Assignment 4

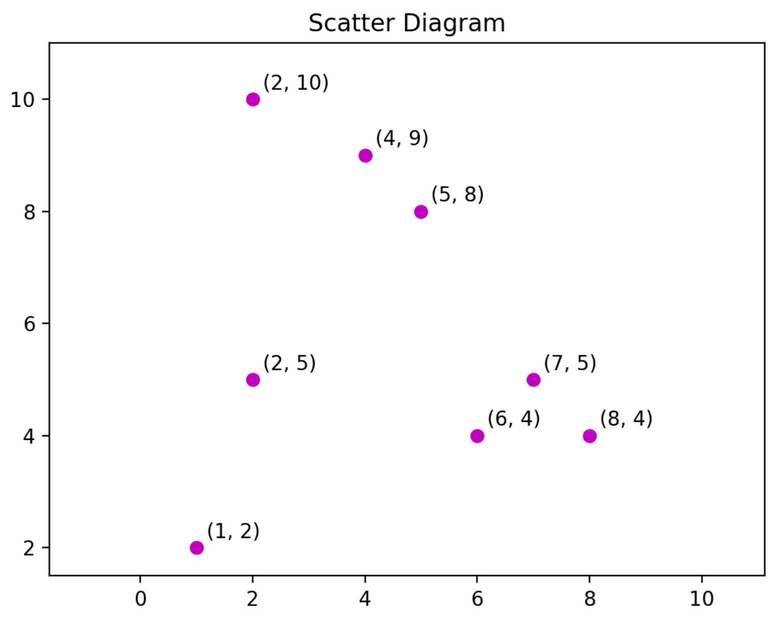
Below is my solution for Part #1 and #2.

Part #1
Objective:

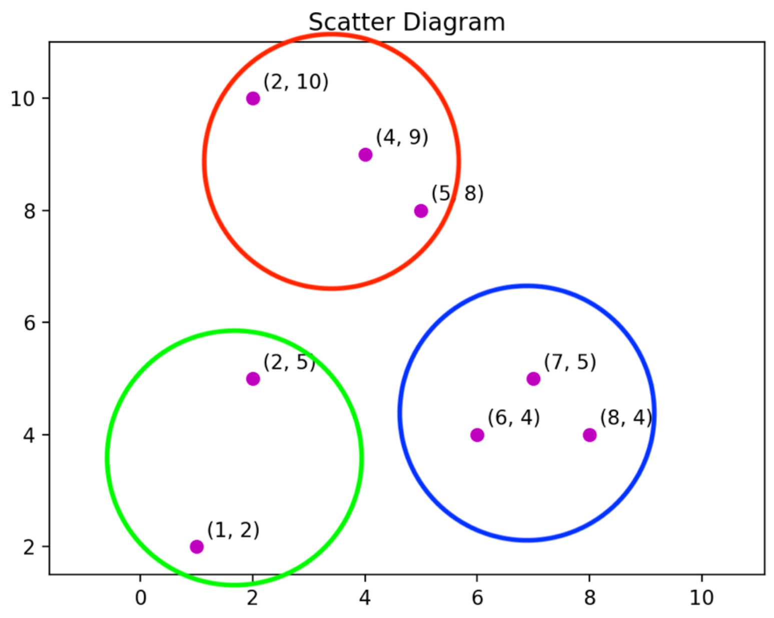
1. K-Means algorithm:

Consider the following eight points in a 2-dimensional space: {(2, 10); (2, 5); (8, 4); (5, 8); (7, 5); (6, 4); (1, 2); (4, 9)}. Suppose we plan to use the Euclidean distance metric, and we are interested in clustering these points into 3 clusters.

a. Plot the data points to see what appropriate clusters might be.



b. How many clusters might be? What are the contents of each cluster?



From the scatter diagram, I would guess that there might be 3 clusters which are:
C1: (1, 2), (2, 5).
C2: (2, 10), (4, 9), (5, 8).
C3: (6, 4), (7, 5), (8, 4).

c. Form three initial clusters with points {(2, 5), (5, 8), (4, 9)} as initial cluster centers (C1, C2, C3).

	(2, 10)	(2, 5)	(8, 4)	(5, 8)	(7, 5)	(6, 4)	(1, 2)	(4, 9)
(2, 5)	5	0	6.0828	4.2426	5	4.1231	3.1623	4.4721
(5, 8)	3.6056	4.2426	5	0	3.6056	4.1231	7.2111	1.4142
(4, 9)	2.2361	4.4721	6.4031	1.4142	5	5.3852	7.6158	0

For plot (6, 4), it has the equal distance to C1 and C2. Since C1 has lower index, we include (6, 4) in C1.

d. What's the center of the cluster after one iteration?

C1: (2, 5), (6, 4), (1, 2).	New center: (3, 3.67).
C2: (8, 4), (5, 8), (7, 5).	New center: (6.67, 5.67).
C3: (2, 10), (4, 9).	New center: (3, 9.5).

e. What's the center of the cluster after one 2nd iteration?

	(2, 10)	(2, 5)	(8, 4)	(5, 8)	(7, 5)	(6, 4)	(1, 2)	(4, 9)
(3, 3.67)	6.4118	1.6667	5.0111	4.7726	4.2164	3.0185	2.6034	5.4263
(6.67, 5.67)	6.3683	4.714	2.1344	2.8674	0.7454	1.7951	6.7495	4.2687
(3, 9.5)	1.118	4.6098	7.433	2.5	6.0208	6.265	7.7621	1.118

C1: (2, 5), (1, 2).	New center: (1.5, 3.5).
C2: (8, 4), (7, 5), (6, 4).	New center: (7, 4.33).

C3: (2, 10), (5, 8), (4, 9).   New center: (3.67, 9).

f. What's the center of the cluster after one 3rd iteration?

	(2, 10)	(2, 5)	(8, 4)	(5, 8)	(7, 5)	(6, 4)	(1, 2)	(4, 9)
(1.5, 3.5)	6.5192	1.5811	6.5192	5.7009	5.7009	4.5277	1.5811	6.0415
(7, 4.33)	7.5572	5.0442	1.0541	4.1767	0.6667	1.0541	6.4377	5.5478
(3.67, 9)	1.9437	4.3333	6.6165	1.6667	5.2068	5.5176	7.4907	0.3333

C1: (2, 5), (1, 2).   New center: (1.5, 3.5).
C2: (8, 4), (7, 5), (6, 4).   New center: (7, 4.33).
C3: (2, 10), (5, 8), (4, 9).   New center: (3.67, 9).
Centers remain unchanged.

g. Compare the results with the section b

Result in section b:
C1: (1, 2), (2, 5).
C2: (2, 10), (4, 9), (5, 8).
C3: (6, 4), (7, 5), (8, 4).
Result after clusting:
C1: (2, 5), (1, 2).
C2: (8, 4), (7, 5), (6, 4).
C3: (2, 10), (5, 8), (4, 9).
The result is just the same as I predicted in section b.

h. How many iterations are required for the clusters to converge?

2 iterations are needed to converge.
Because the clusters remain unchanged in 3rd iteration.

i. What are the resulting centers and resulting clusters? (K=3)

C1: (2, 5), (1, 2).   New center: (1.5, 3.5).
C2: (8, 4), (7, 5), (6, 4).   New center: (7, 4.33).
C3: (2, 10), (5, 8), (4, 9).   New center: (3.67, 9).

2. Hierarchical algorithm:

Use the similarity matrix in Table 1 to perform single and complete link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged. (with enough explanation and calculation)

Single link:

	P1	P2	P3	P4	P5
P1	1.00	0.10	0.41	0.55	0.35

P2	0.10	1.00	0.64	0.47	0.98
P3	0.41	0.64	1.00	0.44	0.85
P4	0.55	0.47	0.44	1.00	0.76
P5	0.35	0.98	0.85	0.76	1.00

Merge P2 and P3.

	P1	P2P5	P3	P4
P1	1.00	0.35	0.41	0.55
P2P5	0.35	1.00	0.85	0.76
P3	0.41	0.85	1.00	0.44
P4	0.55	0.76	0.44	1.00

Merge P2P5 and P3

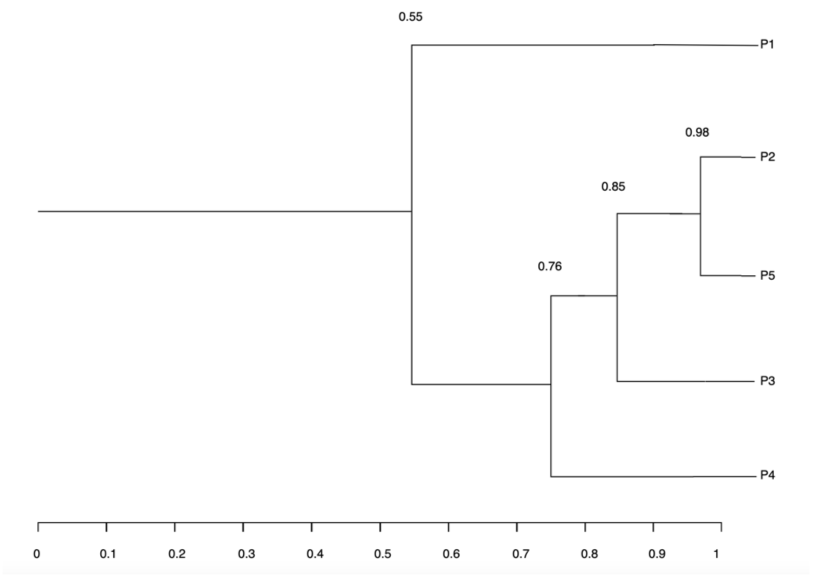
	P1	P2P3P5	P4
P1	1.00	0.41	0.55
P2P3P5	0.41	1.00	0.76
P4	0.55	0.76	1.00

Merge P2P3P5 and P4

	P1	P2P3P4P5
P1	1.00	0.55
P2P3P4P5	0.55	1.00

Merge P2P3P4P5 and P1

Hence, we get the dendrogram:



Complete link:

	P1	P2	P3	P4	P5
P1	1.00	0.10	0.41	0.55	0.35
P2	0.10	1.00	0.64	0.47	0.98
P3	0.41	0.64	1.00	0.44	0.85
P4	0.55	0.47	0.44	1.00	0.76
P5	0.35	0.98	0.85	0.76	1.00

Merge P2 and P3.

	P1	P2P5	P3	P4
P1	1.00	0.10	0.41	0.55
P2P5	0.10	1.00	0.64	0.47
P3	0.41	0.64	1.00	0.44
P4	0.55	0.47	0.44	1.00

Merge P2P5 and P3

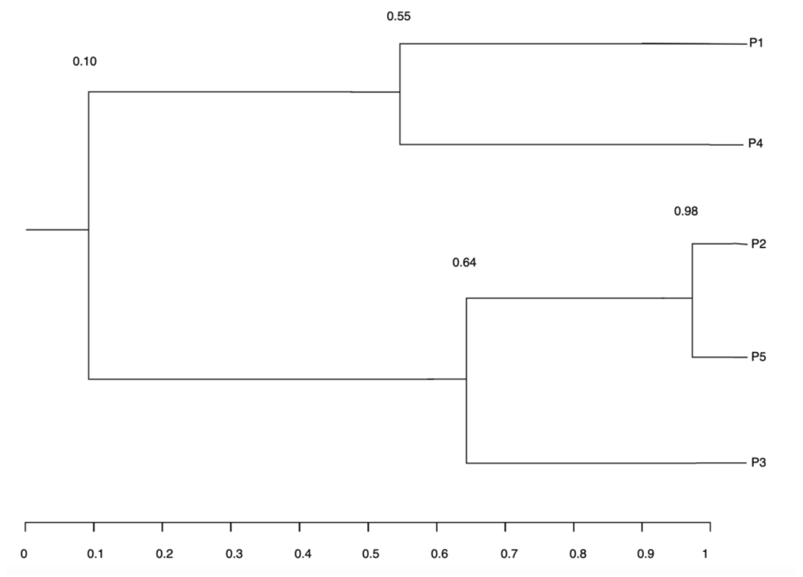
	P1	P2P3P5	P4
P1	1.00	0.10	0.55
P2P3P5	0.10	1.00	0.44
P4	0.55	0.44	1.00

Merge P1 and P4

	P1P4	P2P3P5
P1P4	1.00	0.10
P2P3P5	0.10	1.00

Merge P1P4 and P2P3P5

Hence, we get the dendrogram:



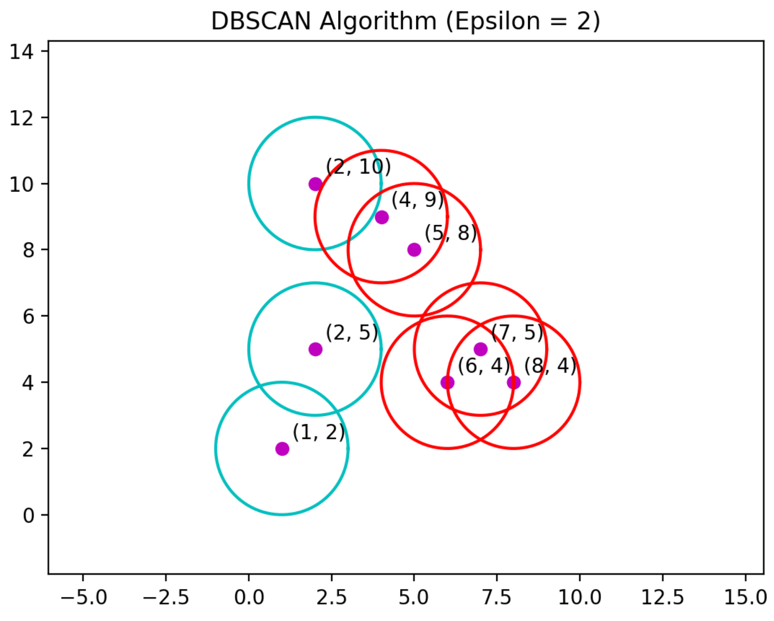
3. DBSCAN algorithm

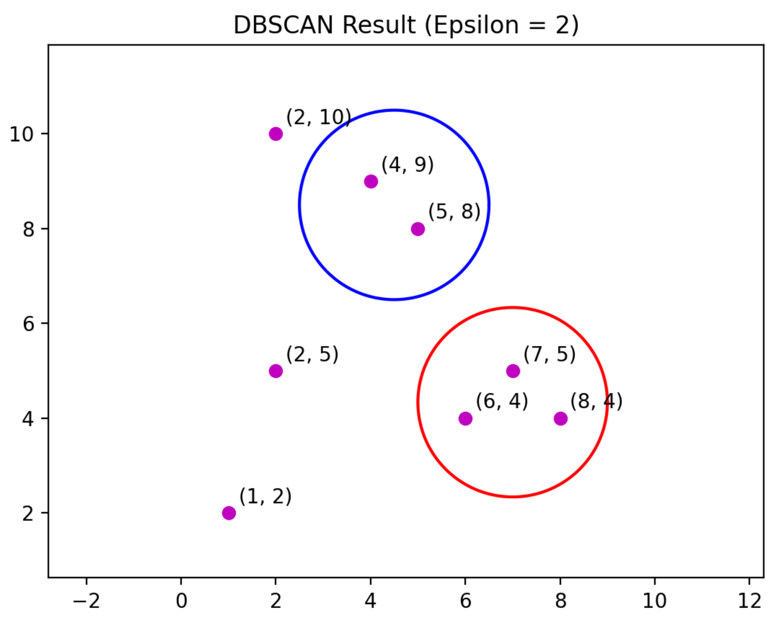
Consider the following eight point in a 2-dimensional space: {(2, 10); (2, 5); (8, 4); (5, 8); (7, 5); (6, 4); (1, 2); (4, 9)}. Suppose we use the Euclidean distance metric.

a. If Epsilon is 2 and min_samples is 2, what are the clusters that DBSCAN would discover. Plot the discovered clusters.

	(2, 10)	(2, 5)	(8, 4)	(5, 8)	(7, 5)	(6, 4)	(1, 2)	(4, 9)
(2, 10)	0	5	8.4853	3.6056	7.0711	7.2111	8.0623	2.2361
(2, 5)	5	0	6.0828	4.2426	5	4.1231	3.1623	4.4721
(8, 4)	8.4853	6.0828	0	5	1.4142	2	7.2801	6.4031
(5, 8)	3.6056	4.2426	5	0	3.6056	4.1231	7.2111	1.4142
(7, 5)	7.0711	5	1.4142	3.6056	0	1.4142	6.7082	5
(6, 4)	7.2111	4.1231	2	4.1231	1.4142	0	5.3852	5.3852
(1, 2)	8.0623	3.1623	7.2801	7.2111	6.7082	5.3852	0	7.6158
(4, 9)	2.2361	4.4721	6.4031	1.4142	5	5.3852	7.6158	0

Since the epsilon is 2 and min sample size is 2, we get:
C1: (8, 4), (7, 5), (6, 4).
C2: (5, 8), (4, 9).



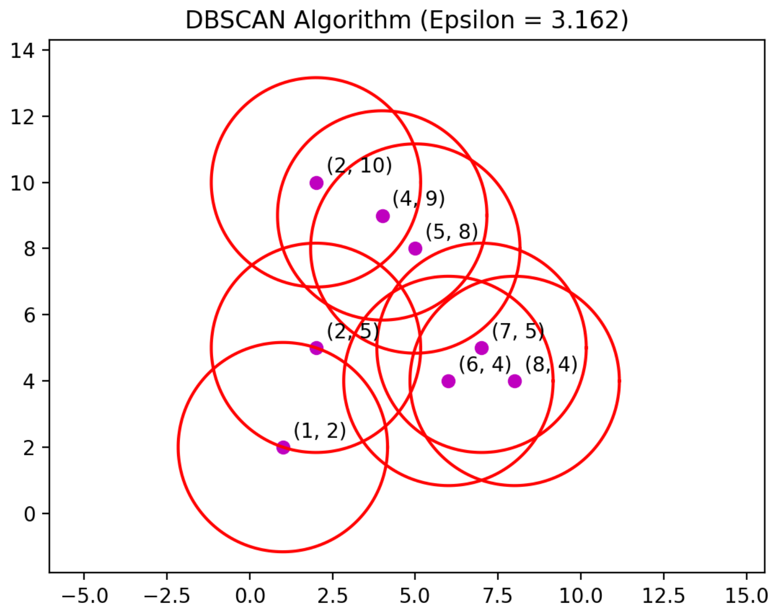


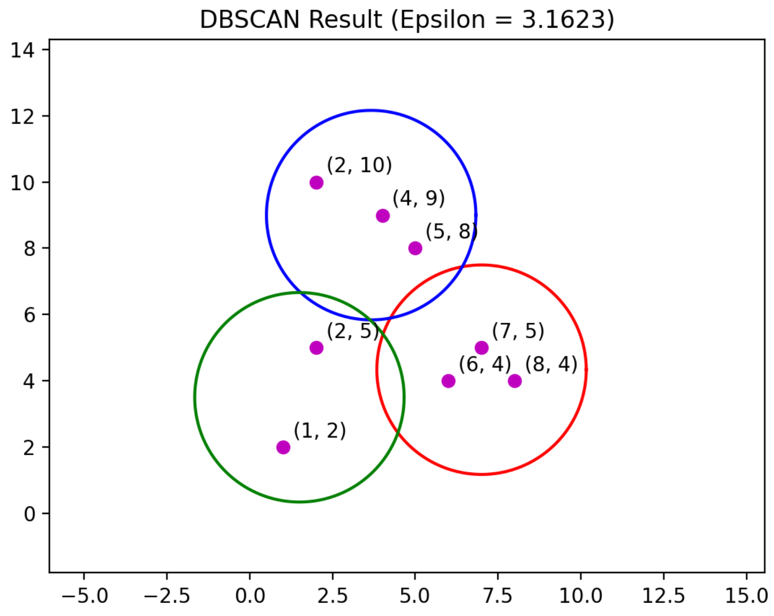
b. What if Epsilon is increased to $\sqrt{10}$?

$$\sqrt{10} \approx 3.1623$$

	(2, 10)	(2, 5)	(8, 4)	(5, 8)	(7, 5)	(6, 4)	(1, 2)	(4, 9)
(2, 10)	0	5	8.4853	3.6056	7.0711	7.2111	8.0623	2.2361
(2, 5)	5	0	6.0828	4.2426	5	4.1231	3.1623	4.4721
(8, 4)	8.4853	6.0828	0	5	1.4142	2	7.2801	6.4031
(5, 8)	3.6056	4.2426	5	0	3.6056	4.1231	7.2111	1.4142
(7, 5)	7.0711	5	1.4142	3.6056	0	1.4142	6.7082	5
(6, 4)	7.2111	4.1231	2	4.1231	1.4142	0	5.3852	5.3852
(1, 2)	8.0623	3.1623	7.2801	7.2111	6.7082	5.3852	0	7.6158
(4, 9)	2.2361	4.4721	6.4031	1.4142	5	5.3852	7.6158	0

Since the epsilon is $\sqrt{10}$ and min sample size is 2, we get:
C1: (8, 4), (7, 5), (6, 4).
C2: (2, 10), (5, 8), (4, 9).
C3: (2, 5), (1, 2).


DBSCAN Algorithm (Epsilon = 3.162)
(2, 10)
(4, 9)
(5, 8)
(2, 5)
(7, 5)
(6, 4)
(8, 4)
(1, 2)
14
12
10
8
6
4
2
0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0


DBSCAN Result (Epsilon = 3.1623)
(2, 10)
(4, 9)
(5, 8)
(2, 5)
(7, 5)
(6, 4)
(8, 4)
(1, 2)
14
12
10
8
6
4
2
0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0

Part #2: BigTabe and Cassandra

Q1. Compare BigTable with Cassandra.
Solution:
Bigtable is a compressed, high performance, proprietary data storage system built on Google File System, Chubby Lock Service, SSTable (log-structured storage like LevelDB) and a few other Google technologies.
Apache Cassandra is a distributed NoSQL database that delivers continuous availability, high performance, and linear scalability that successful applications require. When Cassandra started out, its data model was indeed based on BigTable's.
BigTable is built on GFS, which was designed for consistency. Cassandra's design relies on the local filesystem for performance. There are tradeoffs to both approaches. GFS gave BigTable the replication, consistency, etc for free. While Cassandra had to implement them all from scratch.

Q2. What is Cassandra?
Solution:
Cassandra is a is a distributed NoSQL database that is highly scalable, eventually consistent, distributed, fault tolerant, structured key-value store.

Q3. Explain the concept of tunable consistency in Cassandra.
Solution:
Tunable Consistency means that the consistency levels can be set for each read and write request. Therefore, Cassandra provides a lot of control over how consistent data. Some queries are allowed to be immediately consistent and other queries are allowed to be eventually consistent.

Q4. Define memtable.
Solution:
A memory cache to store the in-memory copy of the data. It is a write-back cache of data partitions that Cassandra looks up by key. Each node has a memtable for each CQL table. The memtable accumulates writes and provides read for data which are not yet stored to disk. The more a table is used, the larger its memtable needs to be. Cassandra can dynamically allocate the right amount of memory for the memtable or you can manage the amount of memory being utilized yourself. The memtable, unlike a write-through cache, stores write until reaching a limit, and then is flushed.

Q5. What is SSTable? How is it different from other relational table.
Solution:
SSTable expands to 'Sorted String Table,' which refers to an important data file in Cassandra and accepts regular written memtables. They are stored on disk and exist for each Cassandra table. Exhibiting immutability, SStables do not allow any further addition and removal of data items once written. For each SSTable, Cassandra creates three separate files like partition index, partition summary and a bloom filter.

Q6. Explain CAP theorem.
Solution:

CAP theorem states that it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees:
Consistency: Every read receives the most recent write or an error
Availability: Every request receives a (non-error) response, without the guarantee that it contains the most recent write
Partition tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes

Q7. Describe difference between Tablet Server and Tablets.
Solution:
A tablet server stores and serves tablets to clients. For a given tablet, one tablet server acts as a leader and the others serve follower replicas of that tablet. Only leaders service write requests, while leaders or followers each service read requests. Leaders are elected using Raft consensus. One tablet server can serve multiple tablets, and one tablet can be served by multiple tablet servers.

Part 3: Programming

Recommendation Systems.

Use Collaborative filtering to find the accuracy of ALS model. Use ratings.dat file. It contains:
User id :: movie id :: ratings :: timestamp.
Your program should report the accuracy of the model.
For details follow the link: https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html
Please use 60% of the data for training and 40% for testing and report the MSE of the model.

Code:
evaluate_ALS_model.py
Result:
Accuracy (MSE) = 0.8655741766296366