BIG DATA ANALYTICS

LAB ASSIGNMENT 9

Mahesh Pachare

FINAL YEAR B.TECH

IT 191080054

Aim :

Implement parallel Kmeans clustering using Spark Mlib

Theory :

Apache Spark :

Spark is a cluster computing system. It is faster than other cluster computing systems (such as Hadoop). It provides high-level APIs in Python, Scala, and Java. Parallel jobs are easy to write in Spark. In this article, we will discuss the different components of Apache Spark.

Spark processes a huge amount of datasets and is the foremost active Apache project of the current time. Spark is written in Scala and provides API in Python, Scala, Java, and R. The most vital feature of Apache Spark is its in-memory cluster computing that extends the speed of the data process. Spark is an additional general and quicker processing platform. It helps us to run programs relatively quicker than Hadoop (i.e.) a hundred times quicker in memory and ten times quicker even on the disk. The main features of spark are:

- Multiple Language Support: Apache Spark supports multiple languages; it provides API's written in Scala, Java, Python or R. It permits users to write down applications in several languages.
- Quick Speed: The most vital feature of Apache Spark is its processing speed. It permits the application to run on a Hadoop cluster, up to one hundred times quicker in memory, and ten times quicker on disk
- Runs Everywhere: Spark will run on multiple platforms while not moving the processing speed. It will run on Hadoop, Kubernetes, Mesos, Standalone, and even within the Cloud.
- General Purpose: It is powered by a plethora of libraries for machine learning (i.e.) MLlib, DataFrames, and SQL at the side of Spark Streaming and GraphX. It is allowed to use a mix of those libraries which are coherently associated with the application. The feature of mixing streaming, SQL, and complicated analytics, within the same application, makes Spark a general framework.
- Advanced Analytics: Apache Spark also supports "Map" and "Reduce" that has been mentioned earlier. However, on the side of MapReduce, it supports Streaming data, SQL queries, Graph algorithms, and Machine learning. Thus, Apache Spark may be used to perform advanced analytics.

Spark Machine Learning Library :

- MLlib in Spark's machine learning (ML) library.
- Its goal is to make practical machine learning scalable and easy.
- At a high level, it provides tools such as:
  - ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
  - Featurization: feature extraction, transformation, dimensionality reduction, and selection
  - Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
  - Persistence: saving and load algorithms, models, and Pipelines
  - Utilities: linear algebra, statistics, data handling, etc.

KMeans clustering :

Unsupervised Machine Learning learning is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine's job in this case is to organize unsorted data according to parallels, patterns, and variations.
The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.
We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the Kmeans algorithm; an unsupervised learning algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into. (It will help if you think of items as points in an n-dimensional space). The algorithm will

categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement.
The algorithm works as follows:

- First, we randomly initialize k points, called means or cluster centroids.
- We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.
- We repeat the process for a given number of iterations and at the end, we have our clusters.
- The "points" mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature x, the items have values in [0,3], we will initialize the means with values for x at [0,3])

EXECUTION STEPS AND OUTPUT :

- **Starting all daemons**

```
hadoopuser@hadoop-master:~/spark/spark-3.4.0-bin-hadoop3/sbin$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoopuser in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
```

```
hadoopuser@hadoop-master:~/spark/spark-3.4.0-bin-hadoop3/sbin$ sudo ./start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /home/hadoopuser/spark/spark-3.4.0-bin-had
oop3/logs/spark-root-org.apache.spark.deploy.master.Master-1-hadoop-master.out
```

```
hadoopuser@hadoop-master:~/spark/spark-3.4.0-bin-hadoop3/sbin$ sudo ./start-slave.sh spark://hadoop-m
aster:7077
This script is deprecated, use start-worker.sh
starting org.apache.spark.deploy.worker.Worker, logging to /home/hadoopuser/spark/spark-3.4.0-bin-had
oop3/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-hadoop-master.out
```

- **Putting input data for K-Means Clustering on HDFS**

```
hadoopuser@hadoop-master: ~
hadoopuser@hadoop-master:~$ hdfs dfs -mkdir /exp9
hadoopuser@hadoop-master:~$ hdfs dfs -mkdir /exp9/kmeans-data
hadoopuser@hadoop-master:~$ hdfs dfs -put /home/hadoopuser/exp9/odiBatting.csv /exp9/kmeans-data/
```

- **Python Code**
  Using PySpark and ML-Lib

```
  GNU nano 6.4                                    k-means.py
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StandardScaler
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
import matplotlib.pyplot as plt
# Create a Spark session
spark = SparkSession.builder.appName("KMeansExample").getOrCreate()
# Load data
data = spark.read.csv("/usr/local/exp9/odiBatting.csv", header=True, inferSchema=True)
# Remove missing values
data = data.na.drop()
# Select the features to be used for clustering
selected_cols = ["Mat", "Inns", "NO", "Runs", "Ave", "100", "50", "0"]
data = data.select(selected_cols)
# Convert the features to vectors
assembler = VectorAssembler(inputCols=selected_cols, outputCol="input_features")
data = assembler.transform(data)
# Standardize the data
scaler = StandardScaler(inputCol="input_features", outputCol="features")
scalerModel = scaler.fit(data)
data = scalerModel.transform(data)
# Evaluate clustering by computing Silhouette score
def evaluate_model(k, data):
    # Instantiate a KMeans object
    kmeans = KMeans().setK(k).setSeed(1)
    # Fit the model to the data
    model = kmeans.fit(data)
    # Make predictions
    predictions = model.transform(data)
    # Evaluate clustering by computing Silhouette score
    evaluator = ClusteringEvaluator()
    silhouette = evaluator.evaluate(predictions)
    return silhouette
# Determine the best value of k using the elbow method
scores = []
ks = range(2, 11)
for k in ks:
    score = evaluate_model(k, data)
    scores.append(score)
    print("Silhouette score for k = ", k, ": ", score)
plt.plot(ks, scores, 'bx-')
plt.xlabel('k')
```

```
  GNU nano 6.4                                    k-means.py *
plt.ylabel('Silhouette score')
plt.title('Elbow Method')
plt.show(block=True)
plt.savefig('/usr/local/exp9/graph/graph.png')
# Train the KMeans model with k = 5
k = 3
kmeans = KMeans().setK(k).setSeed(1)
model = kmeans.fit(data)
# Make predictions
predictions = model.transform(data)
# Show which specific objects belong to each cluster
clusters = predictions.groupBy("prediction").agg({"prediction": "count"})
print("Number of objects in each cluster:")
clusters.show()
for i in range(k):
    print("Objects in cluster ", i, ":")
    cluster_i = predictions.filter(predictions.prediction == i)
    cluster_i.show()
# Stop the Spark session
spark.stop()
```

- **The output of the Python script**

```
hduser@master:/usr/local$ python3 k-means.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/04/23 19:39:29 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
23/04/23 19:39:41 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
Silhouette score for k =  2 :  0.5532618938636694
Silhouette score for k =  3 :  0.45969117924453096
Silhouette score for k =  4 :  0.4787442310280957
Silhouette score for k =  5 :  0.4446874185938474
Silhouette score for k =  6 :  0.408444984197167
Silhouette score for k =  7 :  0.3709224638591419
Silhouette score for k =  8 :  0.35232393272093127
Silhouette score for k =  9 :  0.3553682052251737
Silhouette score for k =  10 :  0.33871999294200883
Number of objects in each cluster:
+----------+-----------------+
|prediction|count(prediction)|
+----------+-----------------+
|         1|               20|
|         2|               48|
|         0|               51|
+----------+-----------------+
```

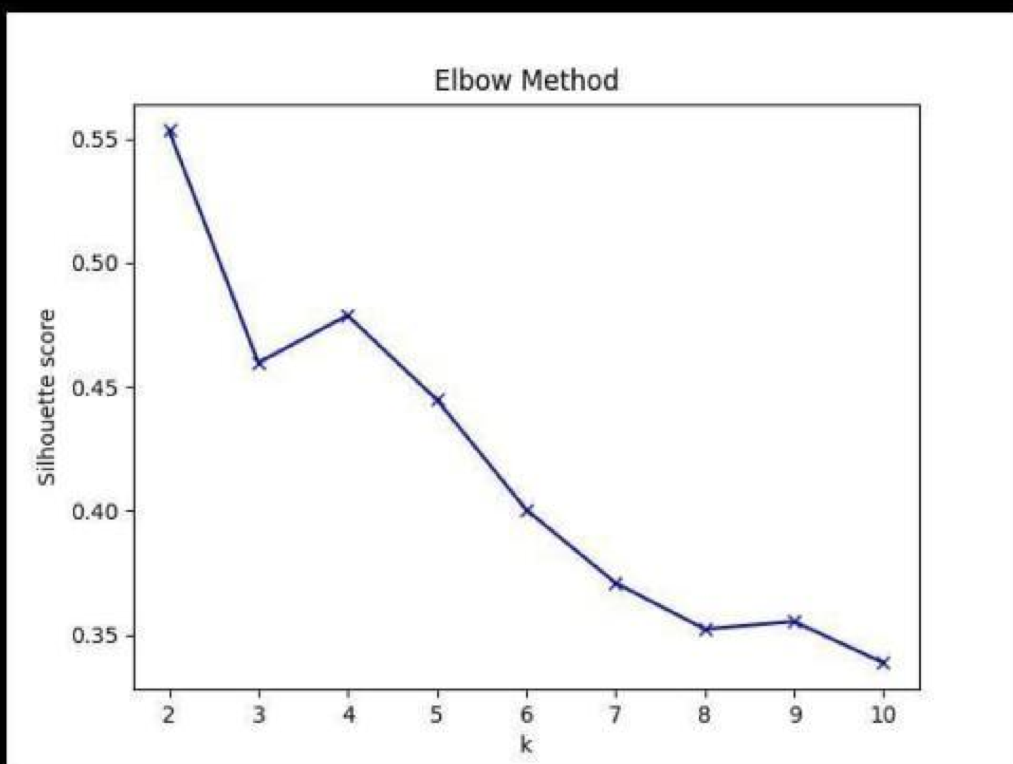
```
Objects in cluster  0 :
+---+----+---------+----+-----+----------+---------------+-----+--------------------+--------------------+----------+
|Mat|Inns|Not-Outs|Runs|  Ave|Centuries|Half Centuries|Ducks|      input_features|            features|prediction|
+---+----+---------+----+-----+----------+---------------+-----+--------------------+--------------------+----------+
|128| 126|       6|5455|45.45|       18|            23|    2|[128.0,126.0,6.0,...|[1.34907700892474...|         0|
|132| 128|       3|5232|41.85|       17|            29|   11|[132.0,128.0,3.0,...|[1.39123566545364...|         0|
|145| 142|       8|6105|45.55|       17|            33|    5|[145.0,142.0,8.0,...|[1.52825129917256...|         0|
|124| 124|       6|5355|45.38|       16|            26|    4|[124.0,124.0,6.0,...|[1.30691835239584...|         0|
|152| 142|      23|6109|51.33|       16|            35|    5|[152.0,142.0,23.0...|[1.60202894809813...|         0|
| 83|  81|      11|3985|56.92|       14|            17|    3|[83.0,81.0,11.0,3...|[0.87479212297463...|         0|
|151| 144|      14|6173|47.48|       13|            39|    5|[151.0,144.0,14.0...|[1.59148928396591...|         0|
|123| 122|       6|4335|37.37|       12|            21|   13|[123.0,122.0,6.0,...|[1.29637868826362...|         0|
|134| 131|       3|4982|38.92|       12|            26|   10|[134.0,131.0,3.0,...|[1.41231499371809...|         0|
|143| 136|      20|5507|47.47|       12|            35|    3|[143.0,136.0,20.0...|[1.50717197090811...|         0|
| 89|  81|       8|3498|47.91|       11|            14|    6|[89.0,81.0,8.0,34...|[0.93803010776798...|         0|
|128| 113|      12|4378|43.34|       11|            25|    5|[128.0,113.0,12.0...|[1.34907700892474...|         0|
|128| 127|      13|5134|45.03|       11|            31|    3|[128.0,127.0,13.0...|[1.34907700892474...|         0|
|145| 142|       3|4317|31.05|       11|            20|   10|[145.0,142.0,3.0,...|[1.52825129917256...|         0|
|147| 143|      11|5238|39.68|       11|            34|   11|[147.0,143.0,11.0...|[1.54933062743701...|         0|
| 83|  78|      10|3599|52.92|       10|            19|    4|[83.0,78.0,10.0,3...|[0.87479212297463...|         0|
|161| 155|      15|6133| 43.8|       10|            36|    9|[161.0,155.0,15.0...|[1.69688592528815...|         0|
| 98|  93|       2|3658|40.19|        9|            20|    9|[98.0,93.0,2.0,36...|[1.03288708495800...|         0|
|117| 115|       6|4357|39.97|        9|            22|    4|[117.0,115.0,6.0,...|[1.23314070347027...|         0|
|148| 123|      23|3872|38.72|        9|            20|   13|[148.0,123.0,23.0...|[1.55987029156923...|         0|
+---+----+---------+----+-----+----------+---------------+-----+--------------------+--------------------+----------+
only showing top 20 rows
```

```
Objects in cluster  1 :
+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+----------+
|Mat|Inns|Not-Outs| Runs|  Ave|Centuries|Half Centuries|Ducks|      input_features|            features|prediction|
+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+----------+
|463| 452|      41|18426|44.83|       49|            96|   20|[463.0,452.0,41.0...|[4.87986449321997...|         1|
|254| 245|      39|12169|59.07|       43|            62|   13|[254.0,245.0,39.0...|[2.67707468958504...|         1|
|375| 365|      39|13704|42.03|       30|            82|   20|[375.0,365.0,39.0...|[3.95237404958421...|         1|
|227| 220|      32| 9205|48.96|       29|            43|   13|[227.0,220.0,32.0...|[2.39250375801497...|         1|
|445| 433|      18|13430|32.36|       28|            68|   34|[445.0,433.0,18.0...|[4.69015053883993...|         1|
|301| 294|      17|10480|37.83|       25|            54|   25|[301.0,294.0,17.0...|[3.17243890379959...|         1|
|404| 380|      41|14234|41.98|       25|            93|   15|[404.0,380.0,41.0...|[4.25802430941873...|         1|
|311| 300|      23|11363|41.02|       22|            72|   16|[311.0,300.0,23.0...|[3.27783554512184...|         1|
|330| 303|      41|10290|39.27|       22|            47|   11|[330.0,303.0,41.0...|[3.47808916363411...|         1|
|299| 289|      32|10405|40.48|       19|            63|   16|[299.0,289.0,32.0...|[3.15135957553514...|         1|
|448| 418|      39|12650|33.37|       19|            77|   28|[448.0,418.0,39.0...|[4.72176953123661...|         1|
|328| 314|      53|11579|44.36|       17|            86|   17|[328.0,314.0,53.0...|[3.45700983536966...|         1|
|287| 279|      11| 9619|35.89|       16|            55|   19|[287.0,279.0,11.0...|[3.02488360594845...|         1|
|288| 273|      40| 9720|41.71|       15|            64|   15|[288.0,273.0,40.0...|[3.03542327008067...|         1|
|304| 278|      40| 8701|36.55|       14|            52|   18|[304.0,278.0,40.0...|[3.20405789619627...|         1|
|344| 318|      40|10889|39.16|       12|            83|   13|[344.0,318.0,40.0...|[3.62564446148525...|         1|
|308| 296|      30| 9284| 34.9|       11|            64|   17|[308.0,296.0,30.0...|[3.24621655272517...|         1|
|350| 297|      84|10773|50.57|       10|            73|   10|[350.0,297.0,84.0...|[3.68888244627860...|         1|
|378| 350|      53|11739|39.52|       10|            83|   20|[378.0,350.0,53.0...|[3.98399304198089...|         1|
|398| 369|      27| 8064|23.57|        6|            39|   30|[398.0,369.0,27.0...|[4.19478632462538...|         1|
+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+----------+
```

Objects in cluster 2 :

Mat	Inns	Not-Outs	Runs	Ave	Centuries	Half Centuries	Ducks	input_features	features	prediction
181	178	14	8113	49.46	27	39	4	[181.0,178.0,14.0...	[1.90767920793264...	2
228	218	39	9577	53.5	25	53	7	[228.0,218.0,39.0...	[2.40304342214720...	2
233	217	39	8581	48.2	21	51	9	[233.0,217.0,39.0...	[2.45574174280832...	2
248	240	16	8094	36.13	21	37	22	[248.0,240.0,16.0...	[2.61383670479169...	2
247	244	19	8824	39.21	20	43	15	[247.0,244.0,19.0...	[2.60329704065947...	2
244	236	20	8500	39.35	18	50	16	[244.0,236.0,20.0...	[2.57167804826279...	2
238	237	28	8648	41.37	17	57	13	[238.0,237.0,28.0...	[2.50844006346944...	2
186	183	19	6927	42.23	16	37	15	[186.0,183.0,19.0...	[1.96037752859377...	2
223	217	14	7090	34.92	16	41	19	[223.0,217.0,14.0...	[2.35034510148608...	2
235	223	17	6951	33.74	15	37	17	[235.0,223.0,17.0...	[2.47682107107277...	2
251	245	9	8273	35.05	15	38	14	[251.0,245.0,9.0,...	[2.64545569718836...	2
219	217	9	7666	36.85	14	51	19	[219.0,217.0,9.0,...	[2.30818644495718...	2
246	228	34	7701	39.69	14	47	16	[246.0,228.0,34.0...	[2.59275737652724...	2
185	185	19	6798	40.95	13	45	11	[185.0,185.0,19.0...	[1.94983786446154...	2
187	167	24	6721	47.0	11	45	7	[187.0,167.0,24.0...	[1.97091719272599...	2
205	203	15	6684	35.55	11	39	15	[205.0,203.0,15.0...	[2.16063114710603...	2
218	216	15	6614	32.9	11	38	19	[218.0,216.0,15.0...	[2.29764678082495...	2
268	251	40	8778	41.6	11	59	6	[268.0,251.0,40.0...	[2.82462998743618...	2
268	259	32	8529	37.57	11	59	13	[268.0,259.0,32.0...	[2.82462998743618...	2
197	194	10	6989	37.98	10	47	8	[197.0,194.0,10.0...	[2.07631383404824...	2

only showing top 20 rows



- **Using PySpark interactive shell to implement KMeans Clustering algorithm**

```
Python 3.8.10 (default, Mar 13 2023, 10:26:41)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/04/23 19:45:44 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.4.0
      /_/

Using Python version 3.8.10 (default, Mar 13 2023 10:26:41)
Spark context Web UI available at http://master:4040
Spark context available as 'sc' (master = local[*], app id = local-1682259346516).
SparkSession available as 'spark'.
```

```
>>> df = spark.read.csv("hdfs://master:9000/kmeans-data/odiBatting.csv", inferSchema=True, header=True)
>>> df.show()
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+
| No|              Player|     Country|Mat|Inns|Not-Outs| Runs|  Ave|Centuries|Half Centuries|Ducks|
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+
|  1|SR Tendulkar (INDIA)|       India|463| 452|      41|18426|44.83|       49|            96|   20|
|  2|     V Kohli (INDIA)|       India|254| 245|      39|12169|59.07|       43|            62|   13|
|  3|RT Ponting (AUS/ICC)|   Australia|375| 365|      39|13704|42.03|       30|            82|   20|
|  4|   RG Sharma (INDIA)|       India|227| 220|      32| 9205|48.96|       29|            43|   13|
|  5|ST Jayasuriya (As...|   Sri Lanka|445| 433|      18|13430|32.36|       28|            68|   34|
|  6|        HM Amla (SA)|South Africa |181| 178|      14| 8113|49.46|       27|            39|    4|
|  7|AB de Villiers (A...|South Africa |228| 218|      39| 9577| 53.5|       25|            53|    7|
|  8|   CH Gayle (ICC/WI)| West Indies|301| 294|      17|10480|37.83|       25|            54|   25|
|  9|KC Sangakkara (As...|   Sri Lanka|404| 380|      41|14234|41.98|       25|            93|   15|
| 10|SC Ganguly (Asia/...|       India|311| 300|      23|11363|41.02|       22|            72|   16|
| 11|     TM Dilshan (SL)|   Sri Lanka|330| 303|      41|10290|39.27|       22|            47|   11|
| 12|    LRPL Taylor (NZ)| New Zealand|233| 217|      39| 8581| 48.2|       21|            51|    9|
| 13|       HH Gibbs (SA)|South Africa |248| 240|      16| 8094|36.13|       21|            37|   22|
| 14|   Saeed Anwar (PAK)|    Pakistan|247| 244|      19| 8824|39.21|       20|            43|   15|
| 15|    BC Lara (ICC/WI)| West Indies|299| 289|      32|10405|40.48|       19|            63|   16|
| 16|DPMD Jayawardene ...|   Sri Lanka|448| 418|      39|12650|33.37|       19|            77|   28|
| 17|     DA Warner (AUS)|   Australia|128| 126|       6| 5455|45.45|       18|            23|    2|
| 18|      ME Waugh (AUS)|   Australia|244| 236|      20| 8500|39.35|       18|            50|   16|
| 19|      AJ Finch (AUS)|   Australia|132| 128|       3| 5232|41.85|       17|            29|   11|
| 20|    S Dhawan (INDIA)|       India|145| 142|       8| 6105|45.55|       17|            33|    5|
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+
only showing top 20 rows
```

```
>>> df.count()
119
>>> len(df.columns)
11
>>> df.printSchema()
root
 |-- No: integer (nullable = true)
 |-- Player: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Mat: integer (nullable = true)
 |-- Inns: integer (nullable = true)
 |-- Not-Outs: integer (nullable = true)
 |-- Runs: integer (nullable = true)
 |-- Ave: double (nullable = true)
 |-- Centuries: integer (nullable = true)
 |-- Half Centuries: integer (nullable = true)
 |-- Ducks: integer (nullable = true)
```

```
>>> from pyspark.ml.linalg import Vector
>>> from pyspark.ml.feature import VectorAssembler, StandardScaler
>>> df.columns
['No', 'Player', 'Country', 'Mat', 'Inns', 'Not-Outs', 'Runs', 'Ave', 'Centuries', 'Half Centuries', 'Ducks']
>>> selected_cols = ["Mat", "Inns", "Not-Outs", "Runs", "Ave", "Centuries", "Half Centuries", "Ducks"]
>>> assembler = VectorAssembler(inputCols=selected_cols, outputCol="input_features")
```

```
>>> data = assembler.transform(df)
>>> scaler = StandardScaler(inputCol="input_features", outputCol="features")
>>> scalerModel = scaler.fit(data)
>>> data = scalerModel.transform(data)
>>> data.show()
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+
| No|              Player|     Country|Mat|Inns|Not-Outs| Runs|  Ave|Centuries|Half Centuries|Ducks|      input_features|            features|
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+
|  1|SR Tendulkar (INDIA)|       India|463| 452|      41|18426|44.83|       49|            96|   20|[463.0,452.0,41.0...|[4.87986449321997...|
|  2|     V Kohli (INDIA)|       India|254| 245|      39|12169|59.07|       43|            62|   13|[254.0,245.0,39.0...|[2.67707468958504...|
|  3|RT Ponting (AUS/ICC)|   Australia|375| 365|      39|13704|42.03|       30|            82|   20|[375.0,365.0,39.0...|[3.95237404958421...|
|  4|    RG Sharma (INDIA)|       India|227| 220|      32| 9205|48.96|       29|            43|   13|[227.0,220.0,32.0...|[2.39250375801497...|
|  5|ST Jayasuriya (As...|   Sri Lanka|445| 433|      18|13430|32.36|       28|            68|   34|[445.0,433.0,18.0...|[4.69015053883993...|
|  6|        HM Amla (SA)|South Africa |181| 178|      14| 8113|49.46|       27|            39|    4|[181.0,178.0,14.0...|[1.90767920793264...|
|  7|AB de Villiers (A...|South Africa |228| 218|      39| 9577| 53.5|       25|            53|    7|[228.0,218.0,39.0...|[2.40304342214720...|
|  8|   CH Gayle (ICC/WI)| West Indies|301| 294|      17|10480|37.83|       25|            54|   25|[301.0,294.0,17.0...|[3.17243890379959...|
|  9|KC Sangakkara (As...|   Sri Lanka|404| 380|      41|14234|41.98|       25|            93|   15|[404.0,380.0,41.0...|[4.25802430941873...|
| 10|SC Ganguly (Asia/...|       India|311| 300|      23|11363|41.02|       22|            72|   16|[311.0,300.0,23.0...|[3.27783554512104...|
| 11|      TM Dilshan (SL)|   Sri Lanka|330| 303|      41|10290|39.27|       22|            47|   11|[330.0,303.0,41.0...|[3.47808916363411...|
| 12|    LRPL Taylor (NZ)| New Zealand|233| 217|      39| 8581| 48.2|       21|            51|    9|[233.0,217.0,39.0...|[2.45574174280832...|
| 13|       HH Gibbs (SA)|South Africa |248| 240|      16| 8094|36.13|       21|            37|   22|[248.0,240.0,16.0...|[2.61383670479169...|
| 14|   Saeed Anwar (PAK)|    Pakistan|247| 244|      19| 8824|39.21|       20|            43|   15|[247.0,244.0,19.0...|[2.60329704065947...|
| 15|    BC Lara (ICC/WI)| West Indies|299| 289|      32|10405|40.48|       19|            63|   16|[299.0,289.0,32.0...|[3.15135957553514...|
| 16|DPMD Jayawardene ...|   Sri Lanka|448| 418|      39|12650|33.37|       19|            77|   28|[448.0,418.0,39.0...|[4.72176953123661...|
| 17|      DA Warner (AUS)|   Australia|128| 126|       6| 5455|45.45|       18|            23|    2|[128.0,126.0,6.0,...|[1.34907780892474...|
| 18|       ME Waugh (AUS)|   Australia|244| 236|      20| 8500|39.35|       18|            50|   16|[244.0,236.0,20.0...|[2.57167804826279...|
| 19|       AJ Finch (AUS)|   Australia|132| 128|       3| 5232|41.85|       17|            29|   11|[132.0,128.0,3.0,...|[1.39123566545364...|
| 20|    S Dhawan (INDIA)|       India|145| 142|       8| 6105|45.55|       17|            33|    5|[145.0,142.0,8.0,...|[1.52825129917256...|
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+
only showing top 20 rows
```

```
>>> from pyspark.ml.clustering import KMeans
>>> kmeans = KMeans().setK(3)
>>> model = kmeans.fit(data)
23/04/23 19:56:10 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
>>> model
KMeansModel: uid=KMeans_6f07f30e5128, k=3, distanceMeasure=euclidean, numFeatures=8
>>> model.transform(data).groupBy("prediction").count().show()
+----------+-----+
|prediction|count|
+----------+-----+
|         1|   14|
|         2|   52|
|         0|   53|
+----------+-----+
```

```
>>> predictions = model.transform(data)
>>> predictions.show()
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+----------+
| No|              Player|     Country|Mat|Inns|Not-Outs| Runs|  Ave|Centuries|Half Centuries|Ducks|      input_features|            features|prediction|
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+----------+
|  1|SR Tendulkar (INDIA)|       India|463| 452|      41|18426|44.83|       49|            96|   20|[463.0,452.0,41.0...|[4.87986449321997...|         1|
|  2|     V Kohli (INDIA)|       India|254| 245|      39|12169|59.07|       43|            62|   13|[254.0,245.0,39.0...|[2.67707468958504...|         1|
|  3|RT Ponting (AUS/ICC)|   Australia|375| 365|      39|13704|42.03|       30|            82|   20|[375.0,365.0,39.0...|[3.95237404958421...|         1|
|  4|    RG Sharma (INDIA)|       India|227| 220|      32| 9205|48.96|       29|            43|   13|[227.0,220.0,32.0...|[2.39250375801497...|         0|
|  5|ST Jayasuriya (As...|   Sri Lanka|445| 433|      18|13430|32.36|       28|            68|   34|[445.0,433.0,18.0...|[4.69015053883993...|         1|
|  6|        HM Amla (SA)|South Africa |181| 178|      14| 8113|49.46|       27|            39|    4|[181.0,178.0,14.0...|[1.90767920793264...|         0|
|  7|AB de Villiers (A...|South Africa |228| 218|      39| 9577| 53.5|       25|            53|    7|[228.0,218.0,39.0...|[2.40304342214720...|         0|
|  8|   CH Gayle (ICC/WI)| West Indies|301| 294|      17|10480|37.83|       25|            54|   25|[301.0,294.0,17.0...|[3.17243890379959...|         1|
|  9|KC Sangakkara (As...|   Sri Lanka|404| 380|      41|14234|41.98|       25|            93|   15|[404.0,380.0,41.0...|[4.25802430941873...|         1|
| 10|SC Ganguly (Asia/...|       India|311| 300|      23|11363|41.02|       22|            72|   16|[311.0,300.0,23.0...|[3.27783554512184...|         1|
| 11|      TM Dilshan (SL)|   Sri Lanka|330| 303|      41|10290|39.27|       22|            47|   11|[330.0,303.0,41.0...|[3.47808916363411...|         1|
| 12|    LRPL Taylor (NZ)| New Zealand|233| 217|      39| 8581| 48.2|       21|            51|    9|[233.0,217.0,39.0...|[2.45574174280832...|         0|
| 13|       HH Gibbs (SA)|South Africa |248| 240|      16| 8094|36.13|       21|            37|   22|[248.0,240.0,16.0...|[2.61383670479169...|         0|
| 14|   Saeed Anwar (PAK)|    Pakistan|247| 244|      19| 8824|39.21|       20|            43|   15|[247.0,244.0,19.0...|[2.60329704065947...|         0|
| 15|    BC Lara (ICC/WI)| West Indies|299| 289|      32|10405|40.48|       19|            63|   16|[299.0,289.0,32.0...|[3.15135957553514...|         1|
| 16|DPMD Jayawardene ...|   Sri Lanka|448| 418|      39|12650|33.37|       19|            77|   28|[448.0,418.0,39.0...|[4.72176953123661...|         1|
| 17|      DA Warner (AUS)|   Australia|128| 126|       6| 5455|45.45|       18|            23|    2|[128.0,126.0,6.0,...|[1.34907780892474...|         2|
| 18|       ME Waugh (AUS)|   Australia|244| 236|      20| 8500|39.35|       18|            50|   16|[244.0,236.0,20.0...|[2.57167804826279...|         0|
| 19|       AJ Finch (AUS)|   Australia|132| 128|       3| 5232|41.85|       17|            29|   11|[132.0,128.0,3.0,...|[1.39123566545364...|         2|
| 20|    S Dhawan (INDIA)|       India|145| 142|       8| 6105|45.55|       17|            33|    5|[145.0,142.0,8.0,...|[1.52825129917256...|         2|
+---+--------------------+------------+---+----+--------+-----+-----+---------+--------------+-----+--------------------+--------------------+----------+
only showing top 20 rows
```

```
>>> model.summary
<pyspark.ml.clustering.KMeansSummary object at 0x7fee12bcec70>
>>> model.save("hdfs://master:9000/models/kmeans-py")
>>> model.clusterCenters()
[array([2.47861083, 2.48776779, 1.77609595, 2.33847943, 5.41431921,
       1.48150806, 2.1711234 , 2.1017532 ]), array([3.78675076, 3.81367997, 2.52875985, 3.9774585 , 5.96651389,
       3.12005953, 3.7097323 , 2.91592667]), array([1.14963413, 1.17328585, 0.56167647, 1.21479937, 5.714106  ,
       1.11917543, 1.0218449 , 1.02544291])]
>>> model.summary.clusterSizes
[53, 14, 52]
>>> exit()
```

CONCLUSION :

In this experiment, I performed KMeans clustering using the Spark MLlib library and learned how to utilize the Spark MLib library to execute machine learning tasks.