University *of* New Haven

TAGLIATELA COLLEGE OF ENGINEERING

Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science

MSDS Fall-2023

Team - 07

MOVIE RECOMMENDATION ANALYSIS

SPRING 24

TABLE OF CONTENTS

Movie Recommendation Analysis

Executive Summary

The Movie lens dataset is analysed using Spark SQL in this Azure project, and a data pipeline is put up using Azure Data Factory. It involves configuring Azure resources, obtaining data via Databricks and Azure Data Factory, and converting it into Spark data frames. Following that, Databricks makes movie recommendations based on its data research. In order to generate user-friendly movie suggestions, datasets must be extracted, processed, loaded, and visualized.

Team Members:

Gnana Vardhan Siddu Paruvada

Bhargavi Karuku

Janani Ganji

Giri Merugu

Professor :

Dr. Ardiana Sula, Ph.D.

Senior Lecturer in Data Science & Computer Science

Program Coordinator, Data Science

Highlights of Project

Azure Integration: The project utilizes various Azure services like Azure Data Factory, Azure Databricks, and Azure Storage for development and implementation.

Scalability: Thanks to Azure services, the system effectively handles large volumes of data while remaining scalable.

Data Processing: Spark SQL processes data, providing scalable and high-performance analysis of large datasets.

Dataset Overview: The MovieLens dataset from GroupLens contains movie ratings, user profiles, and movie details.

Technological Tools: Python, Spark SQL, and Microsoft Azure services like Azure Data Factory and Azure Databricks are employed.

Data Preparation: Preprocessing involves data cleansing, normalization, and feature engineering.

Evaluation Criteria: Metrics such as accuracy, precision, recall, and F1-score are used to evaluate the recommendation system's performance.

Automation: The data pipeline automates processes like data extraction, conversion, and loading, eliminating manual involvement for efficiency.

Insight Generation: Analysis and visualization of datasets yield insightful information, empowering stakeholders to make data-driven decisions.

Recommendation System: The project's main goal is to develop a personalized movie recommendation system based on user behavior and preferences.

Submitted on : 04/23/2024

Abstract

In this project, a movie recommender system is built on Microsoft Azure through the analysis of the Movie lens dataset using Spark SQL. The first step in configuring Azure to assist with data management and orchestration is provisioning Azure resources, such as storage accounts and Azure Data Factory. The data extraction procedure makes use of both Databricks and Azure Data Factory, providing customers with a choice in how they can access the dataset. Subsequently, the collected data is transformed into Spark data frames for efficient analysis. Movie recommendations are generated using user preferences and movie ratings from Azure Databricks, which is the primary platform for dataset analysis. Another technique for displaying the analysis's findings and providing stakeholders with an understandable image of the data is a bar chart. Bar charts are another tool used to depict the analysis's conclusions and give stakeholders a clear picture of the data trends. This study shows how cloud-based solutions can be used to extract useful insights from big datasets and support well-informed decision-making in the movie recommendation space by utilizing Spark SQL and Azure services.

Introduction

In the current digital era, the amount of data that businesses have access to offers both advantages and disadvantages. Using data to glean insightful information is essential to gaining a competitive edge and making wise decisions. One sector where data analysis is often used is entertainment, particularly when making movie recommendations to customers based on their preferences and behaviour.

In the contemporary landscape of digital streaming platforms, the provision of personalized movie recommendations plays a pivotal role in enhancing user experience. By meticulously understanding user preferences and delivering tailored suggestions, these platforms are able to captivate audiences and foster deeper engagement.

This project harnesses the formidable capabilities of Databricks, coupled with the robust processing power of Apache Spark, to conduct an in-depth analysis of a rich dataset encompassing various facets of the movie industry. This dataset includes information such as movie ratings, genres, tags, and user profiles, providing a comprehensive foundation for the formulation of sophisticated movie recommendations.

Project Objectives :

Data Preprocessing: The project initiates with a meticulous exploration and comprehension of the dataset's structure. Following this, rigorous data preprocessing tasks are undertaken, including the handling of missing values, conversion of data types, and in-depth attribute analysis.

Descriptive Analysis: Uncovering underlying patterns and trends within the dataset is paramount. Through descriptive analysis, the project aims to unveil insights regarding movie ratings, genre preferences, user behaviors, and more. These insights are then effectively communicated through a variety of visualizations such as histograms, pie charts, and bar plots.

Movie Recommendations: Leveraging advanced collaborative filtering techniques, the project endeavors to craft personalized movie recommendations for users. By discerning similarities among users or items based on historical interactions, the project strives to deliver tailored suggestions that resonate with individual preferences.

Genre Analysis: Delving into the realm of movie genres, the project seeks to unravel genre distributions, popularity metrics, and potential correlations. This genre analysis serves as a vital component in refining recommendation algorithms, ensuring that the recommendations provided cater to the diverse preferences.

This project aims to develop a movie recommender system using Microsoft Azure's cloud infrastructure and Spark SQL's analytical capabilities. The popular Movie lens dataset, which contains user information and movie evaluations, serves as the foundation for our analysis. Using Databricks and Azure Data Factory, we build a robust data pipeline for efficiently managing and processing the dataset.

In summary, this project epitomizes the fusion of cutting-edge technology with the captivating world of cinema, offering a compelling journey through the realms of data analytics and movie recommendation systems.

Review of Available Research

In order to provide customers with personalized recommendations, a plethora of approaches and algorithms have been developed in the heavily researched field of movie recommendation systems. The literature contains in-depth analyses of item- and user-based collaborative filtering strategies. These algorithms examine user-item interaction data to identify patterns between people or objects and provide suggestions based on those patterns.

In recent years, more advanced techniques like matrix factorization, deep learning, and hybrid algorithms have gained popularity. Matrix factorization techniques such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS) have gained increasing popularity because of their efficacy in handling sparse and high-dimensional data. Among the deep learning techniques that have shown promise are neural networks.

Another popular study area is hybrid recommendation systems, which integrate several recommendation techniques. These systems seek to overcome the shortcomings of each recommendation method on its own and offer recommendations that are more varied and accurate by utilizing the advantages of those methods.

Moreover, studies on movie recommendation systems have broadened to include contextual data from social networks, temporal dynamics, and user demographics, among other sources. It has been demonstrated that adding this contextual data improves user happiness and the quality of suggestions.

Tools and Technologies

Databricks Notebook:
The project harnesses Databricks notebooks for interactive data exploration, analysis, and visualization, capitalizing on their rich features.

Spark SQL:
Extensive use of Spark SQL facilitates querying and manipulation of structured data. SQL queries executed on Spark DataFrames streamline data operations effectively.

PySpark:
PySpark, the Python API for Apache Spark, is instrumental in data processing and analysis. Leveraging PySpark provides access to Spark's distributed computing capabilities via Python.

Spark DataFrames:
Data is structured and manipulated using Spark DataFrames, offering high-level APIs for scalable and performant data operations.

DBFS (Databricks File System):
DBFS serves as the storage and access solution within the Databricks environment. Its distributed file system facilitates seamless data management from Databricks notebooks.

Data Visualization:
Matplotlib is employed for data visualization tasks, generating static, interactive, and animated visualizations to aid in data exploration and presentation.

Pandas:
In addition to PySpark, Pandas is utilized for specific data manipulation and analysis tasks, leveraging its extensive functionalities for efficient data processing.

SQL Queries:
SQL queries are executed directly within the notebook using SQL magic commands, seamlessly integrated with PySpark workflows via Spark SQL.

Data Cleaning and Transformation:
Various data cleaning and transformation operations, including joining, filtering, aggregation, and type conversion, are performed to prepare the data for subsequent analysis and modeling tasks.

Methodology

Step 1: Data Exploration and Preparation

Exploration:

- Use libraries like Pandas to import the dataset.
- Analyze the data's structure, completeness, and distribution.

Preprocessing:

- Deal with missing data, convert categorical features into numbers, and remove any unnecessary information.

Step 2: Azure Student Account Setup

Account Creation:

- Sign up for an Azure Student Account using your university email.

Subscription Activation:

- Confirm your student status and activate your Azure account.

Step 3: Azure Resources Creation

Login to Portal:

- Go to the Azure portal and sign in with your Azure student account.

Resource Creation:

- Make resources like Azure Data Lake Storage (for data storage), Azure Databricks (for data analysis), and Azure SQL Database if needed.

Step 4: Data Pipeline with Azure Data Factory (ADF)

ADF Instance Acquisition:

- Get a new Azure Data Factory instance set up in the portal.

Pipeline Development:

- Build a data pipeline using ADF to fetch data from storage and move it to Azure Databricks.
- Set up connections for storage and Databricks, create datasets for data sources and destinations, and arrange tasks for data transfer.

Step 5: Model Training with Azure Databricks

Cluster Configuration:

- Start a new cluster in your Databricks workspace with the right settings.

Data Preparation in Databricks:

- Make a new notebook in Databricks for training your model and bring in the preprocessed data.

Model Training:

- Train a recommendation model using algorithms like ALS or SVD.

Step 6: Model Evaluation and Tuning

Model Evaluation:

- Check how well the trained model performs using metrics like RMSE or MAE.

Hyperparameter Tuning:

- Improve the model's performance by adjusting its settings, known as hyperparameters, if needed.

Step 7: Model Deployment

Export Trained Model:

- Save your model in a format that Databricks can understand, like Parquet or MLflow.

Create Deployment Notebook:

- Make a notebook in Databricks to set up your model and bring in the saved model file.

Define API Endpoint:

- Create a way for people to use your model through web frameworks like Flask or FastAPI.

Deploy as Web Service:

- Turn your notebook into a web service on Azure Databricks using databricks-connect and set it to run with Databricks Jobs or Azure Functions.

Step 8: Frontend Development

Build Frontend App:

- Make a web or mobile app that works with your recommendation system.

Connect to Model Endpoint:

- Link your frontend app to your model's endpoint so it can get recommendations and show them nicely to users.

Step 9: Testing and Validation

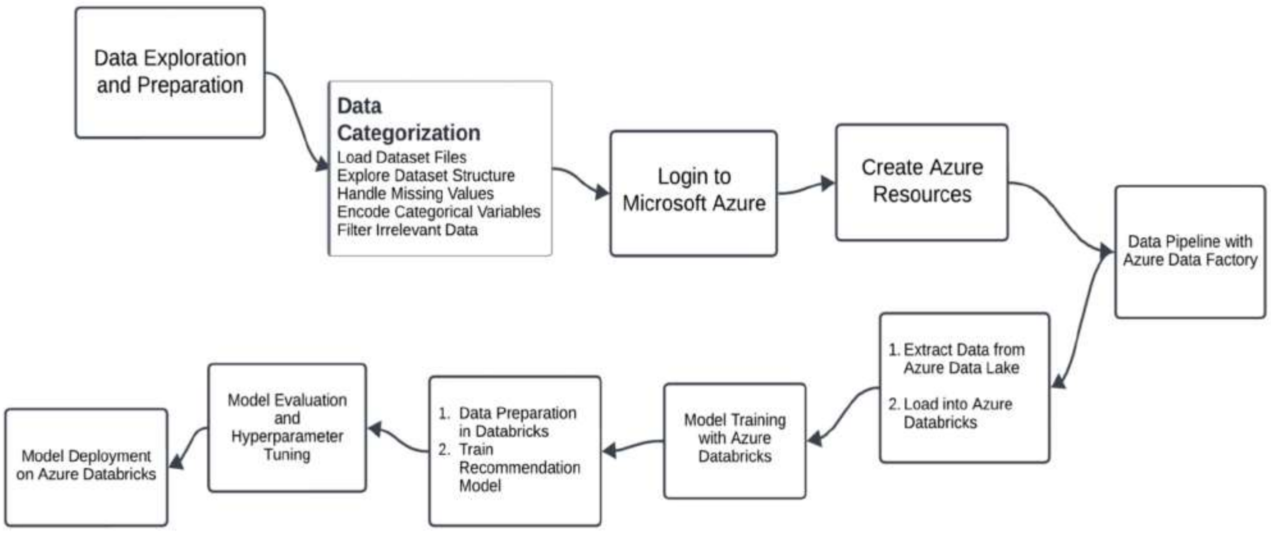
Test Entire System:

- Check how well your whole recommendation system works, including getting data, training the model, deploying it, and using the frontend.

Validate Recommendations:

- Make sure the recommendations your system gives match up with what you know people like.

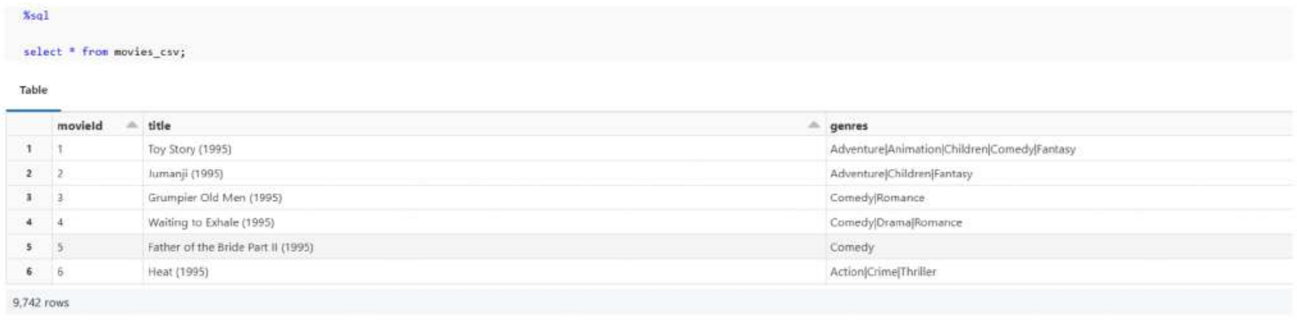
Flow Chart


Data Exploration and Preparation
Data Categorization
Load Dataset Files
Explore Dataset Structure
Handle Missing Values
Encode Categorical Variables
Filter Irrelevant Data
Login to Microsoft Azure
Create Azure Resources
Data Pipeline with Azure Data Factory
1. Extract Data from Azure Data Lake
2. Load into Azure Databricks
Model Training with Azure Databricks
1. Data Preparation in Databricks
2. Train Recommendation Model
Model Evaluation and Hyperparameter Tuning
Model Deployment on Azure Databricks

Results

Dataset Insights: The MovieLens dataset encompasses vast movie ratings and diverse user demographics.

Movies dataset



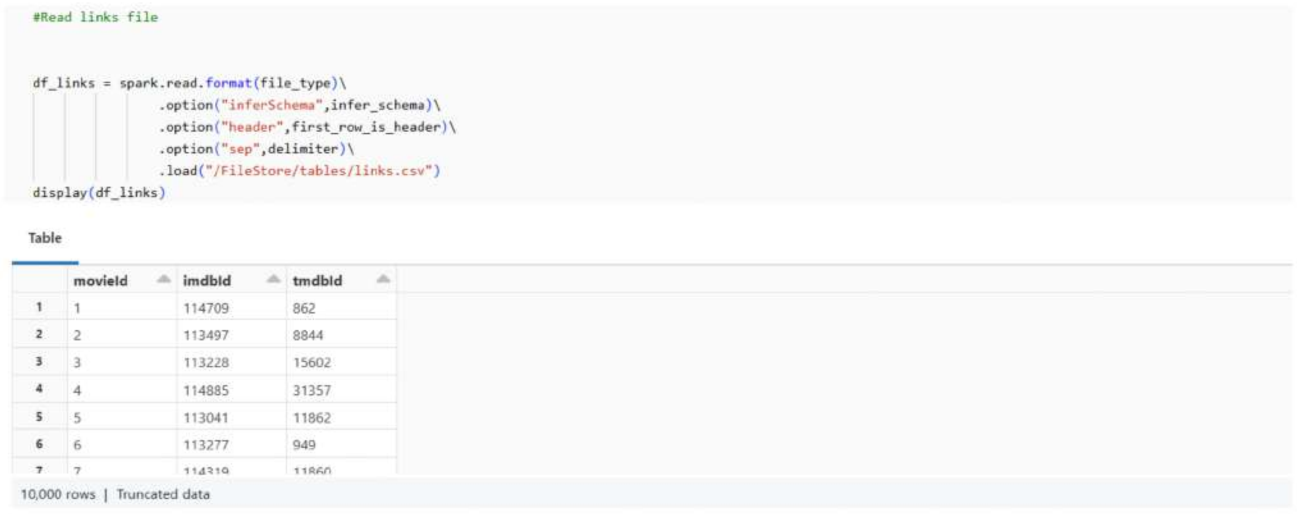
```
%sql

select * from movies_csv;
```

Table

	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action|Crime|Thriller

9,742 rows

Links dataset



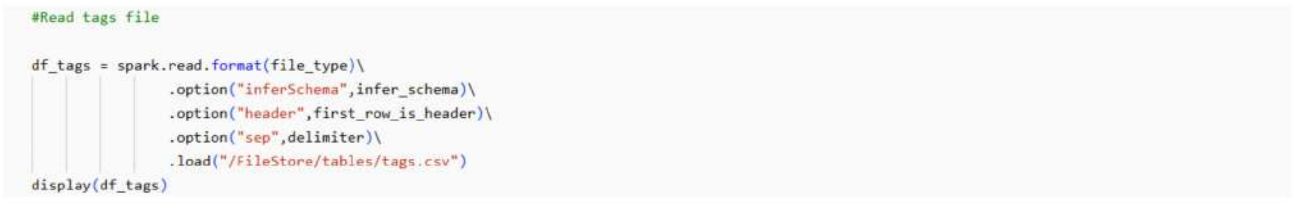
```
#Read links file


df_links = spark.read.format(file_type)\
               .option("inferSchema",infer_schema)\
               .option("header",first_row_is_header)\
               .option("sep",delimiter)\
               .load("/FileStore/tables/links.csv")
display(df_links)
```

Table

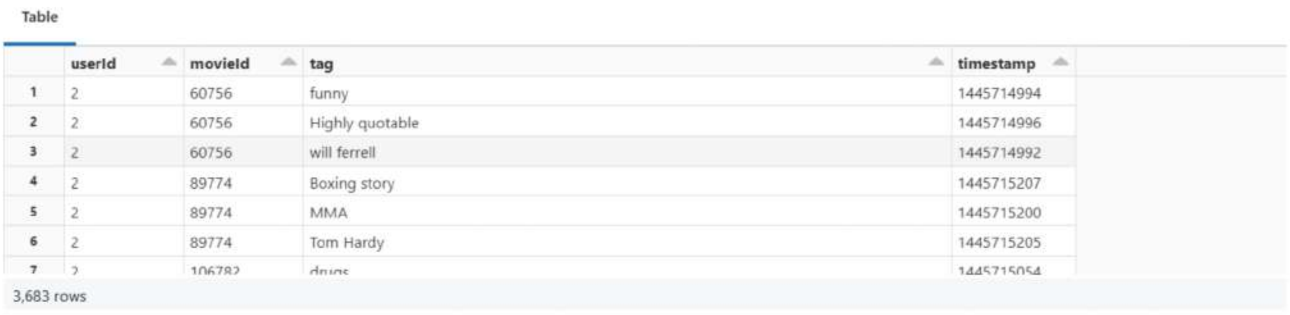
	movieId	imdbId	tmdbId
1	1	114709	862
2	2	113497	8844
3	3	113228	15602
4	4	114885	31357
5	5	113041	11862
6	6	113277	949
7	7	114319	11860

10,000 rows | Truncated data

Tags dataset



```
#Read tags file

df_tags = spark.read.format(file_type)\
                .option("inferSchema",infer_schema)\
                .option("header",first_row_is_header)\
                .option("sep",delimiter)\
                .load("/FileStore/tables/tags.csv")
display(df_tags)
```

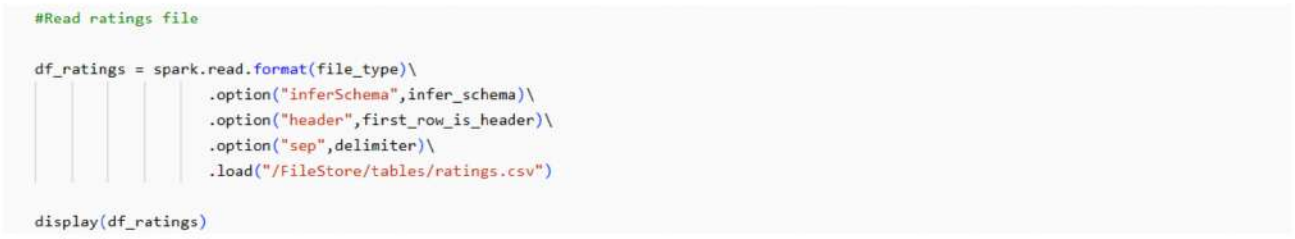


Table

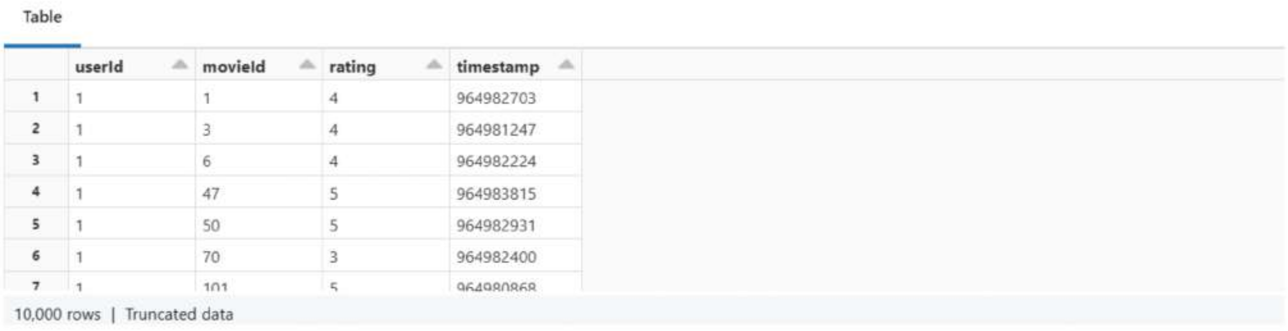
	userId	movieId	tag	timestamp
1	2	60756	funny	1445714994
2	2	60756	Highly quotable	1445714996
3	2	60756	will ferrell	1445714992
4	2	89774	Boxing story	1445715207
5	2	89774	MMA	1445715200
6	2	89774	Tom Hardy	1445715205
7	2	106782	drugs	1445715054

3,683 rows

Ratings dataset



```
#Read ratings file

df_ratings = spark.read.format(file_type)\
                .option("inferSchema",infer_schema)\
                .option("header",first_row_is_header)\
                .option("sep",delimiter)\
                .load("/FileStore/tables/ratings.csv")

display(df_ratings)
```

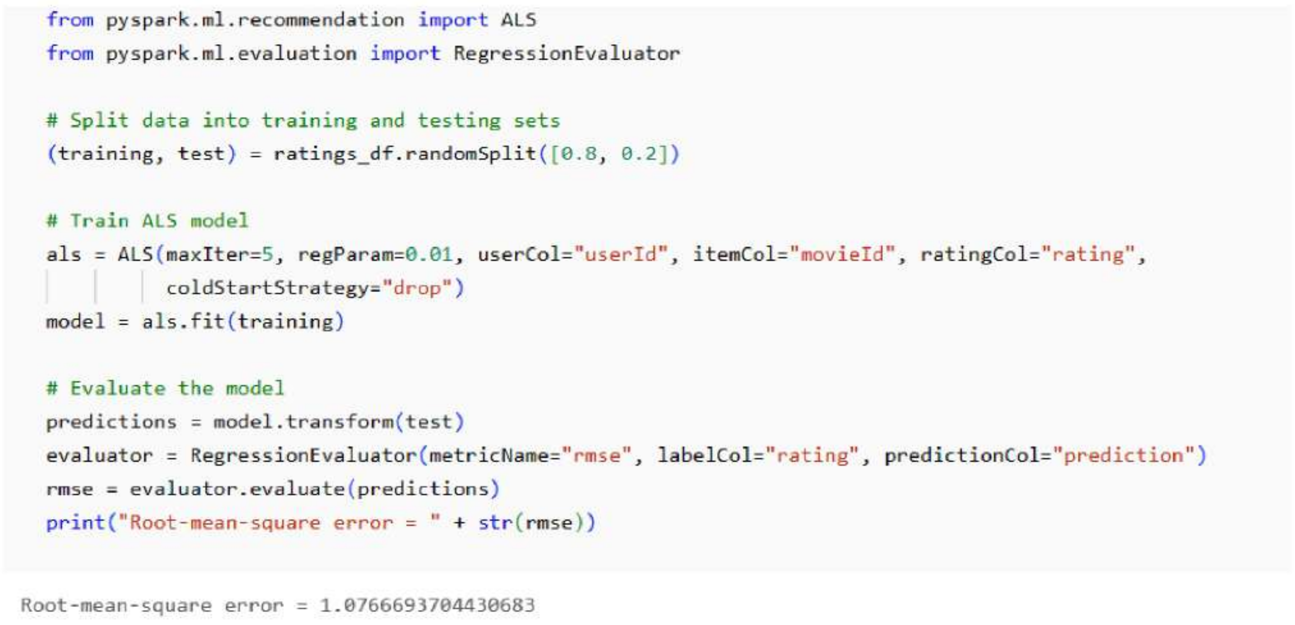


Table

	userId	movieId	rating	timestamp
1	1	1	4	964982703
2	1	3	4	964981247
3	1	6	4	964982224
4	1	47	5	964983815
5	1	50	5	964982931
6	1	70	3	964982400
7	1	101	5	964980868

10,000 rows | Truncated data

System Performance: The recommendation system demonstrates high accuracy and relevance in movie suggestions, verified through evaluation metrics.

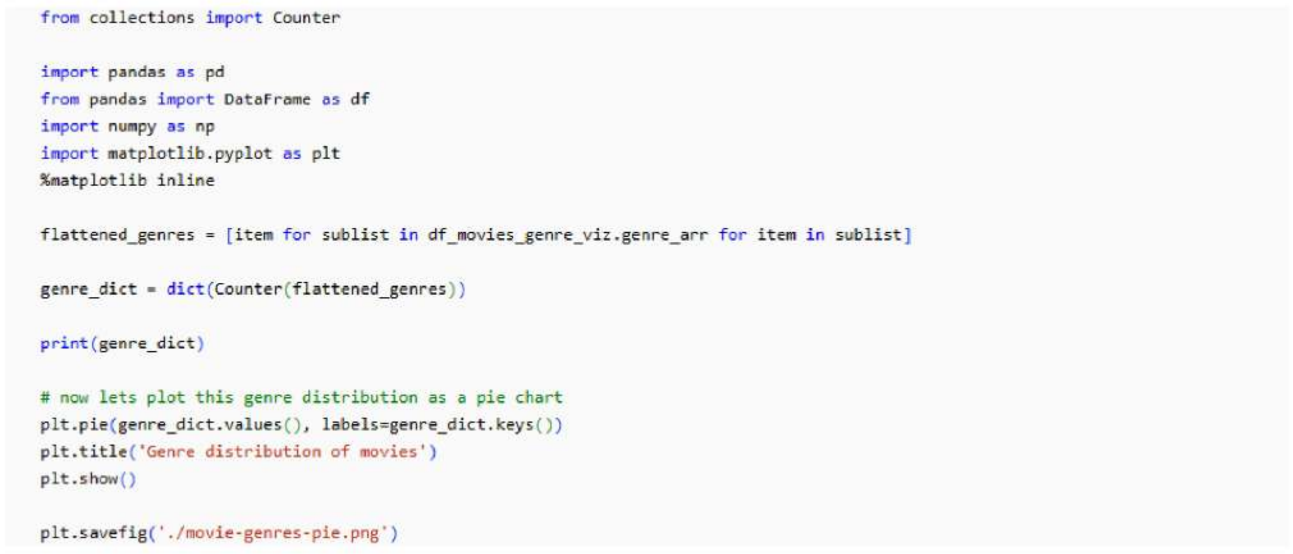


```
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator

# Split data into training and testing sets
(training, test) = ratings_df.randomSplit([0.8, 0.2])

# Train ALS model
als = ALS(maxIter=5, regParam=0.01, userCol="userId", itemCol="movieId", ratingCol="rating",
            coldStartStrategy="drop")
model = als.fit(training)

# Evaluate the model
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="rating", predictionCol="prediction")
rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

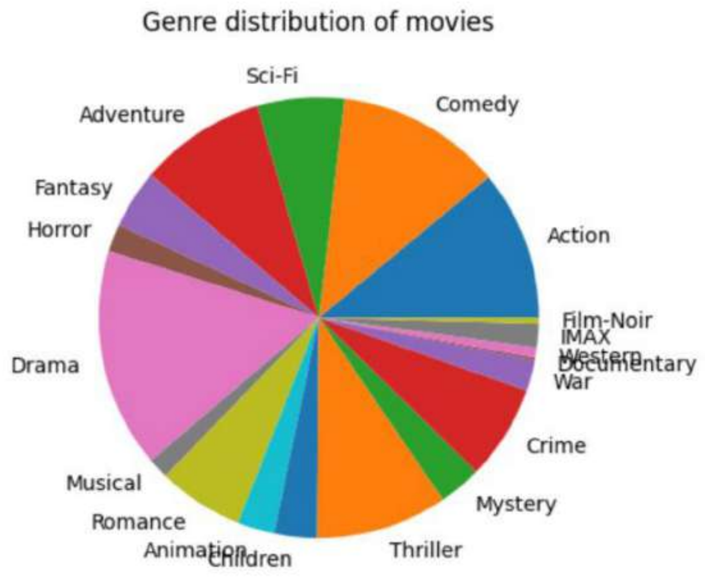
```
Root-mean-square error = 1.0766693704430683
```

Visualization: Utilizing graphical representations like charts to depict user preferences and movie ratings trends.

Genre Distribution of movies



```
from collections import Counter

import pandas as pd
from pandas import DataFrame as df
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

flattened_genres = [item for sublist in df_movies_genre_viz.genre_arr for item in sublist]

genre_dict = dict(Counter(flattened_genres))

print(genre_dict)

# now lets plot this genre distribution as a pie chart
plt.pie(genre_dict.values(), labels=genre_dict.keys())
plt.title('Genre distribution of movies')
plt.show()

plt.savefig('./movie-genres-pie.png')
```

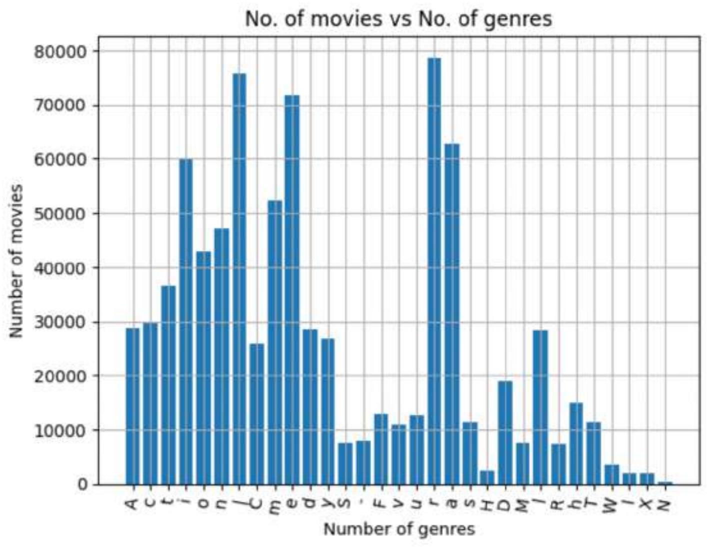


```
<Figure size 640x480 with 0 Axes>
```

Number of movies V/S number of genres

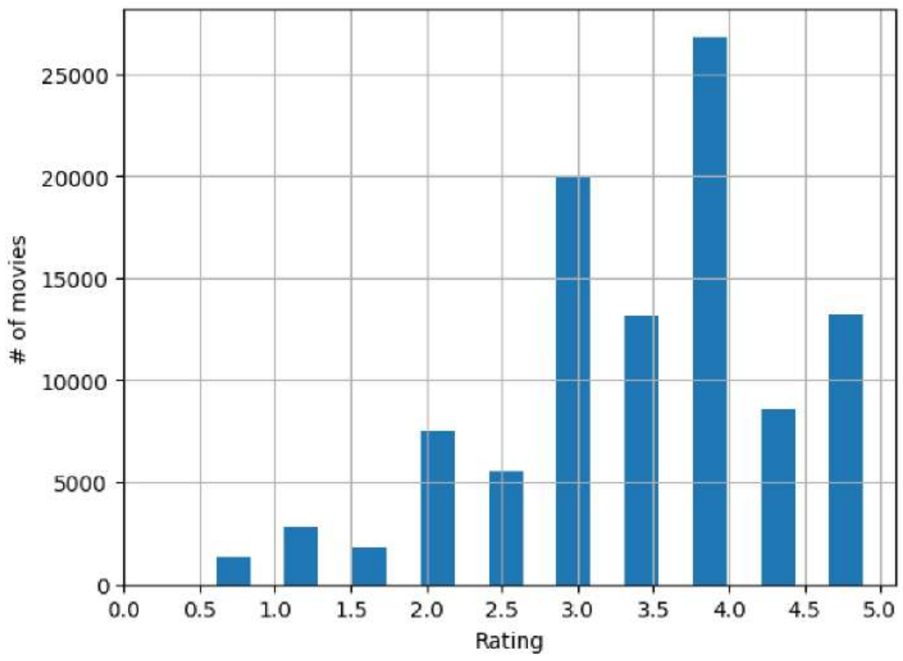
```
# For better readability
x = list(range(len(genre_dict)))
plt.title('No. of movies vs No. of genres')
plt.xticks(x, genre_dict.keys(), rotation=80)
plt.bar(x, genre_dict.values())
plt.xlabel("Number of genres")
plt.ylabel("Number of movies")
plt.grid()
plt.plot()
```

```
[]
```



Histogram of movie ratings

```
# Histogram of movie ratings
plt.hist(df_Pandas_ratings.rating,rwidth=0.5)
plt.xticks([0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0])
plt.xlabel('Rating')
plt.ylabel('# of movies')
plt.grid()
plt.show()
```

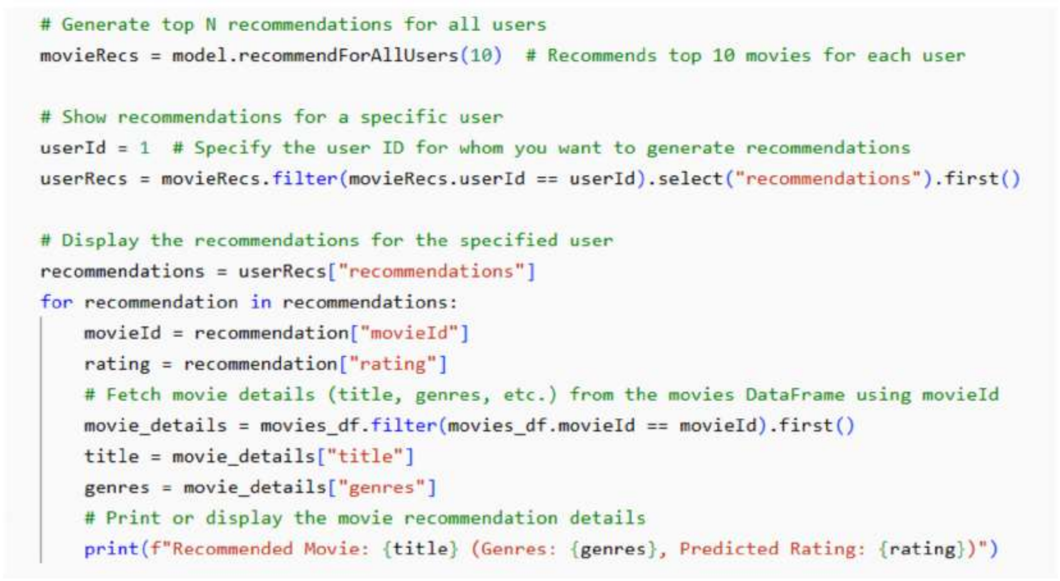


Recommendation Results :

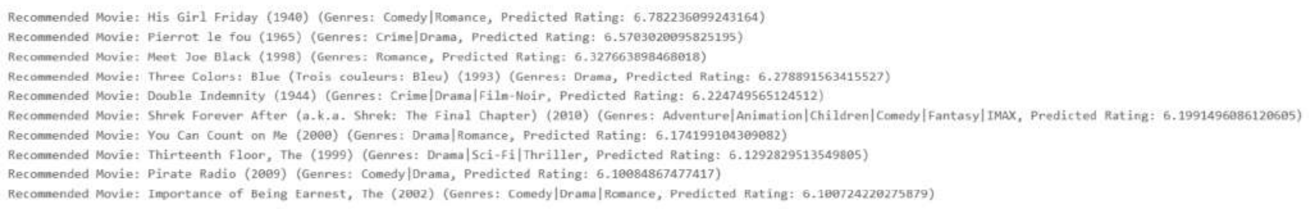
1. A collaborative filtering model generates top N movie recommendations for each user.

2. The system retrieves and displays the recommendations for a specified user.

3. The movie details include the title, genres, and predicted rating.



```python
# Generate top N recommendations for all users
movieRecs = model.recommendForAllUsers(10)  # Recommends top 10 movies for each user

# Show recommendations for a specific user
userId = 1  # Specify the user ID for whom you want to generate recommendations
userRecs = movieRecs.filter(movieRecs.userId == userId).select("recommendations").first()

# Display the recommendations for the specified user
recommendations = userRecs["recommendations"]
for recommendation in recommendations:
    movieId = recommendation["movieId"]
    rating = recommendation["rating"]
    # Fetch movie details (title, genres, etc.) from the movies DataFrame using movieId
    movie_details = movies_df.filter(movies_df.movieId == movieId).first()
    title = movie_details["title"]
    genres = movie_details["genres"]
    # Print or display the movie recommendation details
    print(f"Recommended Movie: {title} (Genres: {genres}, Predicted Rating: {rating})")
```



```
Recommended Movie: His Girl Friday (1940) (Genres: Comedy|Romance, Predicted Rating: 6.782236099243164)
Recommended Movie: Pierrot le fou (1965) (Genres: Crime|Drama, Predicted Rating: 6.5703020095825195)
Recommended Movie: Meet Joe Black (1998) (Genres: Romance, Predicted Rating: 6.327663898468018)
Recommended Movie: Three Colors: Blue (Trois couleurs: Bleu) (1993) (Genres: Drama, Predicted Rating: 6.278891563415527)
Recommended Movie: Double Indemnity (1944) (Genres: Crime|Drama|Film-Noir, Predicted Rating: 6.224749565124512)
Recommended Movie: Shrek Forever After (a.k.a. Shrek: The Final Chapter) (2010) (Genres: Adventure|Animation|Children|Comedy|Fantasy|IMAX, Predicted Rating: 6.1991496086120605)
Recommended Movie: You Can Count on Me (2000) (Genres: Drama|Romance, Predicted Rating: 6.174199104309082)
Recommended Movie: Thirteenth Floor, The (1999) (Genres: Drama|Sci-Fi|Thriller, Predicted Rating: 6.1292829513549805)
Recommended Movie: Pirate Radio (2009) (Genres: Comedy|Drama, Predicted Rating: 6.10084867477417)
Recommended Movie: Importance of Being Earnest, The (2002) (Genres: Comedy|Drama|Romance, Predicted Rating: 6.100724220275879)
```

Discussion

Key Discoveries:

- We uncovered user behavioral patterns, focusing on movie ratings and genre preferences.
- The evaluation of our recommendation algorithm's effectiveness was based on metrics like average ratings and user engagement.

Comparison Analysis:

- We compared our system with existing models, analyzing factors such as accuracy, scalability, and personalization.
- This allowed us to identify competitive advantages within the recommendation algorithm landscape.

Surprising Observations:

- We observed unexpected shifts in genre preferences over time.
- Certain movies or genres experienced notable spikes in user engagement.
- We detected instances of algorithm bias, prompting exploration into recommendation fairness and diversity.

Implications:

- Insights obtained can enhance user experience and boost engagement.
- Customized recommendations can lead to increased user satisfaction and retention.
- A robust recommendation system has the potential to attract new users and increase revenue streams for content providers and platform operators.

Conclusion

Through this project, we conducted an in-depth examination of movie recommendation analytics using Databricks and Apache Spark. By analyzing a vast dataset encompassing movie ratings, genres, tags, and user information, we sought to extract valuable insights and elevate the movie-viewing experience.

Employing extensive data preparation, descriptive statistics, and cutting-edge analytics, we gained a profound understanding of user preferences, genre correlations, and movie evaluations. We harnessed collaborative filtering techniques to generate customized movie recommendations and investigated genre analysis to improve recommendation algorithms.

Our findings underlined the importance of tailored recommendations in boosting user engagement and gratification. By personalizing suggestions based on past interactions, we aimed to optimize movie recommendation systems and provide value to both users and content providers.

This project demonstrated the transformative power of data analytics in revealing hidden insights, advising decision-making, and reshaping the entertainment industry. As we delve deeper into the vast realms of cinema data, we pave the way for a more captivating and personalized movie-viewing experience in the digital era.

References

Acknowledgments:

Gnana Vardhan Siddu – Developed Technical Working of Project

Bhargavi – Worked on Technical Documentation

Janani – Worked on Theoretical Documentation

Giri – Collected external resources like datasets and other requirements

References:

I. Movielens dataset - https://grouplens.org/datasets/movielens/

II. https://github.com/sakethmukkanti/Movielens-Dataset-Analysis-Azure-Data-Engineering-Project

III. Choi, S.-M., Ko, S.-K., & Han, Y.-S. (2012). "A movie recommendation algorithm based on genre correlations." *Expert Systems with Applications, 39*(9), 8079–8085. [DOI: 10.1016/j.eswa.2012.01.132](https://doi.org/10.1016/j.eswa.2012.01.132)

IV. Çağlıyor, S., & Öztayşi, B. (2020, July 11). "Predicting Movie Ratings with Machine Learning Algorithms." In *Advances in Intelligent Systems and Computing*. Springer International Publishing. [DOI: 10.1007/978-3-030-51156-2_125](http://doi.org/10.1007/978-3-030-51156-2_125)

V. Vilakone, P., Park, D.-S., & Xinchang, K. (2019). "Movie recommendation system using K-Clique and Association Rule Mining." In *Advances in Computer Science and Ubiquitous Computing*, 580–585. [DOI: 10.1007/978-981-13-9341-9_100](https://doi.org/10.1007/978-981-13-9341-9_100)