Movie Recommendation with MLlib - Collaborative Filtering

CS570 Big Data Processing Project
By Feven Araya
Instructor: Dr. Chang, Henry

Table of contents

1. Introduction
2. Design
3. Implementation
4. Testing
5. Enhancement
6. Conclusion
7. References

01

Introduction

Project Context

- Introduction to the use of machine learning in the entertainment industry, specifically in recommending movies to users based on their preferences.
- Brief overview of the significance of personalized recommendations in enhancing user experience and engagement on movie platforms.

Objective of the Project

- To develop a machine learning model that accurately predicts user preferences and recommends movies using Collaborative Filtering techniques.
- Aim to leverage Apache Spark's MLlib for efficient processing of large-scale movie rating data from the MovieLens dataset.

Goals

- Implement and evaluate a Collaborative Filtering model using the Alternating Least Squares (ALS) algorithm within MLlib.

02

Design

Description of the MovieLens Dataset

- **Origin and Composition**: The MovieLens dataset is a widely used dataset in recommender systems research, compiled by the GroupLens research group at the University of Minnesota. It consists of millions of movie ratings provided by users, along with user demographic information and movie metadata.
- **Data Structure**: Explain the structure of the dataset, typically comprising user IDs, movie IDs, ratings, and timestamps. Highlight any specific subset or version of the dataset used, such as MovieLens 100k or 1M.
- **Importance of Data**: Discuss the diversity and volume of the data, which makes it ideal for training robust machine learning models for recommendation systems.

Relevance of the MovieLens Dataset

- **Benchmarking Standard**: The dataset serves as a benchmark for evaluating the effectiveness of various recommendation algorithms, allowing comparisons across different methods and studies.
- **Real-World Application**: Using a real-world dataset like MovieLens helps in understanding user behavior patterns and preferences, which is crucial for developing systems that can be deployed in commercial settings.

Design Approach Using Collaborative Filtering in MLlib

- **Algorithm Choice**: Use of Collaborative Filtering, specifically the Alternating Least Squares (ALS) algorithm,
- **Model Framework**: Discuss the integration of the ALS algorithm within Apache Spark's MLlib, **Scalability and Performance**: Highlight the scalability of the Spark platform and how it facilitates faster computations and model training over a distributed computing environment, which is essential for real-time recommendation systems.

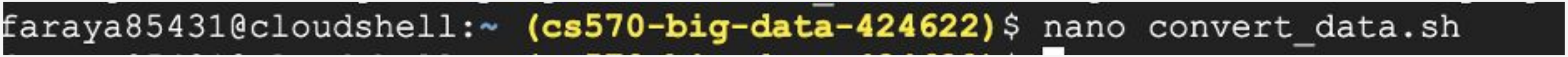
03

Implementation

1. Project: Movie Recommendation with MLlib - Collaborative Filtering (implementaiton

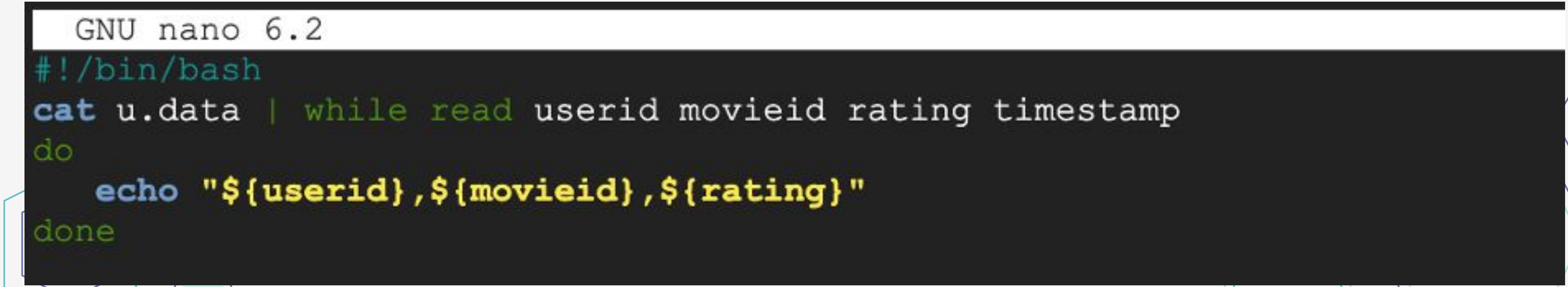
Create the Shell Script

1. Create a new shell script file using a text editor, such as nano or vim.



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ nano convert_data.sh
```

2. Copy and paste one of the following scripts into the text editor.

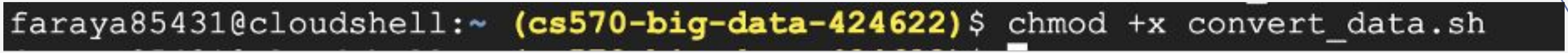


```
  GNU nano 6.2
#!/bin/bash
cat u.data | while read userid movieid rating timestamp
do
   echo "${userid},${movieid},${rating}"
done
```

3. Save the file and exit the text editor (Ctrl+X, then Y, then Enter for nano).

Make the Script Executable

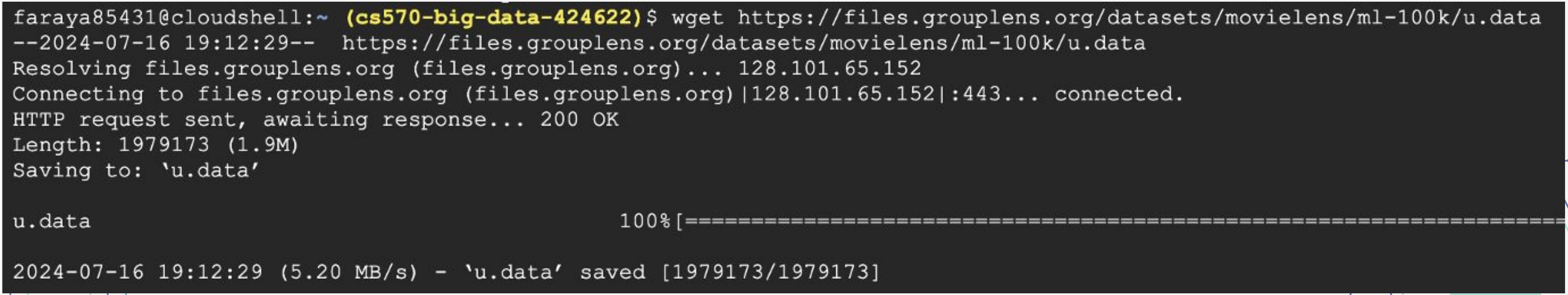
Run the following command to make the script executable



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ chmod +x convert_data.sh
```

Prepare the Input Data

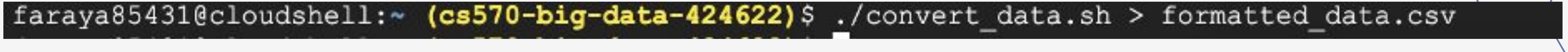
Upload u.data file using the Cloud Shell file upload feature or download it using wget or curl.



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ wget https://files.grouplens.org/datasets/movielens/ml-100k/u.data
--2024-07-16 19:12:29--  https://files.grouplens.org/datasets/movielens/ml-100k/u.data
Resolving files.grouplens.org (files.grouplens.org)... 128.101.65.152
Connecting to files.grouplens.org (files.grouplens.org)|128.101.65.152|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1979173 (1.9M)
Saving to: 'u.data'

u.data                                        100%[=====================================================================

2024-07-16 19:12:29 (5.20 MB/s) - 'u.data' saved [1979173/1979173]
```

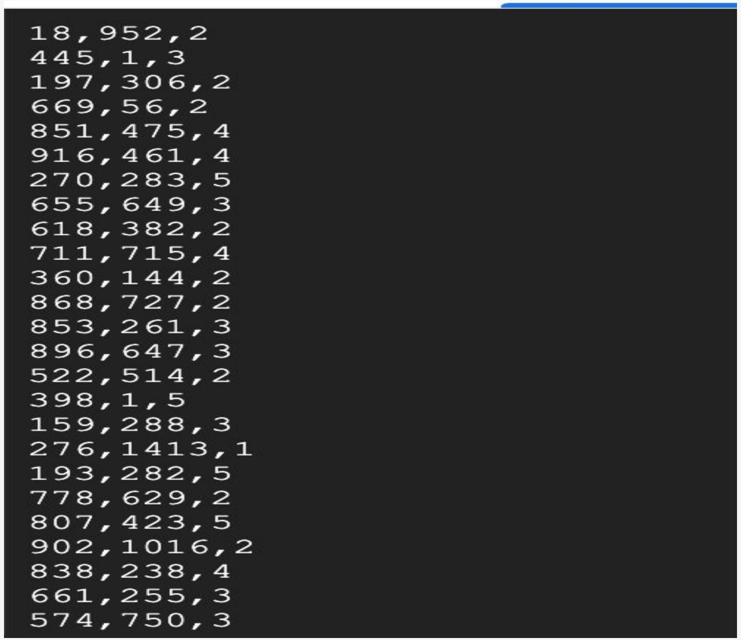
Run the Script

1. Run the script and save the output to a file:



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ ./convert_data.sh > formatted_data.csv
```

2. Cat formatted_data.csv

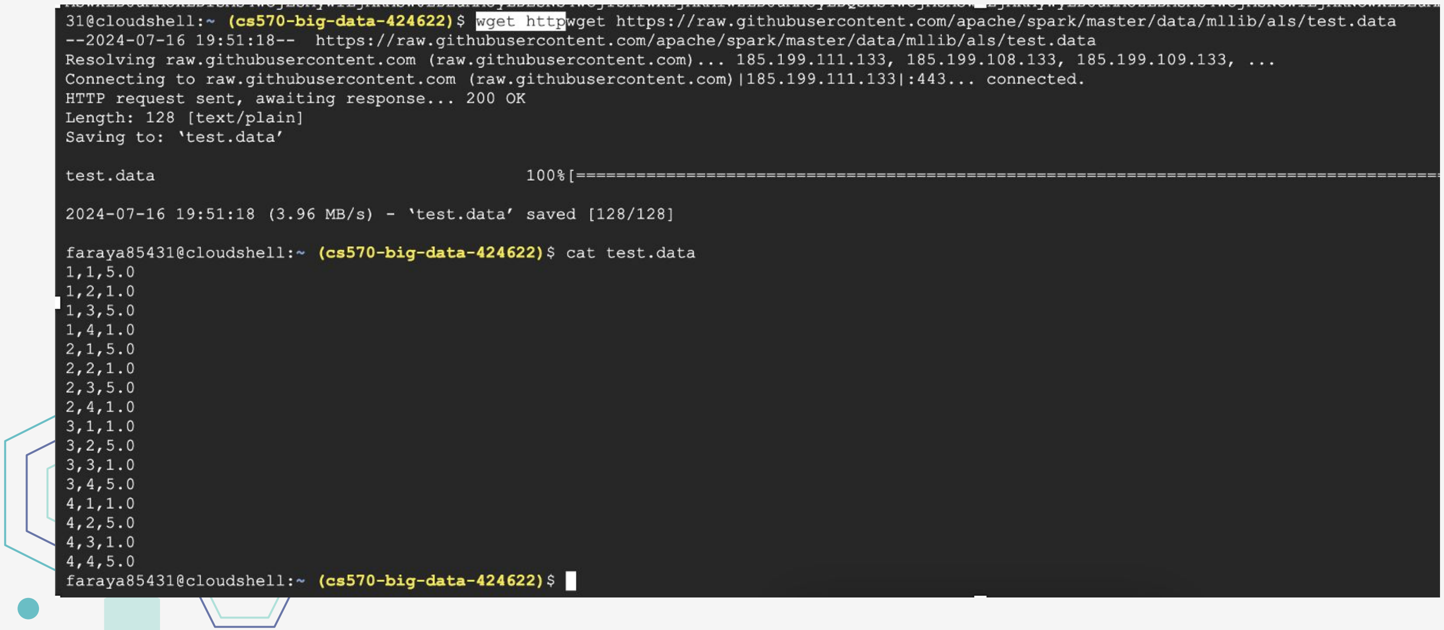


```
18,952,2
445,1,3
197,306,2
669,56,2
851,475,4
916,461,4
270,283,5
655,649,3
618,382,2
711,715,4
360,144,2
868,727,2
853,261,3
896,647,3
522,514,2
398,1,5
159,288,3
276,1413,1
193,282,5
778,629,2
807,423,5
902,1016,2
838,238,4
661,255,3
574,750,3
```

2. Implement this version of MLlib - Collaborative Filtering Examples Python Latest code: recommendation_example.py (complete code)

Download the test.data File

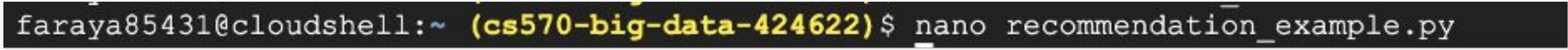
1. Open Cloud Shell.
2. Use the wget command to download the filed



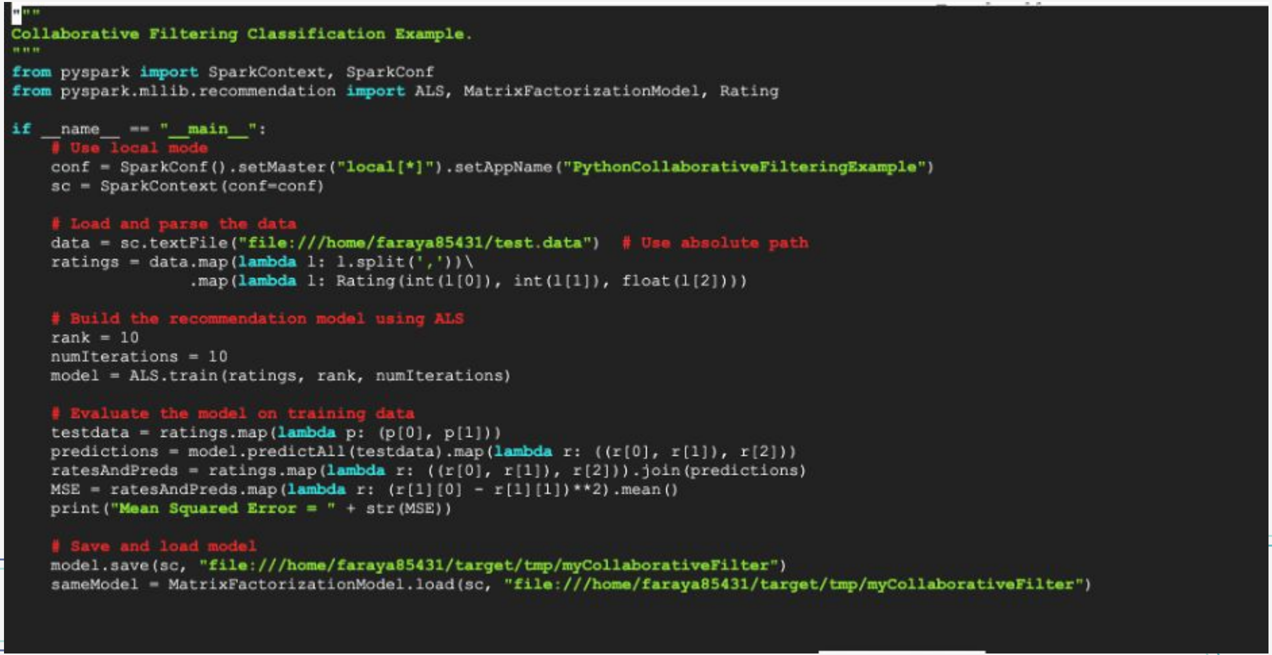
```
31@cloudshell:~ (cs570-big-data-424622)$ wget httpwget https://raw.githubusercontent.com/apache/spark/master/data/mllib/als/test.data
--2024-07-16 19:51:18--  https://raw.githubusercontent.com/apache/spark/master/data/mllib/als/test.data
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 128 [text/plain]
Saving to: ‘test.data’

test.data                                  100%[==========================================================================

2024-07-16 19:51:18 (3.96 MB/s) - ‘test.data’ saved [128/128]

faraya85431@cloudshell:~ (cs570-big-data-424622)$ cat test.data
1,1,5.0
1,2,1.0
1,3,5.0
1,4,1.0
2,1,5.0
2,2,1.0
2,3,5.0
2,4,1.0
3,1,1.0
3,2,5.0
3,3,1.0
3,4,5.0
4,1,1.0
4,2,5.0
4,3,1.0
4,4,5.0
faraya85431@cloudshell:~ (cs570-big-data-424622)$ 
```

Write the recommendation_example.py Script

1. Create and edit the Python script



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ nano recommendation_example.py
```



```
"""
Collaborative Filtering Classification Example.
"""
from pyspark import SparkContext, SparkConf
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating

if __name__ == "__main__":
    # Use local mode
    conf = SparkConf().setMaster("local[*]").setAppName("PythonCollaborativeFilteringExample")
    sc = SparkContext(conf=conf)

    # Load and parse the data
    data = sc.textFile("file:///home/faraya85431/test.data")  # Use absolute path
    ratings = data.map(lambda l: l.split(','))\
              .map(lambda l: Rating(int(l[0]), int(l[1]), float(l[2])))

    # Build the recommendation model using ALS
    rank = 10
    numIterations = 10
    model = ALS.train(ratings, rank, numIterations)

    # Evaluate the model on training data
    testdata = ratings.map(lambda p: (p[0], p[1]))
    predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
    ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
    MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
    print("Mean Squared Error = " + str(MSE))

    # Save and load model
    model.save(sc, "file:///home/faraya85431/target/tmp/myCollaborativeFilter")
    sameModel = MatrixFactorizationModel.load(sc, "file:///home/faraya85431/target/tmp/myCollaborativeFilter")
```

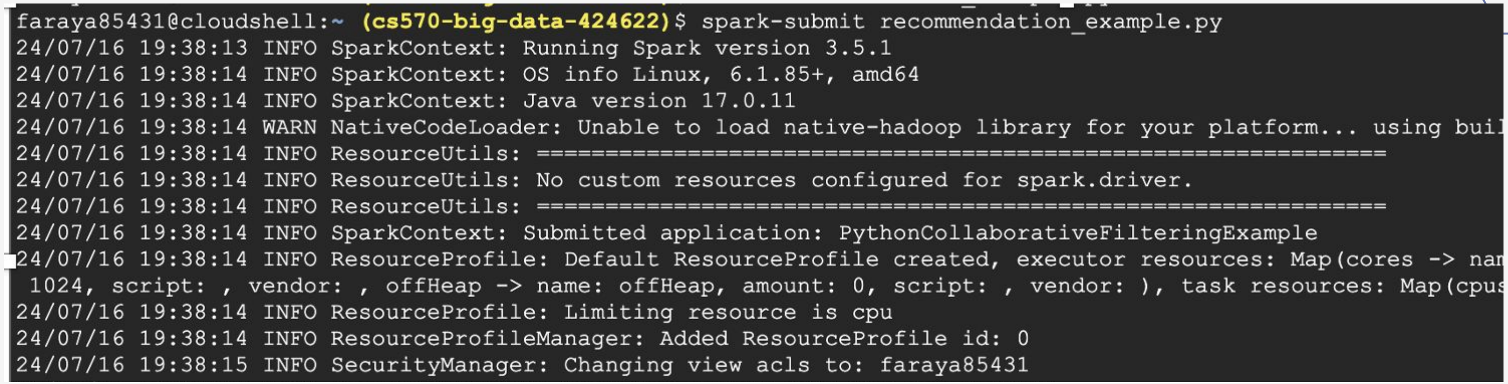
04

Test

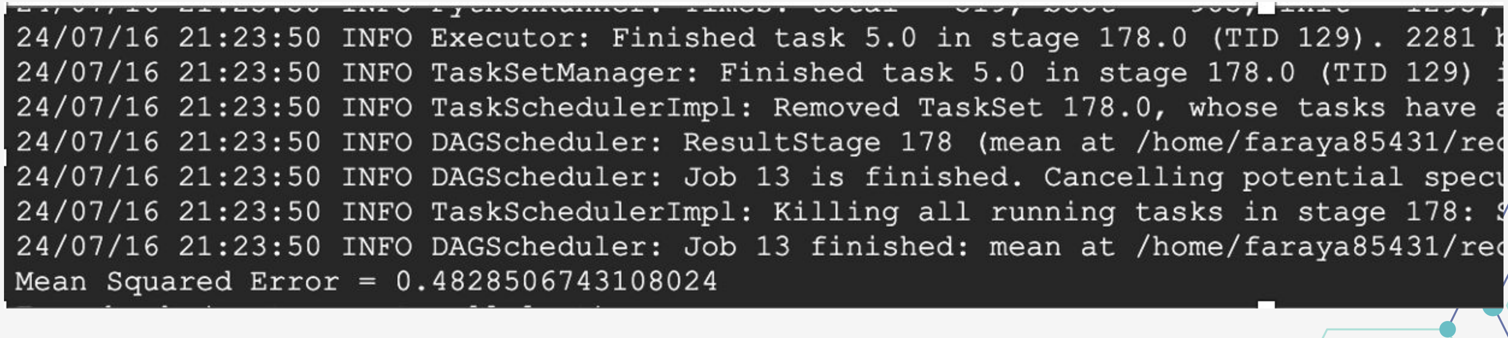
Process to test the project

Run the Script

Execute the script using `spark-submit`:



```
faraya85431@cloudshell:~ (cs570-big-data-424622)$ spark-submit recommendation_example.py
24/07/16 19:38:13 INFO SparkContext: Running Spark version 3.5.1
24/07/16 19:38:14 INFO SparkContext: OS info Linux, 6.1.85+, amd64
24/07/16 19:38:14 INFO SparkContext: Java version 17.0.11
24/07/16 19:38:14 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using buil
24/07/16 19:38:14 INFO ResourceUtils: ==============================================================
24/07/16 19:38:14 INFO ResourceUtils: No custom resources configured for spark.driver.
24/07/16 19:38:14 INFO ResourceUtils: ==============================================================
24/07/16 19:38:14 INFO SparkContext: Submitted application: PythonCollaborativeFilteringExample
24/07/16 19:38:14 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> nam
 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus
24/07/16 19:38:14 INFO ResourceProfile: Limiting resource is cpu
24/07/16 19:38:14 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/07/16 19:38:15 INFO SecurityManager: Changing view acls to: faraya85431
```

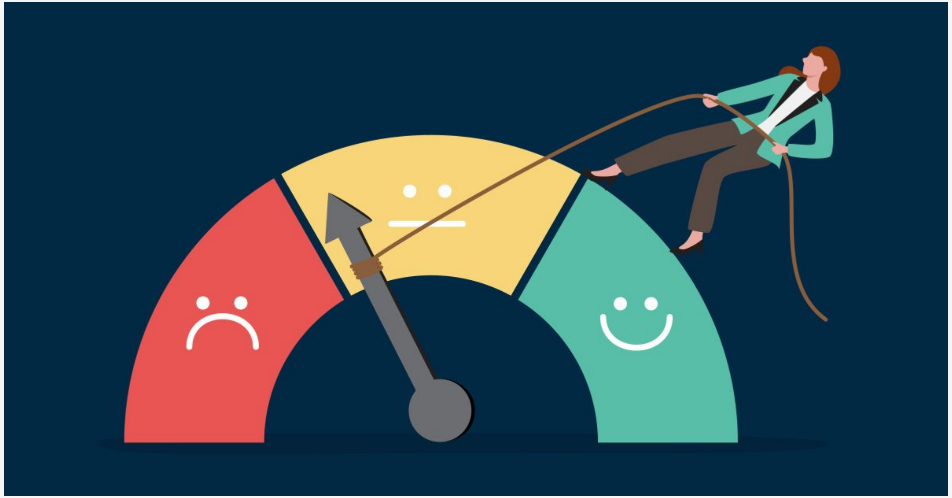


```
24/07/16 21:23:50 INFO Executor: Finished task 5.0 in stage 178.0 (TID 129). 2281 b
24/07/16 21:23:50 INFO TaskSetManager: Finished task 5.0 in stage 178.0 (TID 129) i
24/07/16 21:23:50 INFO TaskSchedulerImpl: Removed TaskSet 178.0, whose tasks have a
24/07/16 21:23:50 INFO DAGScheduler: ResultStage 178 (mean at /home/faraya85431/rec
24/07/16 21:23:50 INFO DAGScheduler: Job 13 is finished. Cancelling potential specu
24/07/16 21:23:50 INFO TaskSchedulerImpl: Killing all running tasks in stage 178: S
24/07/16 21:23:50 INFO DAGScheduler: Job 13 finished: mean at /home/faraya85431/rec
Mean Squared Error = 0.4828506743108024
```

The Mean Squared Error is 0.482850…

05

Enhancements



Can we get better result?

Incorporate Additional Data Sources

- **Content-Based Features**: Include movie metadata like genres, director, cast, and plot descriptions to enrich the recommendation engine. This can help in cases where user interaction data is sparse.
- **User Context**: Factor in contextual information such as time of day, device used, or user location to provide more personalized recommendations.

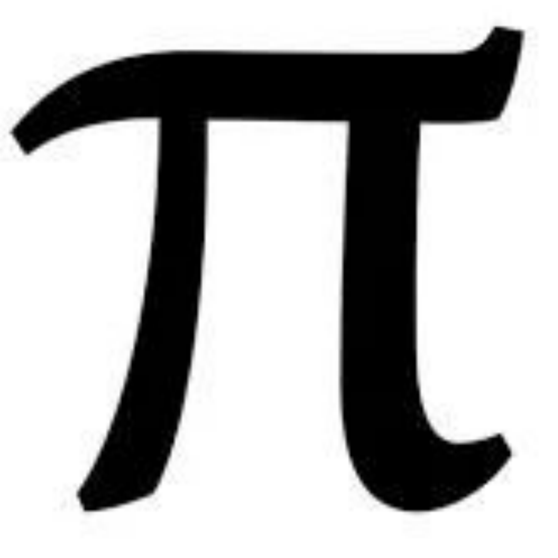
06

Conclusion

- Integrating more complex models, incorporating real-time data, or improving the user interface.
- Express the potential impact of these improvements on user experience and system performance.

07

References



Movie Recommendation with Collaborative Filtering in Pyspark

How to Build a Movie Recommendation System Based on Collaborative Filtering

Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution