FACULTY OF ENGINEERING

Parallel K-Means using Hadoop

Distributed Systems

Presented by

1- **Islam Yousry Abdelwahid 14**
2- **Andrew Adel Sanad 17**
3- **Seif Eldeen Ehab Mostafa 32**

5/15/2022

Table of Contents

1 Problem Definition

The objective of the assignment is to understand how Hadoop and MapReduce framework operate.

Dealing with a very large amount of data is very hard using classical methods and requires very high computational power, but an easy way is to deal with this data in a parallel manner as offered in the Hadoop and MapReduce framework where processing the data is done in a very easy way of just 2 main functions map and reduce and the processing is parallelized efficiently.

Instead of using the sklearn KMeans operation we will compute the KMean clusters of the iris dataset using Hadoop and MapReduce which performs the KMean algorithm in parallel then we are going to compare the results with the output of the normal sklearn as well as the cluster accuracy.

The main requirements are:

- Implement a paralleled KMean version using Hadoop and MapReduce.
- Run an unparalleled KMean version using sklearn.
- Compare the results between the two in terms of time taken.
- Check the cluster accuracy.

2 Algorithms

For the MapReduce algorithm every function is implemented twice first time is for the first implementation where a random cluster is assigned to each point and the initial centroid for each cluster is calculated.

1. **MapperSetup**

After getting the number of clusters from the context, mapping is done for the first iteration by splitting each point by ‘,’ delimiter then assigning each point a random cluster either 1, 2 or 3 and storing the whole point as a text array in the context.

2. **ReducerSetup**

At first saving the output location, number of clusters and using it to create an array for the centroid and the cluster names for each cluster.

Then for each point we save its classification to the new cluster id for the current iteration then compute the new centroid by summing all the point and dividing them by the total number of points.

At the end we save the new centroid in its file for the first iteration.

3. **Mapper**

After saving the number of current iteration, number of clusters and output path, we use the iteration number to get the centroid of previous iteration saved in its file.

In mapping we parse each point from string to its double value then calculate the Euclidean distance between the point and each centroid to find the smallest distance and assign the new cluster to the point.

4. **Reducer**

After saving important variables including the centroids of the previous iterations, we save the name of clusters for every id then calculate the new centroid for the current iteration then save the new centroid in a file.

5. **StringToTextWritable**

A class used to convert an array of strings to an array of texts in order to be writable in the context.

6. **Converge**

Using the centroid of the current iteration and the centroid of the previous iterations the difference between them is calculated using Euclidean and test if the difference is more than a certain threshold or not.

The program ends whenever the threshold is met, or maximum number of iterations is reached.

7. **updateCentroid**

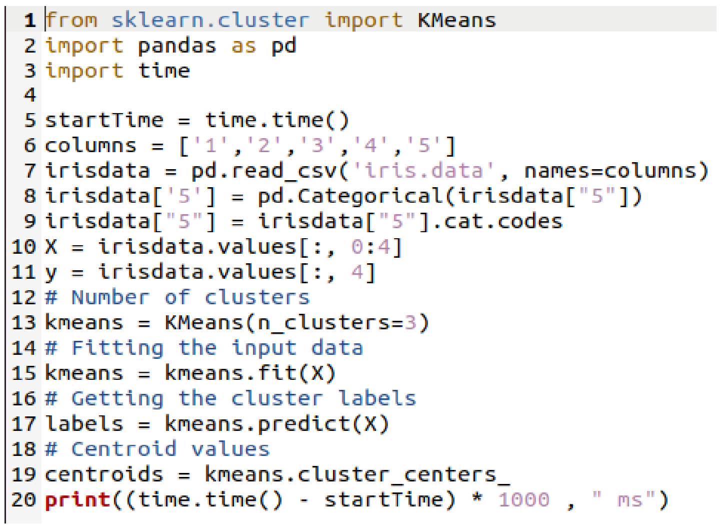
A new file is created using the number of iteration for the current centroid and the 3 centroids are saved in the file with each centroid has the corresponding names of classifications of the point inside the cluster.

8. **getCentroid**

Is used to open the file of the previous iteration and reads the centroid to be used to test for convergence and during mapping to calculate the Euclidean distances.

9. **Unparalleled K-Means algorithm**

- The main algorithm is done where a random centroids are chosen from the points or random clusters are assigned to each point then centroids are calculated, after that running through every point the distance is computed between the point and each centroid using Euclidean distance to assign the point to its new centroid then for the new cluster new centroid is calculated as the mean value of the points in the cluster this is done sequentially until difference centroids of 2 consecutive iterations is zero or less than a certain threshold.
- In our code instead of implementing all that we just use the function KMeans in the sklearn.cluster.

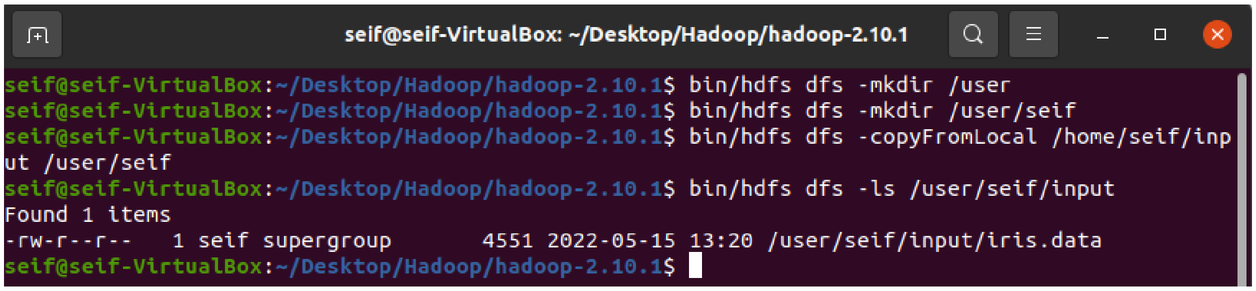


```
from sklearn.cluster import KMeans
import pandas as pd
import time

startTime = time.time()
columns = ['1','2','3','4','5']
irisdata = pd.read_csv('iris.data', names=columns)
irisdata['5'] = pd.Categorical(irisdata["5"])
irisdata["5"] = irisdata["5"].cat.codes
X = irisdata.values[:, 0:4]
y = irisdata.values[:, 4]
# Number of clusters
kmeans = KMeans(n_clusters=3)
# Fitting the input data
kmeans = kmeans.fit(X)
# Getting the cluster labels
labels = kmeans.predict(X)
# Centroid values
centroids = kmeans.cluster_centers_
print((time.time() - startTime) * 1000 , " ms")
```

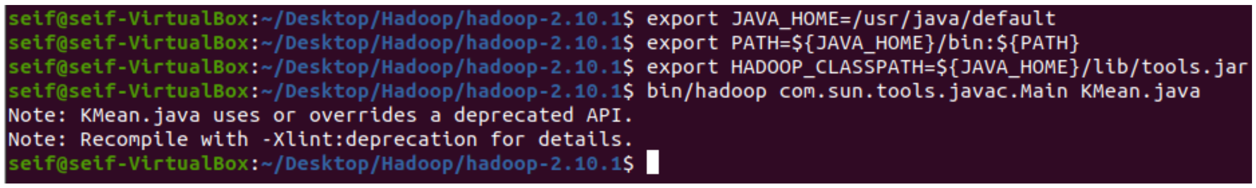
3 Implementation

- Creating directories and moving the data to HDFS



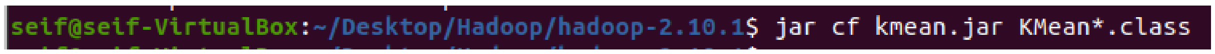
```
seif@seif-VirtualBox: ~/Desktop/Hadoop/hadoop-2.10.1
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hdfs dfs -mkdir /user
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hdfs dfs -mkdir /user/seif
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hdfs dfs -copyFromLocal /home/seif/inp
ut /user/seif
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hdfs dfs -ls /user/seif/input
Found 1 items
-rw-r--r--   1 seif supergroup       4551 2022-05-15 13:20 /user/seif/input/iris.data
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ 
```

- Compiling the java file



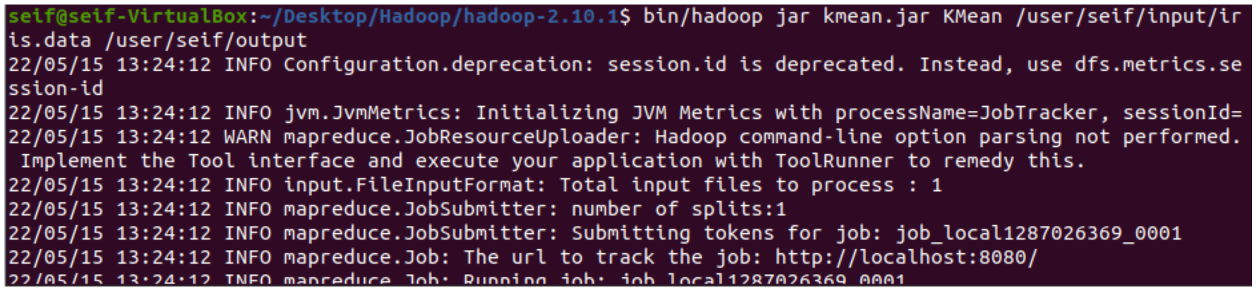
```
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ export JAVA_HOME=/usr/java/default
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ export PATH=${JAVA_HOME}/bin:${PATH}
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hadoop com.sun.tools.javac.Main KMean.java
Note: KMean.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ 
```

- Creating JAR file

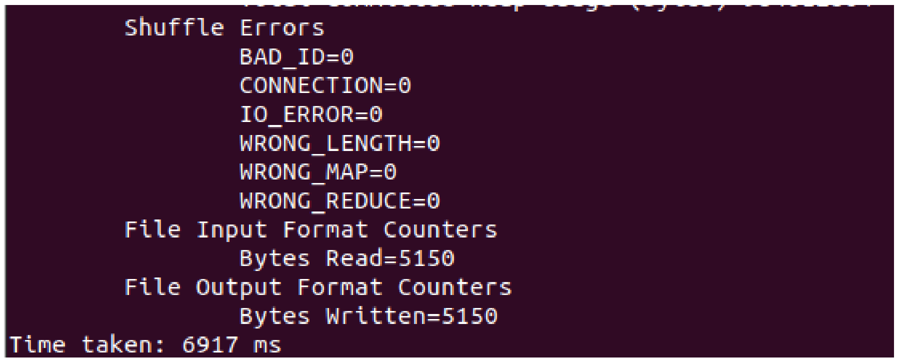


```
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ jar cf kmean.jar KMean*.class
```

- Running the JAR file on the input data

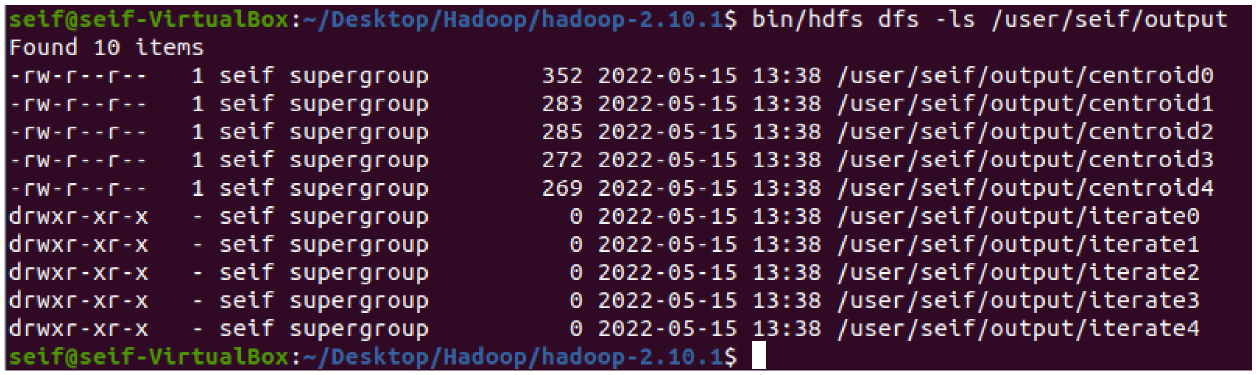


```
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hadoop jar kmean.jar KMean /user/seif/input/ir
is.data /user/seif/output
22/05/15 13:24:12 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.se
ssion-id
22/05/15 13:24:12 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
22/05/15 13:24:12 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed.
 Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/05/15 13:24:12 INFO input.FileInputFormat: Total input files to process : 1
22/05/15 13:24:12 INFO mapreduce.JobSubmitter: number of splits:1
22/05/15 13:24:12 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1287026369_0001
22/05/15 13:24:12 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
22/05/15 13:24:12 INFO mapreduce.Job: Running job: job_local1287026369_0001
```



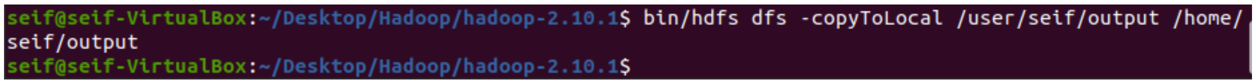
```
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=5150
        File Output Format Counters
                Bytes Written=5150
Time taken: 6917 ms
```

- The output files

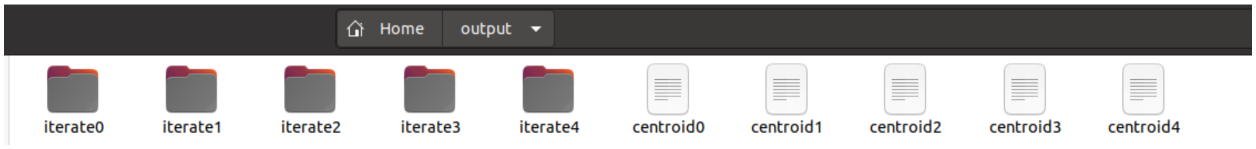


```
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hdfs dfs -ls /user/seif/output
Found 10 items
-rw-r--r--   1 seif supergroup        352 2022-05-15 13:38 /user/seif/output/centroid0
-rw-r--r--   1 seif supergroup        283 2022-05-15 13:38 /user/seif/output/centroid1
-rw-r--r--   1 seif supergroup        285 2022-05-15 13:38 /user/seif/output/centroid2
-rw-r--r--   1 seif supergroup        272 2022-05-15 13:38 /user/seif/output/centroid3
-rw-r--r--   1 seif supergroup        269 2022-05-15 13:38 /user/seif/output/centroid4
drwxr-xr-x   - seif supergroup          0 2022-05-15 13:38 /user/seif/output/iterate0
drwxr-xr-x   - seif supergroup          0 2022-05-15 13:38 /user/seif/output/iterate1
drwxr-xr-x   - seif supergroup          0 2022-05-15 13:38 /user/seif/output/iterate2
drwxr-xr-x   - seif supergroup          0 2022-05-15 13:38 /user/seif/output/iterate3
drwxr-xr-x   - seif supergroup          0 2022-05-15 13:38 /user/seif/output/iterate4
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$
```

- Copying the files to local

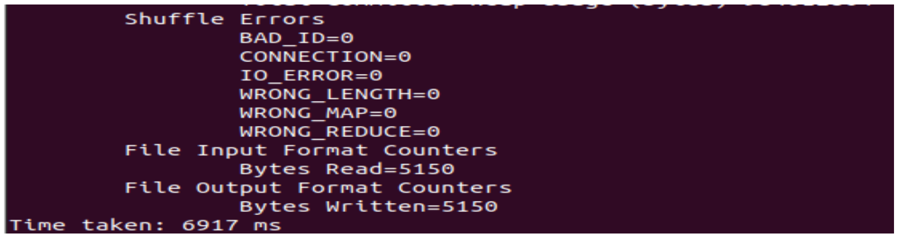


```
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$ bin/hdfs dfs -copyToLocal /user/seif/output /home/
seif/output
seif@seif-VirtualBox:~/Desktop/Hadoop/hadoop-2.10.1$
```



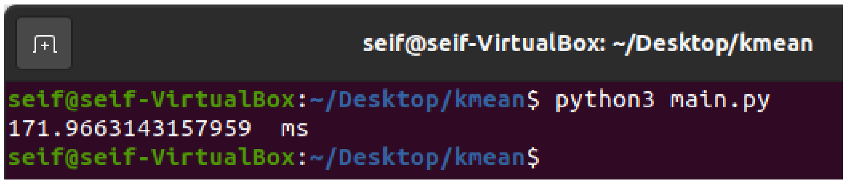
4 Results

- Time taken by the MapReduce framework (notice: the lab is run on an ubuntu virtual machine where a limited RAM and CPU power are used)



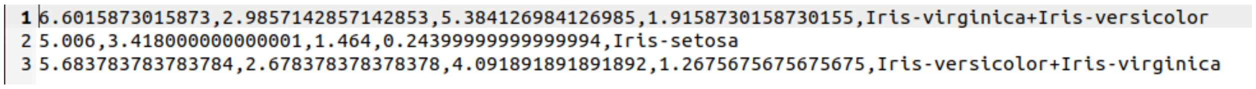
```
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=5150
        File Output Format Counters
                Bytes Written=5150
Time taken: 6917 ms
```

- Time taken by unparalleled K-Mean (notice: sklearn.clusters.kmean is used not a kmean algorithm from scratch where there is some optimization is done in the function for performance)



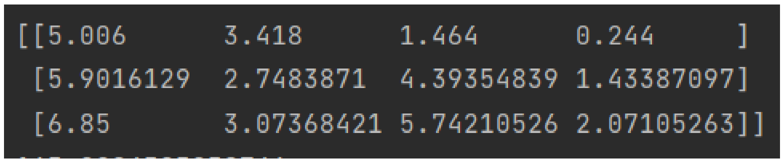
```
seif@seif-VirtualBox: ~/Desktop/kmean
seif@seif-VirtualBox:~/Desktop/kmean$ python3 main.py
171.9663143157959  ms
seif@seif-VirtualBox:~/Desktop/kmean$
```

- Centroids of last iteration of MapReduce



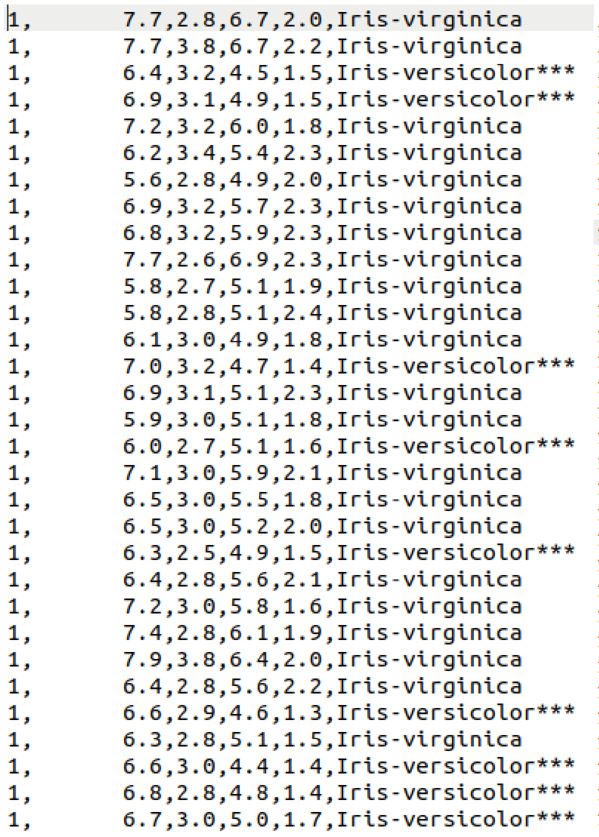
```
6.6015873015873,2.9857142857142853,5.384126984126985,1.9158730158730155,Iris-virginica+Iris-versicolor
5.006,3.418000000000001,1.464,0.24399999999999994,Iris-setosa
5.683783783783784,2.678378378378378,4.091891891891892,1.2675675675675675,Iris-versicolor+Iris-virginica
```

- Centroids of the unparalleled K-Means using sklearn

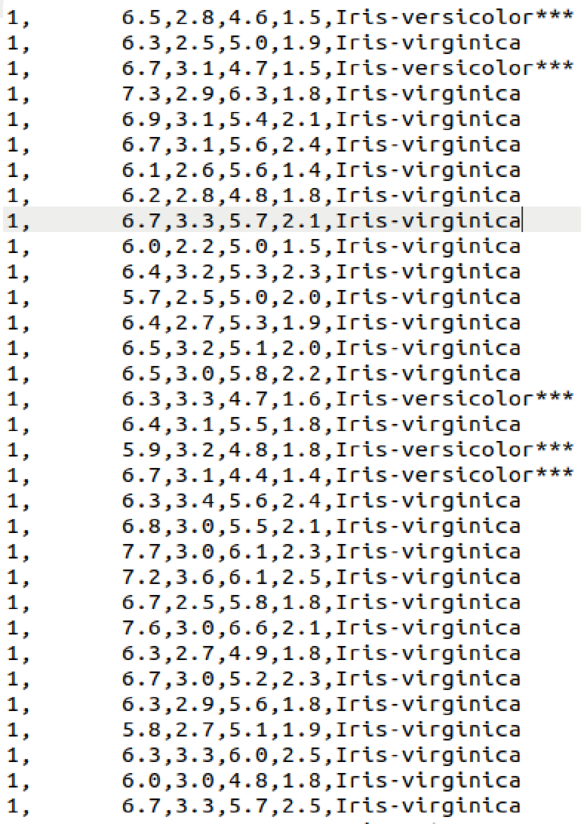


```
[[5.006      3.418      1.464      0.244     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

- Observing the data for cluster accuracy
  - For first cluster it is for Iris-virginica the algorithm clustered some Iris-versicolor as Iris-virginica with error of 22.2%

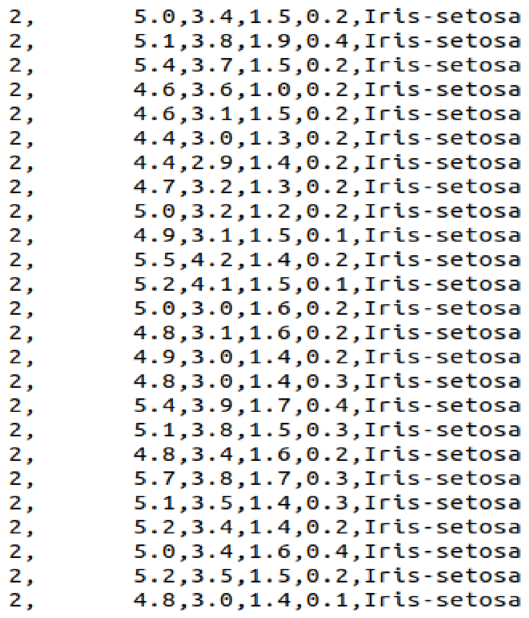


```
1,      7.7,2.8,6.7,2.0,Iris-virginica
1,      7.7,3.8,6.7,2.2,Iris-virginica
1,      6.4,3.2,4.5,1.5,Iris-versicolor***
1,      6.9,3.1,4.9,1.5,Iris-versicolor***
1,      7.2,3.2,6.0,1.8,Iris-virginica
1,      6.2,3.4,5.4,2.3,Iris-virginica
1,      5.6,2.8,4.9,2.0,Iris-virginica
1,      6.9,3.2,5.7,2.3,Iris-virginica
1,      6.8,3.2,5.9,2.3,Iris-virginica
1,      7.7,2.6,6.9,2.3,Iris-virginica
1,      5.8,2.7,5.1,1.9,Iris-virginica
1,      5.8,2.8,5.1,2.4,Iris-virginica
1,      6.1,3.0,4.9,1.8,Iris-virginica
1,      7.0,3.2,4.7,1.4,Iris-versicolor***
1,      6.9,3.1,5.1,2.3,Iris-virginica
1,      5.9,3.0,5.1,1.8,Iris-virginica
1,      6.0,2.7,5.1,1.6,Iris-versicolor***
1,      7.1,3.0,5.9,2.1,Iris-virginica
1,      6.5,3.0,5.5,1.8,Iris-virginica
1,      6.5,3.0,5.2,2.0,Iris-virginica
1,      6.3,2.5,4.9,1.5,Iris-versicolor***
1,      6.4,2.8,5.6,2.1,Iris-virginica
1,      7.2,3.0,5.8,1.6,Iris-virginica
1,      7.4,2.8,6.1,1.9,Iris-virginica
1,      7.9,3.8,6.4,2.0,Iris-virginica
1,      6.4,2.8,5.6,2.2,Iris-virginica
1,      6.6,2.9,4.6,1.3,Iris-versicolor***
1,      6.3,2.8,5.1,1.5,Iris-virginica
1,      6.6,3.0,4.4,1.4,Iris-versicolor***
1,      6.8,2.8,4.8,1.4,Iris-versicolor***
1,      6.7,3.0,5.0,1.7,Iris-versicolor***
```

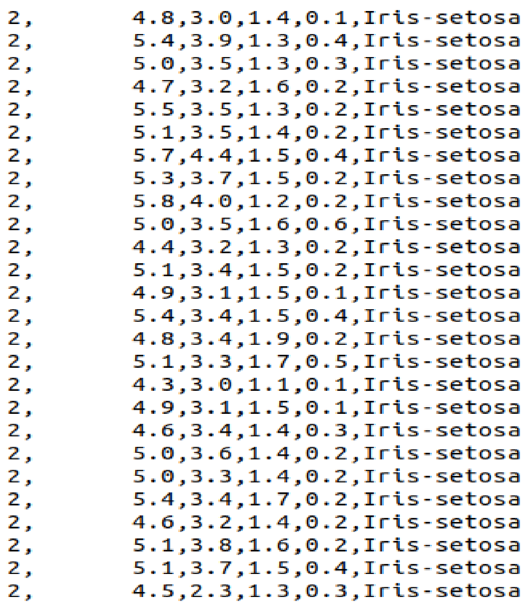


```
1,      6.5,2.8,4.6,1.5,Iris-versicolor***
1,      6.3,2.5,5.0,1.9,Iris-virginica
1,      6.7,3.1,4.7,1.5,Iris-versicolor***
1,      7.3,2.9,6.3,1.8,Iris-virginica
1,      6.9,3.1,5.4,2.1,Iris-virginica
1,      6.7,3.1,5.6,2.4,Iris-virginica
1,      6.1,2.6,5.6,1.4,Iris-virginica
1,      6.2,2.8,4.8,1.8,Iris-virginica
1,      6.7,3.3,5.7,2.1,Iris-virginica
1,      6.0,2.2,5.0,1.5,Iris-virginica
1,      6.4,3.2,5.3,2.3,Iris-virginica
1,      5.7,2.5,5.0,2.0,Iris-virginica
1,      6.4,2.7,5.3,1.9,Iris-virginica
1,      6.5,3.2,5.1,2.0,Iris-virginica
1,      6.5,3.0,5.8,2.2,Iris-virginica
1,      6.3,3.3,4.7,1.6,Iris-versicolor***
1,      6.4,3.1,5.5,1.8,Iris-virginica
1,      5.9,3.2,4.8,1.8,Iris-versicolor***
1,      6.7,3.1,4.4,1.4,Iris-versicolor***
1,      6.3,3.4,5.6,2.4,Iris-virginica
1,      6.8,3.0,5.5,2.1,Iris-virginica
1,      7.7,3.0,6.1,2.3,Iris-virginica
1,      7.2,3.6,6.1,2.5,Iris-virginica
1,      6.7,2.5,5.8,1.8,Iris-virginica
1,      7.6,3.0,6.6,2.1,Iris-virginica
1,      6.3,2.7,4.9,1.8,Iris-virginica
1,      6.7,3.0,5.2,2.3,Iris-virginica
1,      6.3,2.9,5.6,1.8,Iris-virginica
1,      5.8,2.7,5.1,1.9,Iris-virginica
1,      6.3,3.3,6.0,2.5,Iris-virginica
1,      6.0,3.0,4.8,1.8,Iris-virginica
1,      6.7,3.3,5.7,2.5,Iris-virginica
```

  - For second cluster it is for Iris-setosa and the algorithm classified all points correctly with 0% error

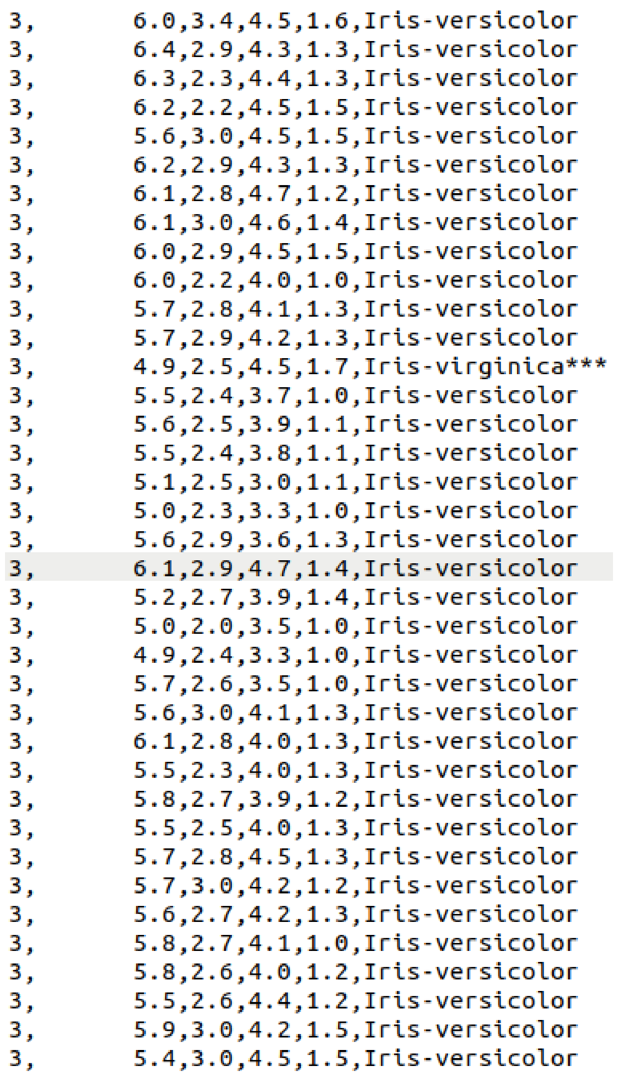


```
2,      5.0,3.4,1.5,0.2,Iris-setosa
2,      5.1,3.8,1.9,0.4,Iris-setosa
2,      5.4,3.7,1.5,0.2,Iris-setosa
2,      4.6,3.6,1.0,0.2,Iris-setosa
2,      4.6,3.1,1.5,0.2,Iris-setosa
2,      4.4,3.0,1.3,0.2,Iris-setosa
2,      4.4,2.9,1.4,0.2,Iris-setosa
2,      4.7,3.2,1.3,0.2,Iris-setosa
2,      5.0,3.2,1.2,0.2,Iris-setosa
2,      4.9,3.1,1.5,0.1,Iris-setosa
2,      5.5,4.2,1.4,0.2,Iris-setosa
2,      5.2,4.1,1.5,0.1,Iris-setosa
2,      5.0,3.0,1.6,0.2,Iris-setosa
2,      4.8,3.1,1.6,0.2,Iris-setosa
2,      4.9,3.0,1.4,0.2,Iris-setosa
2,      4.8,3.0,1.4,0.3,Iris-setosa
2,      5.4,3.9,1.7,0.4,Iris-setosa
2,      5.1,3.8,1.5,0.3,Iris-setosa
2,      4.8,3.4,1.6,0.2,Iris-setosa
2,      5.7,3.8,1.7,0.3,Iris-setosa
2,      5.1,3.5,1.4,0.3,Iris-setosa
2,      5.2,3.4,1.4,0.2,Iris-setosa
2,      5.0,3.4,1.6,0.4,Iris-setosa
2,      5.2,3.5,1.5,0.2,Iris-setosa
2,      4.8,3.0,1.4,0.1,Iris-setosa
```



```
2,      4.8,3.0,1.4,0.1,Iris-setosa
2,      5.4,3.9,1.3,0.4,Iris-setosa
2,      5.0,3.5,1.3,0.3,Iris-setosa
2,      4.7,3.2,1.6,0.2,Iris-setosa
2,      5.5,3.5,1.3,0.2,Iris-setosa
2,      5.1,3.5,1.4,0.2,Iris-setosa
2,      5.7,4.4,1.5,0.4,Iris-setosa
2,      5.3,3.7,1.5,0.2,Iris-setosa
2,      5.8,4.0,1.2,0.2,Iris-setosa
2,      5.0,3.5,1.6,0.6,Iris-setosa
2,      4.4,3.2,1.3,0.2,Iris-setosa
2,      5.1,3.4,1.5,0.2,Iris-setosa
2,      4.9,3.1,1.5,0.1,Iris-setosa
2,      5.4,3.4,1.5,0.4,Iris-setosa
2,      4.8,3.4,1.9,0.2,Iris-setosa
2,      5.1,3.3,1.7,0.5,Iris-setosa
2,      4.3,3.0,1.1,0.1,Iris-setosa
2,      4.9,3.1,1.5,0.1,Iris-setosa
2,      4.6,3.4,1.4,0.3,Iris-setosa
2,      5.0,3.6,1.4,0.2,Iris-setosa
2,      5.0,3.3,1.4,0.2,Iris-setosa
2,      5.4,3.4,1.7,0.2,Iris-setosa
2,      4.6,3.2,1.4,0.2,Iris-setosa
2,      5.1,3.8,1.6,0.2,Iris-setosa
2,      5.1,3.7,1.5,0.4,Iris-setosa
2,      4.5,2.3,1.3,0.3,Iris-setosa
```

- For third cluster it is for Iris- versicolor the algorithm clustered 1 Iris- virginica as Iris-versicolor with error 2.7%



```
3,      6.0,3.4,4.5,1.6,Iris-versicolor
3,      6.4,2.9,4.3,1.3,Iris-versicolor
3,      6.3,2.3,4.4,1.3,Iris-versicolor
3,      6.2,2.2,4.5,1.5,Iris-versicolor
3,      5.6,3.0,4.5,1.5,Iris-versicolor
3,      6.2,2.9,4.3,1.3,Iris-versicolor
3,      6.1,2.8,4.7,1.2,Iris-versicolor
3,      6.1,3.0,4.6,1.4,Iris-versicolor
3,      6.0,2.9,4.5,1.5,Iris-versicolor
3,      6.0,2.2,4.0,1.0,Iris-versicolor
3,      5.7,2.8,4.1,1.3,Iris-versicolor
3,      5.7,2.9,4.2,1.3,Iris-versicolor
3,      4.9,2.5,4.5,1.7,Iris-virginica***
3,      5.5,2.4,3.7,1.0,Iris-versicolor
3,      5.6,2.5,3.9,1.1,Iris-versicolor
3,      5.5,2.4,3.8,1.1,Iris-versicolor
3,      5.1,2.5,3.0,1.1,Iris-versicolor
3,      5.0,2.3,3.3,1.0,Iris-versicolor
3,      5.6,2.9,3.6,1.3,Iris-versicolor
3,      6.1,2.9,4.7,1.4,Iris-versicolor
3,      5.2,2.7,3.9,1.4,Iris-versicolor
3,      5.0,2.0,3.5,1.0,Iris-versicolor
3,      4.9,2.4,3.3,1.0,Iris-versicolor
3,      5.7,2.6,3.5,1.0,Iris-versicolor
3,      5.6,3.0,4.1,1.3,Iris-versicolor
3,      6.1,2.8,4.0,1.3,Iris-versicolor
3,      5.5,2.3,4.0,1.3,Iris-versicolor
3,      5.8,2.7,3.9,1.2,Iris-versicolor
3,      5.5,2.5,4.0,1.3,Iris-versicolor
3,      5.7,2.8,4.5,1.3,Iris-versicolor
3,      5.7,3.0,4.2,1.2,Iris-versicolor
3,      5.6,2.7,4.2,1.3,Iris-versicolor
3,      5.8,2.7,4.1,1.0,Iris-versicolor
3,      5.8,2.6,4.0,1.2,Iris-versicolor
3,      5.5,2.6,4.4,1.2,Iris-versicolor
3,      5.9,3.0,4.2,1.5,Iris-versicolor
3,      5.4,3.0,4.5,1.5,Iris-versicolor
```

5 Conclusion

- Although MapReduce performs the kmean in a paralleled way by performing mapping then reducing the classical sklearn kmean algorithm still had less time taken and better efficiency.
- Although convergence threshold is as low as 0.01 there has been some errors in the first cluster but the errors are not related the MapReduce algorithm it is related to the KMean algorithm itself as a clustering technique.