

BIG DATA PROJECT

IMDB DATSET

GAURAV PUJARI – PES2201800149

ABHISHEK YOGAN – PES2201800164

ABSTRACT- Movie recommendation system predict what kind of movies a user will like based on the attributes present in previous movies and its beneficial for organizations that collect large amounts of data from the customers and improving their marketing strategies and provide the best suggestions possible. Our project aims at recommending top 3 movies for every user who has rated previous movies and enhance the user satisfaction. We have used ALS model for our project.

First we import the necessary libraries required:

```
from pyspark.sql import SparkSession
from pyspark.sql import Row
from pyspark.sql import functions as func
import os
import sys
import pandas as pd
from pyspark.ml.recommendation import ALS
from pyspark.sql.types import *
import re
from pyspark.sql.functions import *
from pyspark.ml.evaluation import RegressionEvaluator
```

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="darkgrid")
plt.style.use("seaborn-pastel")
```

Only 2 files were necessary out of all the files from the dataset which are title.basics and title.ratings for analysis and recommendation system. The files were taken from the imdb dataset in Kaggle.

creating a spark session.

```
spark = SparkSession.builder.appName("SparkSQL").getOrCreate()
```

we will be loading our files in hdfs and reading it from hdfs to perform the required analysis and visualization.

```
read_file1=spark.read.option("header","true").option("inferSchema",
"true").option("sep","\t").csv("hdfs://127.0.0.1:9000/input1/title.basics.tsv")
```

```
read_file2=spark.read.option("header","true").option("inferSchema",
"true").option("sep","\t").csv("hdfs://127.0.0.1:9000/input1/title.ratings.tsv")
```

#once the dataset is loaded pre-processing is done and we will be removing (dropping)the columns which are not required. And since our movie id which is tconst not numeric, we will convert it to numeric.

```
drop_cols1=['isAdult','startYear','endYear','runtimeMinutes']
```

```
df1 = read_file1.drop(*drop_cols1)
```

```
#read_file1=read_file1.select([column for column in read_file1.columns if
column not in drop_cols1]).columns
```

```
#df1.show()
```

```
#df1.tconst=pd.Categorical(df1.tconst)
```

```
#df1["tconst_new"]=df1.tconst.cat.codes
```

```
df2 = df1.withColumn('tconst', regexp_replace('tconst', '^tt', '0'))
```

```
df3 = read_file2.withColumn('tconst', regexp_replace('tconst', '^tt', '0'))
```

#typecasting

```
df2 = df2.withColumn("tconst", df2["tconst"].cast('int'))
```

```
df3 = df3.withColumn("tconst", df3["tconst"].cast('int'))
```

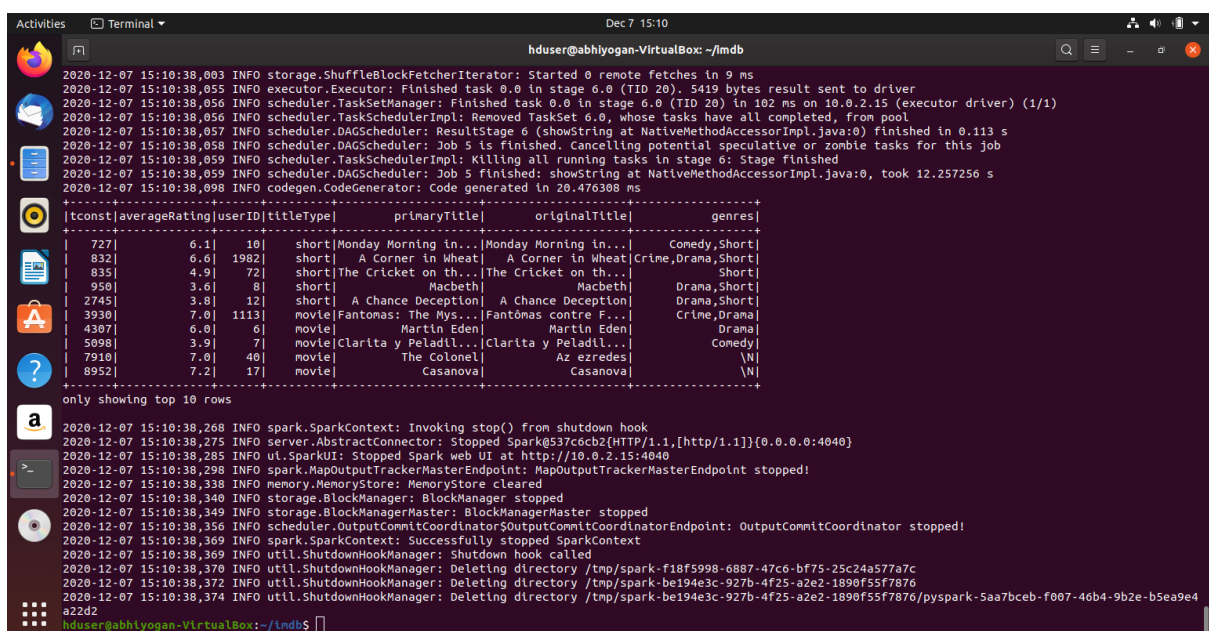
```
#df3 = df3.withColumn("tconst",round(rand()*(1000-5)+5,0))
```

#joining the dataframes.

```
joined_table=df3.join(df2, ['tconst']).dropDuplicates()
```

```
joined_table.show(10)
```

```
joined_table = joined_table.withColumn("tconst",round(rand()*(1000-5)+5,0))
```



The screenshot shows a terminal window with Spark logs and a table of movie data. The logs indicate the completion of various tasks and the shutdown of the Spark context. The table displays the top 10 rows of data, including columns for tconst, averageRating, user ID, title type, primary title, original title, and genres.

tconst	averageRating	user ID	titleType	primaryTitle	originalTitle	genres
727	6.1	10	short	Monday Morning in...	Monday Morning in...	Comedy,Short
832	6.6	1982	short	A Corner in Wheat	A Corner in Wheat	Crime,Drama,Short
835	4.9	72	short	The Cricket on th...	The Cricket on th...	Short
950	3.6	8	short	Macbeth	Macbeth	Drama,Short
2745	3.8	12	short	A Chance Deception	A Chance Deception	Drama,Short
3930	7.0	1113	movie	Fantomas: The Mys...	Fantômas contre F...	Crime,Drama
4307	6.0	6	movie	Martin Eden	Martin Eden	Drama
5098	3.9	7	movie	Clarita y Peladill...	Clarita y Peladill...	Comedy
7910	7.0	40	movie	The Colonel	Az ezredes	\N
8952	7.2	17	movie	Casanova	Casanova	\N

Queries:

Finding the most rated movie in our data set

```
def mostRatedMovie():
```

```
    mostRated = joinedTable.groupby('tconst').count().orderBy(func.asc("count"))
```

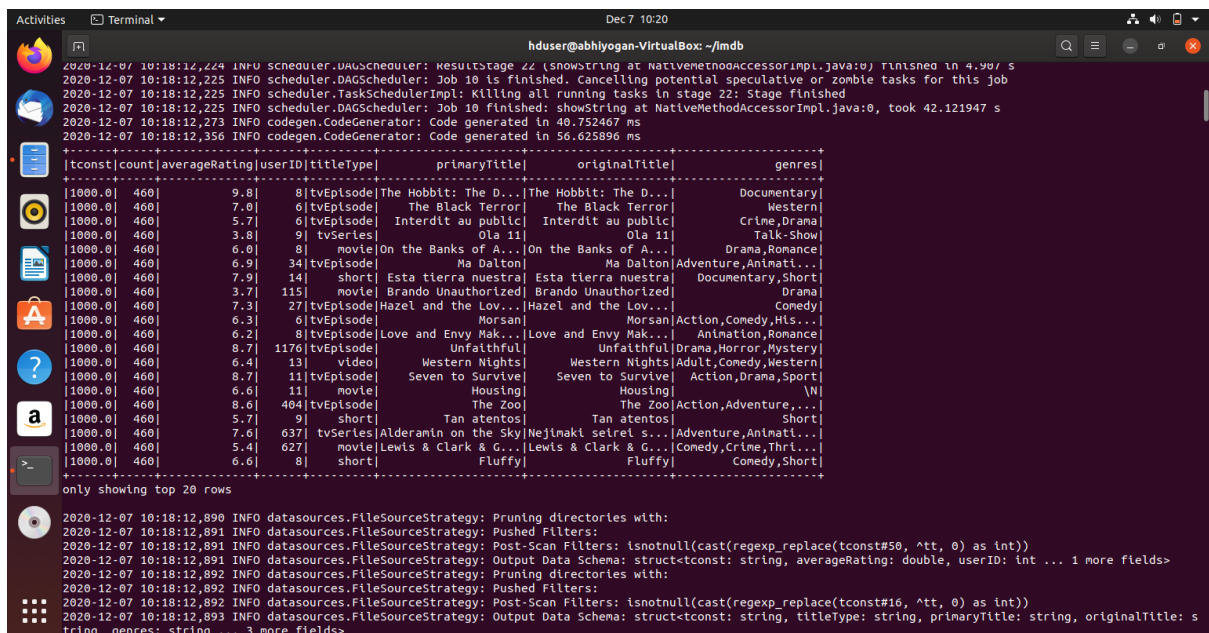
```
    mostRated = mostRated.join(joinedTable, ['tconst'])
```

```
    mostRated = mostRated.orderBy("count", descending = False)
```

```
    return mostRated
```

```
#y = mostRatedMovie()
```

```
#y.show()
```



The screenshot shows a terminal window with a PySpark query result. The query is: `mostRatedMovie().show()`. The result is a table with columns: `tconst`, `count`, `averageRating`, `userID`, `titleType`, `primaryTitle`, `originalTitle`, and `genres`. The table shows the top 20 rows of results, with the first row being `|1000.0| 460| 9.8| 8| tvEpisode| The Hobbit: The D...| The Hobbit: The D...| Documentary|`. The terminal also shows some log messages from the Spark scheduler and code generator.

tconst	count	averageRating	userID	titleType	primaryTitle	originalTitle	genres
1000.0	460	9.8	8	tvEpisode	The Hobbit: The D...	The Hobbit: The D...	Documentary
1000.0	460	7.0	6	tvEpisode	The Black Terror	The Black Terror	Western
1000.0	460	5.7	6	tvEpisode	Interdit au public	Interdit au public	Crime, Drama
1000.0	460	3.8	9	tvSeries	Ola 11	Ola 11	Talk-Show
1000.0	460	6.0	8	movie	On the Banks of A...	On the Banks of A...	Drama, Romance
1000.0	460	6.9	34	tvEpisode	Ma Dalton	Ma Dalton	Adventure, Animati...
1000.0	460	7.9	14	short	Esta tierra nuestra	Esta tierra nuestra	Documentary, Short
1000.0	460	3.7	115	movie	Brando Unauthorized	Brando Unauthorized	Drama
1000.0	460	7.3	27	tvEpisode	Hazel and the Lov...	Hazel and the Lov...	Comedy
1000.0	460	6.3	6	tvEpisode	Morsan	Morsan	Action, Comedy, Hls...
1000.0	460	6.2	8	tvEpisode	Love and Envy Mak...	Love and Envy Mak...	Animation, Romance
1000.0	460	8.7	1176	tvEpisode	Unfaithful	Unfaithful	Drama, Horror, Mystery
1000.0	460	6.4	13	video	Western Nights	Western Nights	Adult, Comedy, Western
1000.0	460	8.7	11	tvEpisode	Seven to Survive	Seven to Survive	Action, Drama, Sport
1000.0	460	6.6	11	movie	Housing	Housing	\N
1000.0	460	8.6	404	tvEpisode	The Zoo	The Zoo	Action, Adventure, ...
1000.0	460	5.7	9	short	Tan atentos	Tan atentos	Short
1000.0	460	7.6	637	tvSeries	Alderanin on the Sky	Nejinaki setrel s...	Adventure, Animati...
1000.0	460	5.4	627	movie	Lewis & Clark & G...	Lewis & Clark & G...	Comedy, Crime, Thrl...
1000.0	460	6.6	8	short	Fluffy	Fluffy	Comedy, Short

This tells what movies users like the most.

To find the most like genre:

```
def mostLikedGenre():
```

```
    genre = joinedTable.groupby('genres').count().orderBy(func.desc("count"))
```

```
    genre = genre.join(joinedTable, ['genres'])
```

```
    genre = genre.orderBy("count", ascending = False)
```

```
    return genre
```

```

Activities Terminal ▾ Dec 7 10:48
hduser@abhlyogan-VirtualBox: ~/lmdb

2020-12-07 10:47:32,208 INFO executor.Executor: Finished task 198.0 in stage 33.0 (TID 1849). 12984 bytes result sent to driver
2020-12-07 10:47:32,210 INFO scheduler.TaskSetManager: Finished task 198.0 in stage 33.0 (TID 1849) in 103 ms on 10.0.2.15 (executor driver) (200/200)
2020-12-07 10:47:32,210 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 33.0, whose tasks have all completed, from pool
2020-12-07 10:47:32,211 INFO scheduler.DAGScheduler: ResultStage 33 (showString at NativeMethodAccessorImpl.java:0) finished in 6.678 s
2020-12-07 10:47:32,211 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 33: Stage finished
2020-12-07 10:47:32,214 INFO scheduler.DAGScheduler: Job 13 finished: showString at NativeMethodAccessorImpl.java:0, took 62.081069 s
2020-12-07 10:47:32,236 INFO codegen.CodeGenerator: Code generated in 16.897696 ms
2020-12-07 10:47:32,262 INFO codegen.CodeGenerator: Code generated in 14.689677 ms

+-----+
|genres|count|tconst|averageRating|userID|titleType|primaryTitle|originalTitle|
+-----+
|Comedy|89083| 432.0|          5.1| 30| movie|Hans Majestät får...|Hans Majestät får...|
|Comedy|89083| 512.0|          5.1| 16| movie| Son du vill ha mej| Son du vill ha mej|
|Comedy|89083| 469.0|          6.4| 37| movie| Kantor Ideal| Kantor Ideal|
|Comedy|89083| 682.0|          6.1| 47| movie| Lady from Lisbon| Lady from Lisbon|
|Comedy|89083| 276.0|          4.1| 44| movie|Old Mother Riley'...|Old Mother Riley'...|
|Comedy|89083| 300.0|          6.3| 22| movie| Power| Power|
|Comedy|89083| 308.0|          5.7| 186| movie| Make Your Own Bed| Make Your Own Bed|
|Comedy|89083| 560.0|          6.7| 7| movie| Spoiling the Game|Strich durch die ...|
|Comedy|89083| 843.0|          6.2| 214| movie|Here Come the Mar...|Here Come the Mar...|
|Comedy|89083| 172.0|          5.8| 6| movie|Shepherd of the O...|Shepherd of the O...|
+-----+
only showing top 10 rows

2020-12-07 10:47:32,597 INFO spark.SparkContext: Invoking stop() from shutdown hook
2020-12-07 10:47:32,610 INFO server.AbstractConnector: Stopped Spark@4f84aee3(HTTP/1.1,[http://1.1])([0.0.0.0:4040])
2020-12-07 10:47:32,616 INFO ut.SparkUI: Stopped Spark web UI at http://10.0.2.15:4040
2020-12-07 10:47:32,671 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2020-12-07 10:47:33,102 INFO memory.MemoryStore: MemoryStore cleared
2020-12-07 10:47:33,103 INFO storage.BlockManager: BlockManager stopped
2020-12-07 10:47:33,108 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2020-12-07 10:47:33,114 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
2020-12-07 10:47:33,145 INFO spark.SparkContext: Successfully stopped SparkContext
2020-12-07 10:47:33,145 INFO util.ShutdownHookManager: Shutdown hook called
2020-12-07 10:47:33,147 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-202aadd3-764f-466b-931b-44327079f586
2020-12-07 10:47:33,158 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-202aadd3-764f-466b-931b-44327079f586/pyspark-7a0addb1-d22b-45e0-8660-30a90cd
fefaz
2020-12-07 10:47:33,163 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-f80aafb1-5b37-452e-90fb-c3753c0a3b80
hduser@abhlyogan-VirtualBox: ~/lmdb $

```

We find that comedy is the most liked genre.

Graph Plotting:

A bar plot for the most rated movie:

```
plot = most Rated_Movie()
```

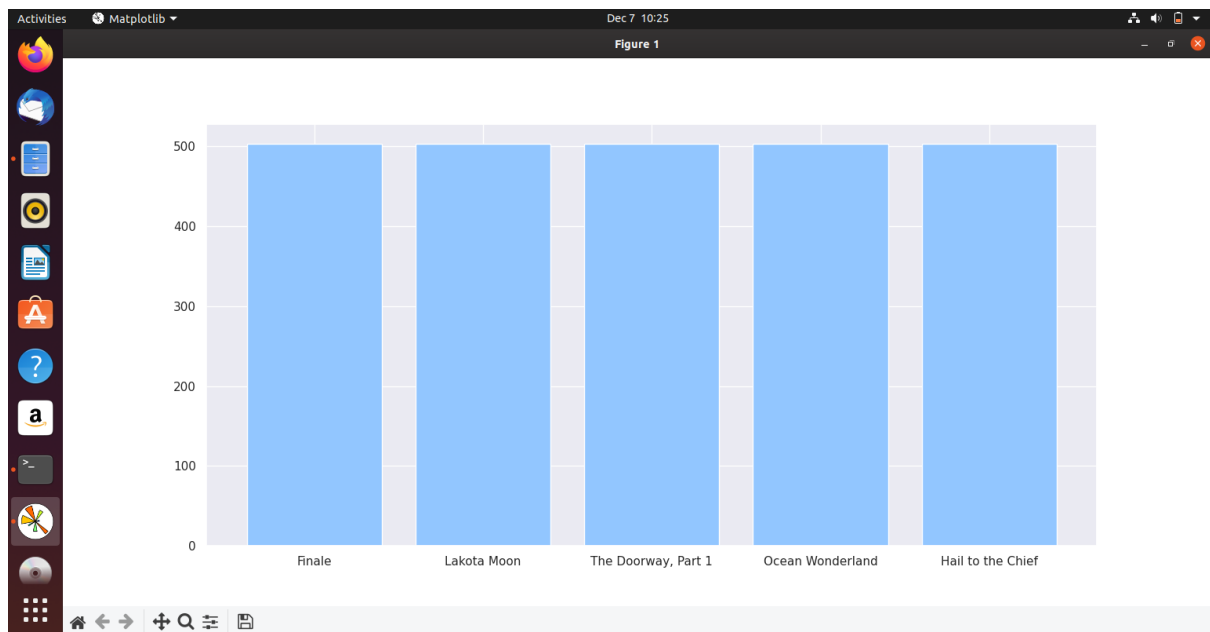
```
grouping = plot.orderBy("count", descending = False).select(['primaryTitle', 'count']).limit(5)
```

```
s = grouping.toPandas()['primaryTitle'].values.tolist()
```

```
t = grouping.toPandas()['count'].values.tolist()
```

```
plt.bar(s,t)
```

```
plt.show()
```



Plotting a histogram plot for the most liked genre:

```
b = most_liked_genre()
```

```
grouping = b.orderBy("count", ascending = False).select(['genres', 'count']).limit(5)
```

```
s = grouping.toPandas()['genres'].values.tolist()
```

```
t = grouping.toPandas()['count'].values.tolist()
```

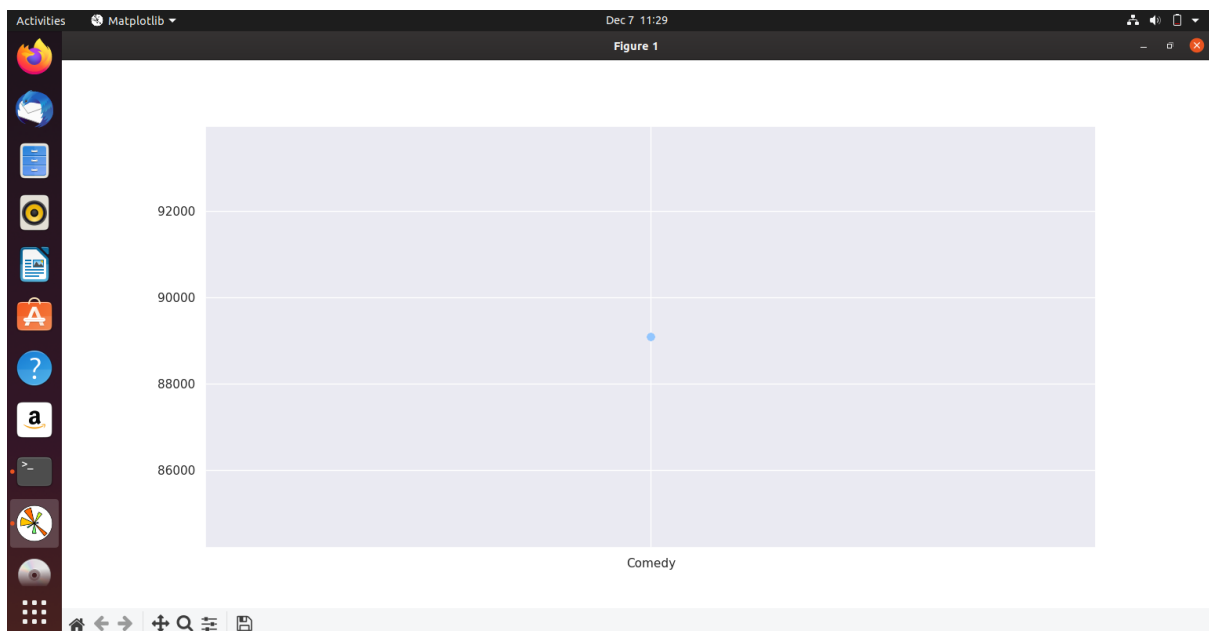
```
#plt.scatter(s,t)
```

`#plt.show()`



Since comedy was the only most liked genre we got the plot for only comedy genre.

scatter plot for genre and avg rating:



ALS MODEL:

#Now we perform alternating least squares method , userid is not the only factor affecting the rating ,genre also plays a role , genre is the latent factor in our project.

```
als= ALS(maxIter=10, regParam=0.5, userCol="userID",itemCol="tconst",
ratingCol="averageRating", nonnegative = True, implicitPrefs=False,
coldStartStrategy="drop")
```

```
train, test = joined_table.randomSplit([0.8, 0.2])
```

#training the model

```
alsModel = als.fit(train)
```

#generating predictions

```
prediction = alsModel.transform(test)
```

```
prediction.show(10)
```

The screenshot shows a terminal window with Spark logs and a table of movie predictions. The logs indicate the completion of the ALS model training and the generation of predictions. The table displays the top 10 rows of predictions, including columns for tconst, averageRating, userID, titleType, primaryTitle, originalTitle, genres, and prediction.

tconst	averageRating	userID	titleType	primaryTitle	originalTitle	genres	prediction
148.0	8.3	31	video	Go: A Film About ...	Go: A Film About ...	Documentary	6.3388205
148.0	4.9	31	movie	Goldörne bent	Goldörne bent	Comedy	6.3388205
148.0	5.1	31	video	Peeping in a Girl...	Peeping in a Girl...	Short	6.3388205
148.0	7.8	85	tvEpisode	The Courtesans	The Courtesans	Comedy,Drama	6.3580084
148.0	8.2	85	tvEpisode	Flying on Empty	Flying on Empty	Crime,Documentary...	6.3580084
148.0	4.2	65	tvEpisode	Richard Ramirez: ...	Richard Ramirez: ...	Biography,Documen...	6.3704777
148.0	5.9	53	tvEpisode	Seize the Day	Seize the Day	Action,Adventure,...	6.316226
148.0	5.7	296	tvEpisode	Escape	Escape	Comedy,Crime,Fantasy	6.2399054
148.0	5.0	108	tvMovie	Little Spirit: Ch...	Little Spirit: Ch...	Animation,Family,...	6.104413
148.0	7.4	34	tvEpisode	How Hitler's Body...	How Hitler's Body...	Documentary,History	6.3774376

Root mean square value turns out to be 2.2740 which is a decent value , taking into consideration our dataset which was sparse and had NAN values.

```
evaluator = RegressionEvaluator(metricName="mse", labelCol="averageRating",
predictionCol="prediction")
```

```
mse = evaluator.evaluate(prediction)
```



```
print("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx")
```

```
print(mse)
```

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2.274052029222673
2020-12-07 17:07:23,285 INFO spark.SparkContext: Invoking stop() from sh
2020-12-07 17:07:23,305 INFO server.AbstractConnector: Stopped Spark@1ae
2020-12-07 17:07:23,311 INFO ui.SparkUI: Stopped Spark web UI at http://
2020-12-07 17:07:23,353 INFO spark.MapOutputTrackerMasterEndpoint: MapOu
2020-12-07 17:07:23,605 INFO memory.MemoryStore: MemoryStore cleared
2020-12-07 17:07:23,607 INFO storage.BlockManager: BlockManager stopped
2020-12-07 17:07:23,616 INFO storage.BlockManagerMaster: BlockManagerMas
2020-12-07 17:07:23,655 INFO scheduler.OutputCommitCoordinator$OutputCom
2020-12-07 17:07:23,672 INFO spark.SparkContext: Successfully stopped Sp
2020-12-07 17:07:23,672 INFO util.ShutdownHookManager: Shutdown hook cal
2020-12-07 17:07:23,674 INFO util.ShutdownHookManager: Deleting director
c4892
2020-12-07 17:07:23,691 INFO util.ShutdownHookManager: Deleting director
2020-12-07 17:07:23,700 INFO util.ShutdownHookManager: Deleting director
hduser@abhilyogan-VirtualBox: ~/imdb$
```

recommendation output of top 3 movies for all the users.:

```
recommended_movie_df = alsModel.recommendForAllUsers(3)
```

```
recommended_movie_df.show(10, False)
```

```
Activities Terminal
Dec 6 14:57
hduser@abhilyogan-VirtualBox: ~/imdb

2020-12-06 14:57:28,376 INFO executor.Executor: Finished task 0.0 in stage 122.0 (TID 1607). 6596 bytes result sent to driver
2020-12-06 14:57:28,376 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 122.0 (TID 1607) in 160 ms on 10.0.2.15 (executor driver) (1/1)
2020-12-06 14:57:28,376 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 122.0, whose tasks have all completed, from pool
2020-12-06 14:57:28,377 INFO scheduler.DAGScheduler: ResultStage 122 (showString at NativeMethodAccessorImpl.java:0) finished in 0.182 s
2020-12-06 14:57:28,378 INFO scheduler.DAGScheduler: Job 13 is finished. Cancelling potential speculative or zombie tasks for this job
2020-12-06 14:57:28,381 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 122: Stage finished
2020-12-06 14:57:28,382 INFO scheduler.DAGScheduler: Job 13 finished: showString at NativeMethodAccessorImpl.java:0, took 11.039861 s
2020-12-06 14:57:28,395 INFO codegen.CodeGenerator: Code generated in 11.264418 ms
+-----+
|userID|recommendations|
+-----+
|1580 |[ [911, 6.332087], [420, 6.3067093], [552, 6.2979646] ] |
|4900 |[ [911, 6.5055256], [420, 6.479452], [552, 6.470468] ] |
|7340 |[ [911, 7.3480457], [420, 7.3191943], [552, 7.3090453] ] |
|15790 |[ [911, 6.1296306], [420, 6.1850634], [552, 6.0965908] ] |
|71510 |[ [911, 6.195559], [420, 6.170728], [552, 6.1621723] ] |
|471 |[ [911, 6.4241652], [420, 6.3984184], [552, 6.3895464] ] |
|1591 |[ [911, 6.5079], [420, 6.4818172], [552, 6.47283] ] |
|4101 |[ [911, 7.0331755], [420, 7.0049877], [552, 6.9952745] ] |
|25591 |[ [911, 7.02045], [420, 6.992313], [552, 6.9826174] ] |
|31951 |[ [911, 6.750786], [420, 6.72373], [552, 6.714407] ] |
+-----+
only showing top 10 rows

2020-12-06 14:57:28,567 INFO spark.SparkContext: Invoking stop() from shutdown hook
2020-12-06 14:57:28,584 INFO server.AbstractConnector: Stopped Spark@4f84aee3(HTTP/1.1,[http/1.1]){0.0.0.0:4040}
2020-12-06 14:57:28,603 INFO ui.SparkUI: Stopped Spark web UI at http://10.0.2.15:4040
2020-12-06 14:57:28,609 INFO storage.BlockManagerInfo: Removed broadcast_57_piece0 on 10.0.2.15:37745 in memory (size: 43.4 KiB, free: 357.4 MiB)
2020-12-06 14:57:28,672 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2020-12-06 14:57:28,866 INFO memory.MemoryStore: MemoryStore cleared
2020-12-06 14:57:28,866 INFO storage.BlockManager: BlockManager stopped
2020-12-06 14:57:28,879 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2020-12-06 14:57:28,953 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
2020-12-06 14:57:28,983 INFO spark.SparkContext: Successfully stopped SparkContext
2020-12-06 14:57:28,983 INFO util.ShutdownHookManager: Shutdown hook called
2020-12-06 14:57:28,984 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-9e995f8a-81b5-43e4-b2c3-86587696aa30/pyspark-919b53db-4dbc-4ddb-a7da-05e583e
a0543
2020-12-06 14:57:28,988 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-b6457813-f5e2-4bba-8abd-c50732c97c3e
2020-12-06 14:57:28,992 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-9e995f8a-81b5-43e4-b2c3-86587696aa30
hduser@abhilyogan-VirtualBox: ~/imdb$
```

Conclusions:

Considering the dataset our recommender system provides fairly good results. Adding more valuable data to the dataset could make the recommender

system even better considering the fact that how our dataset was filled with garbage values.