyspark

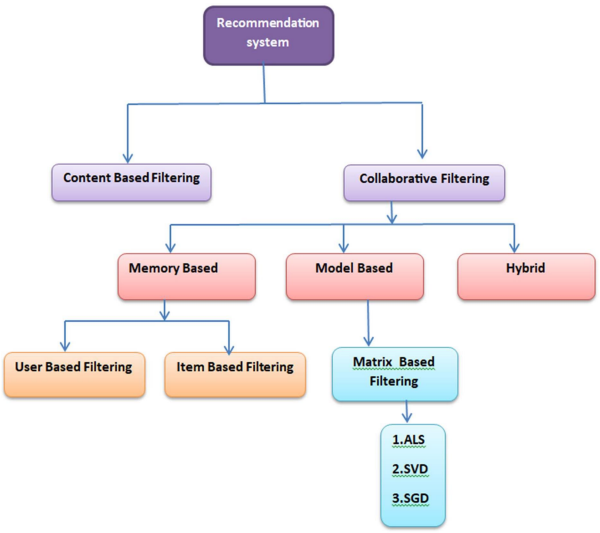
The Python interface for Apache Spark, also known as PySpark, enables the creation of Spark applications using Python APIs. This interface enables Python programmers to connect with Spark clusters, allowing for communication between Python and Spark. Spark is a widely used large-scale analytical processing engine in the data science and machine learning communities, with Spark Streaming, Spark Machine learning libraries (MLLib), and Spark SQL making up the stack for both batch and real-time data processing.

3.1.3 Machine Learning Algorithms using Pyspark

Machine learning, a type of AI, allows computer programmers to produce accurate forecasts without being explicitly programmed to do so. By using historical data as input to predict future outcomes, machine learning algorithms have become a crucial tool for developing new products and gaining insight into consumer behavior and market trends. Many of the world's most successful companies, such as Uber and Google, heavily rely on machine learning in their daily operations, giving them a competitive advantage over their competitors. Machine learning has become a significant factor in setting businesses apart from their rivals in today's world.

4. Data and Methodology

This paper employs the collaborative filtering technique with the



ALS algorithm to predict top-rated movies that can be recommended to users. The ALS algorithm is used to factorize a large matrix into a smaller representation matrix.

Fig 1: System Architecture

a. Dataset

The dataset provided below served as the basis for constructing several models. By considering this dataset, similar models can be developed, trained, and validated to enhance accuracy.

This dataset consists of the following files:

Source	Information about Dataset and statistics
Kaggle	https://www.kaggle.com/rounakbanik/the-movies-dataset
Movies_ .csv	The main file for Movies Metadata. It includes data on 45,000 films included in the Full Movie Lens dataset. Posters, backdrops, finances, earnings, release dates, language support, producing nations, and companies are among the features..
Keywords.csv	Contains the movie plot keywords for the films we watch with Movie Lens. They are accessible as JSON Object that has been stringified.
Credits.csv	Includes cast and crew bios for all of our films. Accessible as a JSON Object that has been stringified.
Links.csv	The data file contains all of the movies listed in the Full Movie Lens dataset's TMDB and IMDB IDs.
Rating.csv	The 45,000 films in this dataset's Full Movie Lens Dataset total 26 million ratings and 750 000 tag applications from 270,000 users.

b. Movie lens Dataset

The Movie-Lens 25M movie ratings dataset, which includes 25 million ratings and one million tag applications from 162,000 users for 62,000 movies, provides a reliable benchmark for building various models. Additionally, the tag genome data, consisting of 15 million relevance scores across 1,129 tags, is included. While sites that offer movie recommendations use standardized rating and reference systems to categorize films based on their content, these systems are inadequate for recommendation systems on social networking sites.

c. Collaborative Filtering

The recommendation system employs collaborative filtering, and matrix factorization is used to predict unknown information. The study shows that using Spark-ML and ML-libraries with a model-based collaborative filtering approach can predict incomplete information when modeling consumers or products with latent features. To learn the latent factors, the ALS algorithm was implemented.

d. ALS Algorithm

This work is based on artificial intelligence and the ALS algorithm, which is a form of regression analysis comparable to regression analysis in that regression provides a line with pieces of data that are utilized to construct via regression. The distance's sum of squares reduces the separation from some of the supplied data sets in this way. These lines are employed to forecast the value of the function, which is then satisfied by the independent variables.
Alternating Least Squares (ALS) matrix factorization attempts to estimate the ratings matrix R as the product of two lower-rank matrices, X and Y, i.e., X * Yt = R. Typically these approximations are called 'factor' matrices.

4.1 IMPLEMENTATION

a. Technologies and Tools

TABLE -1 : Standards & Minimum Specifications:

Software specifications	Configuration details
Python Libraries	NumPy, Pandas, SKlearn, Pyspark, vega, Matplotlib
Dataset	Movielens Dataset (Kaggle)

b. Setting up and Importing the Required Packages

We can use the Google Colabs! pip install commands to install packages like pyspark, Vega-datasets, JSON, etc.

```
!pip install -U vega_datasets notebook vega
```

```
!pip install ujson

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: ujson in /usr/local/lib/python3.7/dist-packages (5.5.0)

%env JOBLIB_TEMP_FOLDER=/tmp
!pip install pyspark
```

Fig 2: Install packages

After building the required modules, import the necessary libraries. Matplotlib, seaborn, and plot are some of the imported libraries used for data visualization, whereas PySpark is an open-source Python modeling package.

```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 8)
import os
import gc
import ujson as json
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.patches as patches
import seaborn as sns

import plotly as py
# import plotly.express as px
import plotly.graph_objs as go
from plotly.subplots import make_subplots
from plotly.offline import download_plotlyjs
from plotly.offline import init_notebook_mode
from plotly.offline import plot,iplot

init_notebook_mode(connected=True)

import altair as alt
from altair.vega import vs
from IPython.display import HTML
alt.renderers.enable('notebook')

from IPython.display import HTML
from IPython.display import Image
from IPython.display import display
from IPython.core.display import display
from IPython.core.display import HTML
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import warnings
warnings.filterwarnings('ignore')

plt.style.use('seaborn')
color_pal = [x['color'] for x in plt.rcParams['axes.prop_cycle']]
%config InlineBackend.figure_format = 'svg'
th_props = [('font-size', '13px'), ('background-color', 'white'), ('color', '#666666')]
td_props = [('font-size', '15px'), ('background-color', 'white')]
styles = [dict(selector="td", props=td_props), dict(selector="th", props=th_props)]
```

Fig 3: Background color plots

c. Importing Dataset

We read the Movie lens dataset using the NumPy and Pandas packages.

```
ratings = pd.read_csv(DATA_PATH+'ratings.csv')
links = pd.read_csv(DATA_PATH+'links.csv')
metadata = pd.read_csv(DATA_PATH+'movies_metadata.csv')
```

Fig 4: Reading CSV files

Then, using statistical representation, we do exploratory data analysis to find various trends that are present in the dataset.

d. Exploratory Data Analysis

The initial step after reading the dataset is to ascertain its size, shape, and other characteristics that will be used in the dataset.

```
get_df_info(ratings)
```

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523

```
'Number of Rows: 26024289, Number of Columns: 4'
'Data Types'
```

	Column	Type
0	userId	int64
1	movieId	int64
2	rating	float64
3	timestamp	int64

```
'memory usage: 794.2 MB'
'Missing Values'
'No Missing Values'
```

Fig 5: Get Info of ratings

It is to get info from the links obtained and is shown in the below figure.

```
get_df_info(links)
```

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0

```
'Number of Rows: 45843, Number of Columns: 3'
'Data Types'
```

	Column	Type
0	movieId	int64
1	imdbId	int64
2	tmdbId	float64

```
'memory usage: 1.0 MB'
'Missing Values'
```

	Column	Missing Values
0	tmdbId	219

Fig 6 : Get Information of links

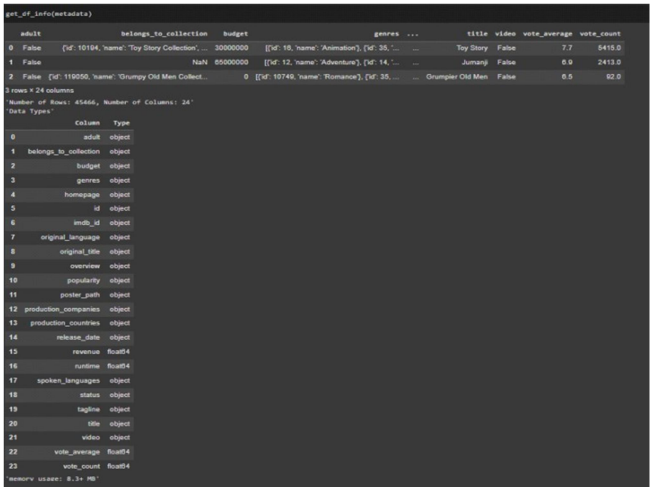


Fig 7 : Get Information of metadata

Since the dimensions of the datasets are found we need to check for any other anomalies in the dataset which might hinder the modelling process. Such checking for class imbalance over the rating frequency if all the movies are equally rated then it would be difficult to build a recommender system.

```
plt.rcParams['figure.figsize'] = (7, 6)
plt.hist(ratings['rating'], bins=10);
plt.title('Ratings Count', size=10)
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show();
```

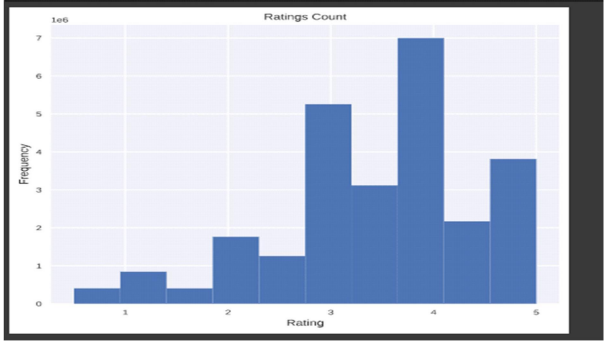


Fig 8: Frequency vs Rating

For a better visualization, let's represent by a pie chart with the percent representation.

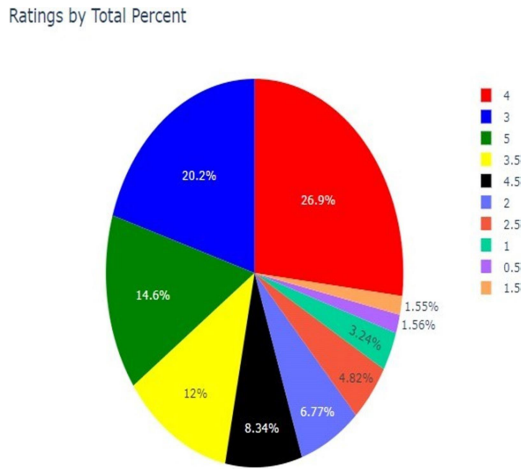


Fig 9 :Pie chart representation by ratings by total percent

Finding the grouped classes with numerous ratings by looking for the feature of ids for each movie rating included in the dataset. The movies with ordered ids are represented below.

```
df_movies = get_movie(df_aux)
df_movies
```

	title	overview	vote_average	release_year
0	Forrest Gump	A man with a low IQ has accomplished great thi...	8.2	1994
1	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	8.5	1994
2	Pulp Fiction	A burger-loving hit man, his philosophical par...	8.3	1994
3	The Silence of the Lambs	FBI trainee, Clarice Starling ventures into a ...	8.1	1991
4	The Matrix	Set in the 22nd century, The Matrix tells the ...	7.9	1999
5	Star Wars	Princess Leia is captured and held hostage by ...	8.1	1977
6	Jurassic Park	A wealthy entrepreneur secretly creates a them...	7.6	1993
7	Schindler's List	The true story of how businessman Oskar Schind...	8.3	1993
8	Braveheart	Enraged at the slaughter of Murron, his new br...	7.7	1995
9	Toy Story	Led by Woody, Andy's toys live happily in his ...	7.7	1995

Fig 10: Get information about movies based on titles

Reducing the number of rows to effectively examine the characteristics further. At first, we got the movie ratings using the user ID, and then, using EDA analysis, the familiar words were gathered from the top 1000 movies. The results are in, and effective analysis is shown below:

```
df_aux = ratings['movieId'].value_counts().reset_index().head(1001).rename(columns={'index': 'movieId', 'movieId': 'count'})
df_aux['movieId'] = df_aux['movieId'].astype(str)
df_aux = get_movie(df_aux)
get_df_info(df_aux)
```

	title	overview	vote_average	release_year
0	Forrest Gump	A man with a low IQ has accomplished great thi...	8.2	1994
1	The Shawshank Redemption	Framed in the 1940s for the double murder of h...	8.5	1994
2	Pulp Fiction	A burger-loving hit man, his philosophical par...	8.3	1994

'Number of Rows: 1000, Number of Columns: 4'
'Data Types'

	Column	Type
0	title	object
1	overview	object
2	vote_average	float64
3	release_year	int64

'memory usage: 31.4+ KB'
'Missing Values'
'No Missing Values'

Fig 11:Get ratings by using movieID

Checking id for each movie and Performing Eda on the titles of the top 1000 movies using nlp so as to find the most frequent words in those movies.

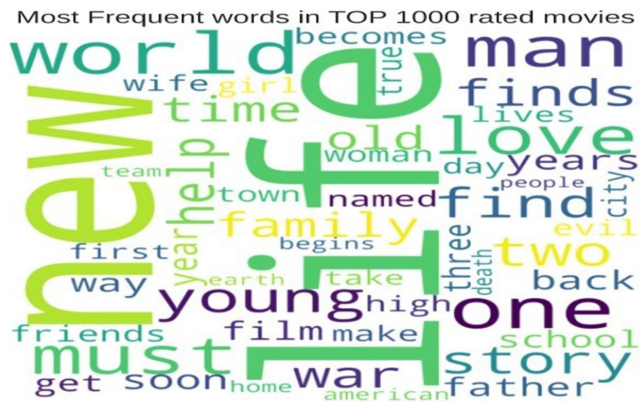


Fig 12 : Most Frequent words in Movies using EDA

Next step, we are going to show recommendations for all users. By querying movie metadata, after setup, we are going to show which movie is going to be recommended for a particular userID.

e. Setup and model

Setting the Spark context and creating the recommendation models

Setting up: The setup method must be called first since it starts the localhost Spark connection and initializes the training environment. All other functions must occur after the setup function. Data and target are two requirements that must be met. There are all the other options.

```
import pyspark.sql.functions as sql_func
from pyspark.sql.types import *
from pyspark.ml.recommendation import ALS, ALSModel
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from pyspark.mllib.evaluation import RegressionMetrics, RankingMetrics
from pyspark.ml.evaluation import RegressionEvaluator

sc = SparkContext('local')
spark = SparkSession(sc)
```

Fig 13: Spark Session

The model is configured to specify the columns directly. Because we are looking at ratings greater than 0, set non-negative to "True." Additionally, the model offers the choice of selecting implicit ratings. Set it to "False" since we are dealing with explicit ratings.

The evaluating set's objects can be found using it that are not in the training set when employing basic random splits, such as those found in Spark's Cross Validator or Train Validation Split. When using the model, Spark by default assigns predictions of NaN. Transform the model when a user and/or item factor is missing. To prevent receiving NaN assessment metrics, we set a cold start strategy to "drop."

```
als = ALS(
         rank=30,
         maxIter=4,
         regParam=0.1,
         userCol='userId',
         itemCol='movieId',
         ratingCol='rating',
         coldStartStrategy='drop',
         implicitPrefs=False
        )
model = als.fit(training)

predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName='mae', labelCol='rating',
                                predictionCol='prediction')

mae = evaluator.evaluate(predictions)
print(f'MAE (Test) = {mae}')
```

Fig 14: Model trained by using ALS algorithm

The model is trained using ALS and parameters used are rank, maxIter,regParam,userCol, items,ratingCol,coldStrategy,implicitPrefs. Model transform is tested and stored in predictions and evaluated using Regression Evaluator.

5. Analysis and Results

Show the most recommended user for each movie (Item Based on Recommendation System). The model shows recommendations choosing 5 userId.

```
model.recommendForAllUsers(1).show(5)

+------+--------------------+
|userId|     recommendations|
+------+--------------------+
|     1|[{101862, 6.740889}]|
|     2|[{133881, 4.52920...|
|     3|[{158832, 4.260253}]|
|     4|[{164937, 5.50250...|
|     5|[{164937, 6.145594}]|
+------+--------------------+
only showing top 5 rows
```

Fig 15: Recommendations for top 5 userId

Now, this displays the movie that was recommended for a particular userId.

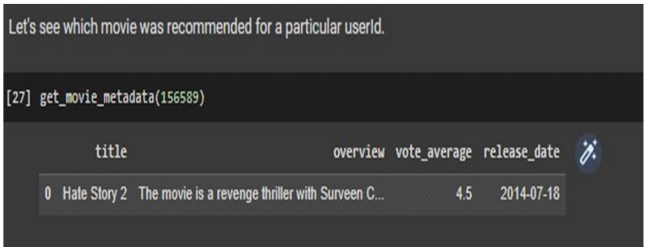


Fig 16: Movie recommended for a specific userID

6. Conclusion

Through experimental research, we proposed a recommendation engine that employs collaborative filtering method to forecast the most popular movies. The dataset from Movie Lens was utilized for performance evaluation and analysis. To address scalability, sparsity, and cold-start issues, we employed CF along with the ALS (alternating least squares) algorithm. The primary evaluator used was MAE, which was recorded at 0.651. Tuning hyperparameters may result in better results for other models implemented.

mistakes and duplicate NaN values in the dataset were difficulties we encountered while processing the data.

We discussed everything once we had shared it amongst ourselves, and documentation is performed by everyone of us collaboratively.

8. REFERENCES

[1] Alam, T., & Awan, M. (2018). Domain analysis of information extraction techniques. . *Int. J. Multidiscip. Sci*.

[2] Rehma, A., Awan, M., & Butt, I. (2018). Comparison and Evaluation of Information Retrieval Models. *VFAST Trans. Softw*.

[3] Amato, F., Moscato, V., Picariello, A., & Sperlí, G. K. (2017). : A system for knowledge-based access to multimedia art collections. *IEEE*.

[4] Chen, L., Yuan, Y., Yang, J., & Zahir, A. (2021). Improving the Prediction Quality in Memory-Based Collaborative Filtering Using Categorical Features. *IEEE*.

[5] Khadse, V., Basha, A., Iyengar, N., & Caytiles, R. (2018). Recommendation Engine for Predicting Best Rated Movies. *Int. J. Adv. Sci*.

[6] Khalid, Z., & Zebaree, S. (2021). Big data analysis for data visualization. *A review. Int. J. Sci. Bus*.