WALMART SALES DATA PREDICTION USING APACHE SPARK

BY:

S.SAI SIDDARDHA

2015BCS0032

TABLE OF CONTENTS

Objective

- ❖ The Main Objectives of this Project is :
- ❏ Understanding and Targeting Customers
- ❏ Taking Strategic Decisions
- ❏ Cost Optimization
- ❏ Improving Customer Experiences

Abstract

- ➔ We all know Information technology in this 21st century is reaching the skies with large-scale of data to be processed and studied to make sense of data where the traditional approach is no more effective. Now, retailers need a 360-degree view of their consumers, without which, they can miss competitive edge of the market.
- ➔ Retailers have to create effective promotions and offers to meet its sales and marketing goals, otherwise they will forgo the major opportunities that the current market offers.
- ➔ Big Data application enables these retail organizations to use prior year's data to better forecast and predict the coming year's sales.

Contd..

- ➔ In this project,I am analyzing the data sets of world’s largest retailers, Walmart Store to determine the business drivers and predict which departments are affected by the different scenarios (such as temperature, fuel price and holidays) and their impact on sales at stores’ of different locations.
- ➔ I have made use of Scala and Python API of the Spark framework to gain new insights into the consumer behaviors and comprehend Walmart’s marketing efforts and their data-driven strategies through visual representation of the analyzed data.

Spark introduction

- Apache Spark is a fast and general-purpose cluster computing system. Designed for large-scale data processing, it run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Spark can run on Hadoop,Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.
- There are multiple open source stream processing platforms such as Apache Kafka, Apache Flink, Apache Storm, Apache Samra, etc.

Literature Survey

- ➢ In 2015, Harsoor & Patil worked on forecasting Sales of Walmart Store using big data applications: Hadoop, MapReduce and Hive so that resources are managed efficiently. This paper used the same sales data set that we utilized for analysis, however they forecasted the sales for upcoming 39 weeks.
- ➢ In 2013, Katal, Wazid, & Goudar performed thorough studies about handling a Big Data; their issues, challenges, various tools and good practices. Technical challenges like scalability, fault tolerance, data quality and heterogeneous data processing was also mentioned. They have proposed Parallel Programming Model like Distributed file system,MapReduce and Spark as a good tool for Big Data .

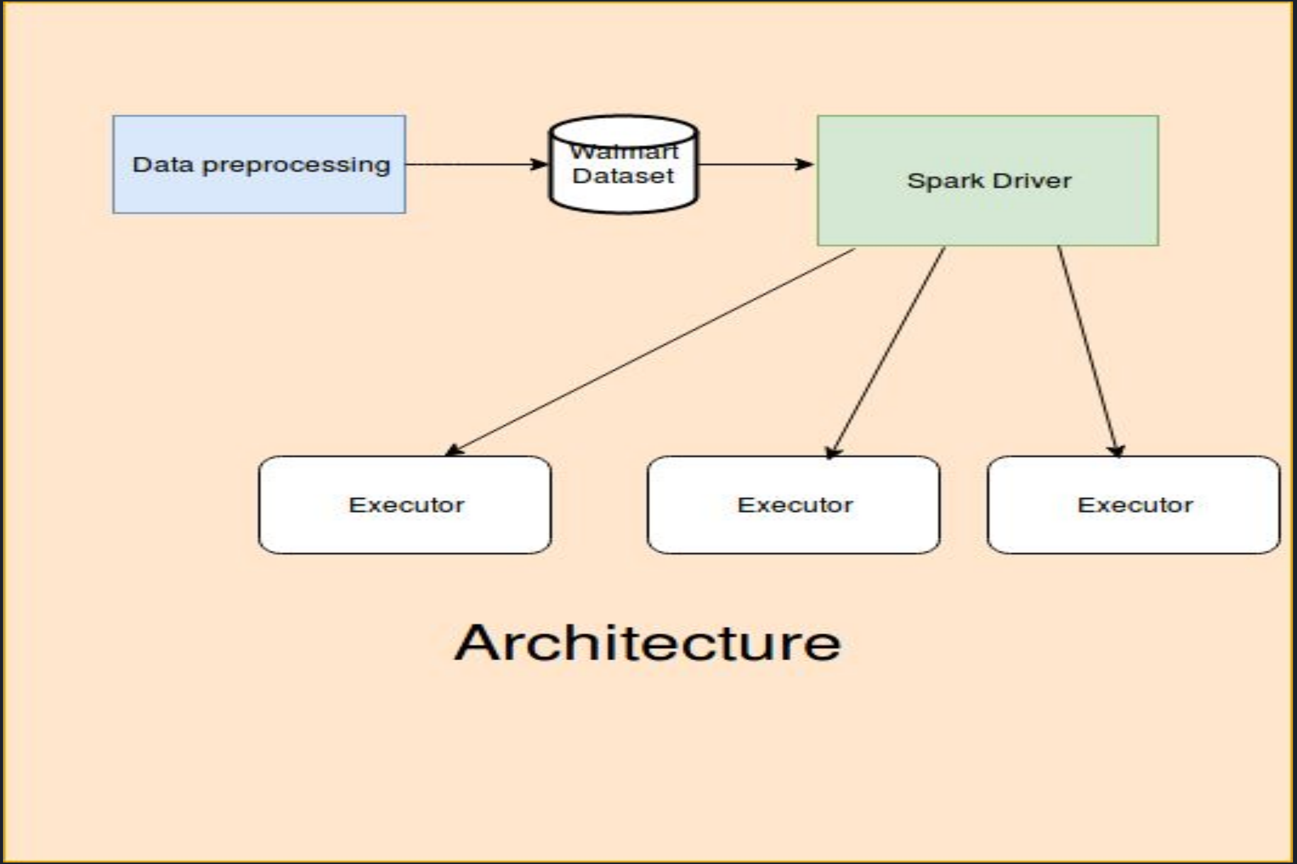
contd..

- ❑ In 2015, Riyaz& Surekha worked on MapReduce on Hadoop to build a data analytical engine for weather, temperature analysis for National Climate Data Centre. This paper had all the details and results about MapReduce program execution. Their findings concluded that MapReduce with Hadoop is good for weather data analysis.
- ❑ In 2017, Chouksey & Chauhan performed weather forecast using MapReduce and Spark in order to formulate earlier weather warnings so that people and businesses are prepared for undesirable weather condition. Weather has greater influence in agriculture sector, sporting, tourism and government planning

Contd...

- ❖ In 2013, Zaslavsky, Perera, & Georgakopoulos recommended the use of Hadoop, Apache Spark and NoSQL Technology to process billions of sensing devices data. They explained Sensing as a service and big data.
- ❖ In 2017, Inoublia, Aridhib, Meznic, & Jungd worked on experimental evaluation and a comparative study of Healthcare scientific applications which decided health status using interconnected sensors over the human body. This included breath, insulin, cardiovascular, glucose, blood and body temperature.

Proposed Architecture

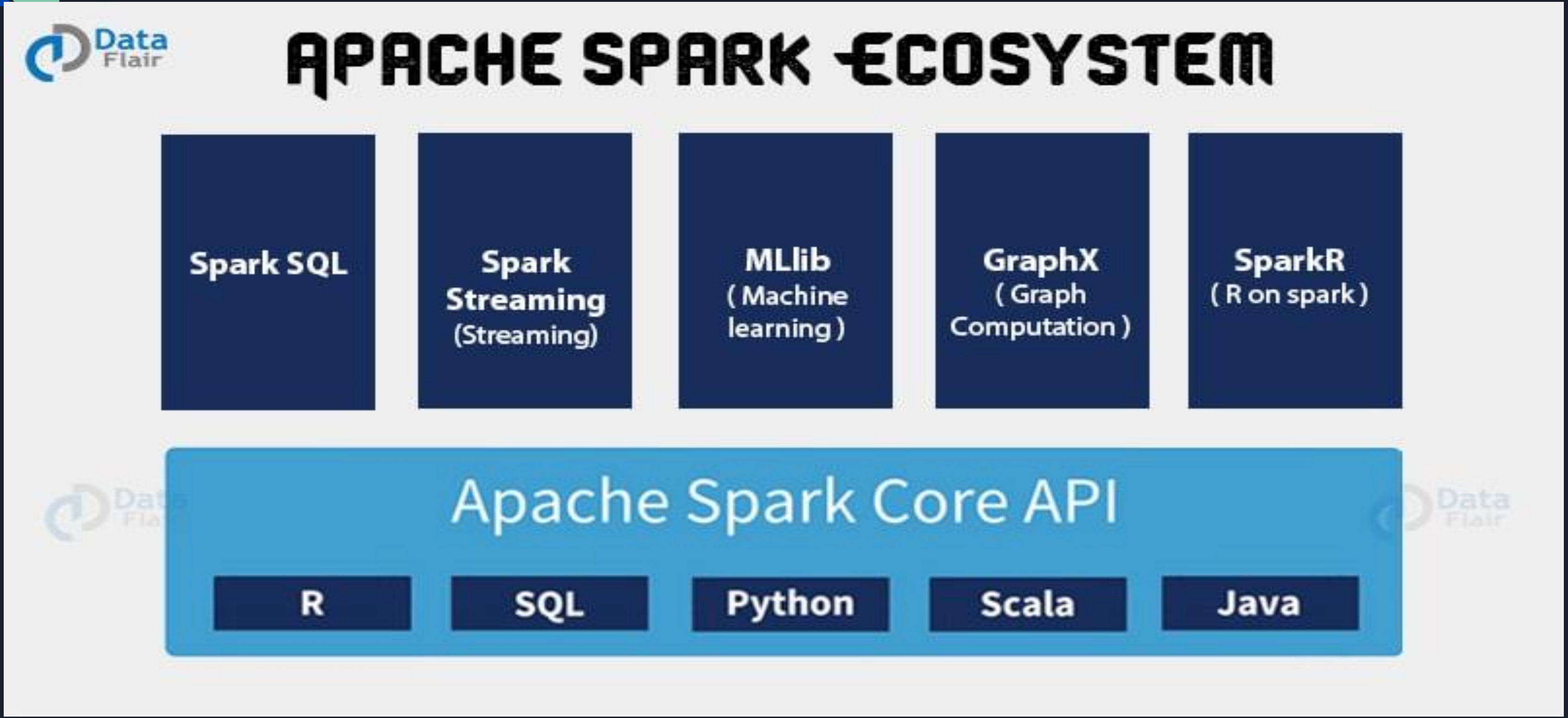


Apache Spark Features

Some of the features of apache spark are:

- ➢ Speed
- ➢ Polyglot
- ➢ Advanced Analytics
- ➢ In memory computation
- ➢ Hadoop Integration
- ➢ Machine learning

Spark Ecosystem



Spark Components

1. Spark SQL

   Spark SQL is a component on top of Spark Core that introduces a new data abstraction called Schema RDD, which provides support for structured and semi-structured data.

❖ Spark Streaming

   Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

- MLlib (Machine Learning Library)

  MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

- GraphX

  GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Resilient Distributed Datasets

- ➔ Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.
- ➔ Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

Spark Running Modes

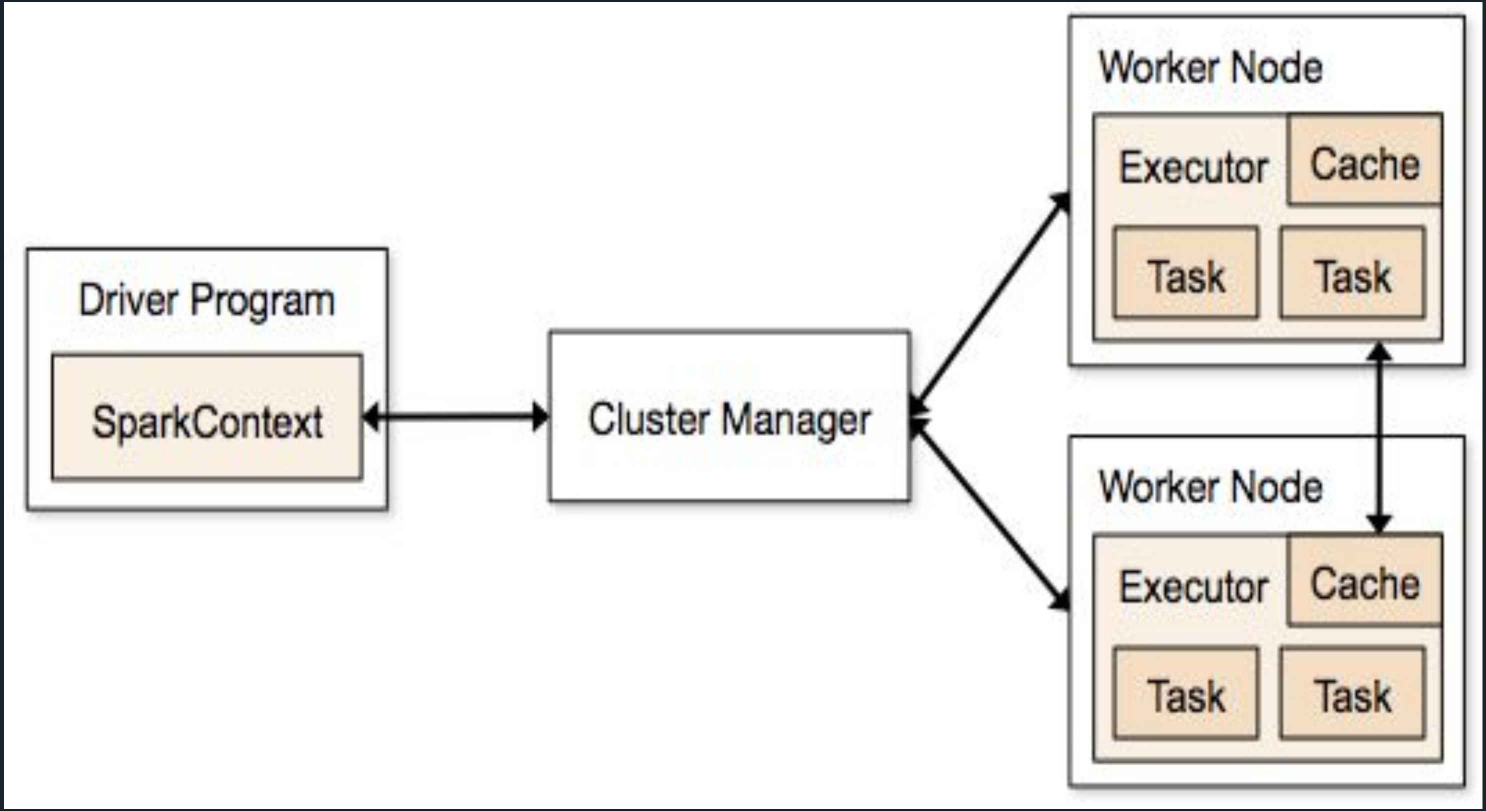
- When for execution, we submit a spark job to local or on a cluster, the behaviour of spark job totally depends on one parameter, that is the “Driver” component. Where “Driver” component of spark job will reside, it defines the behaviour of spark job.
- Basically, there are two types of “Deploy modes” in spark, such as “Client mode” and “Cluster mode”.

Client Mode

- ★ The behaviour of spark job depends on the “driver” component. So here,”driver” component of spark job will run on the machine from which job is submitted. Hence, this spark mode is basically “client mode”.
- ★ When job submitting machine is within or near to “spark infrastructure”. Since there is no high network latency of data movement for final result generation between “spark infrastructure” and “driver”, then, this mode works very fine.

Cluster Mode

- A spark cluster has a single Master and any number of Slaves/Workers. The driver and the executors run their individual Java processes and users can run them on the same horizontal spark cluster or on separate machines i.e. in a vertical spark cluster or in mixed machine configuration.
- Similarly, here “driver” component of spark job will not run on the local machine from which job is submitted. Hence, this spark mode is basically “cluster mode”. In addition, here spark job will launch “driver” component inside the cluster.


Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task
Worker Node
Executor
Cache
Task
Task

Spark Cluster Setup

- ➢ Apache Spark follows a master/slave architecture with two main daemons and a cluster manager –
  - Master Daemon — (Master/Driver Process)
  - Worker Daemon –(Slave Process)
  - Cluster Manager
- ➢ Setup:

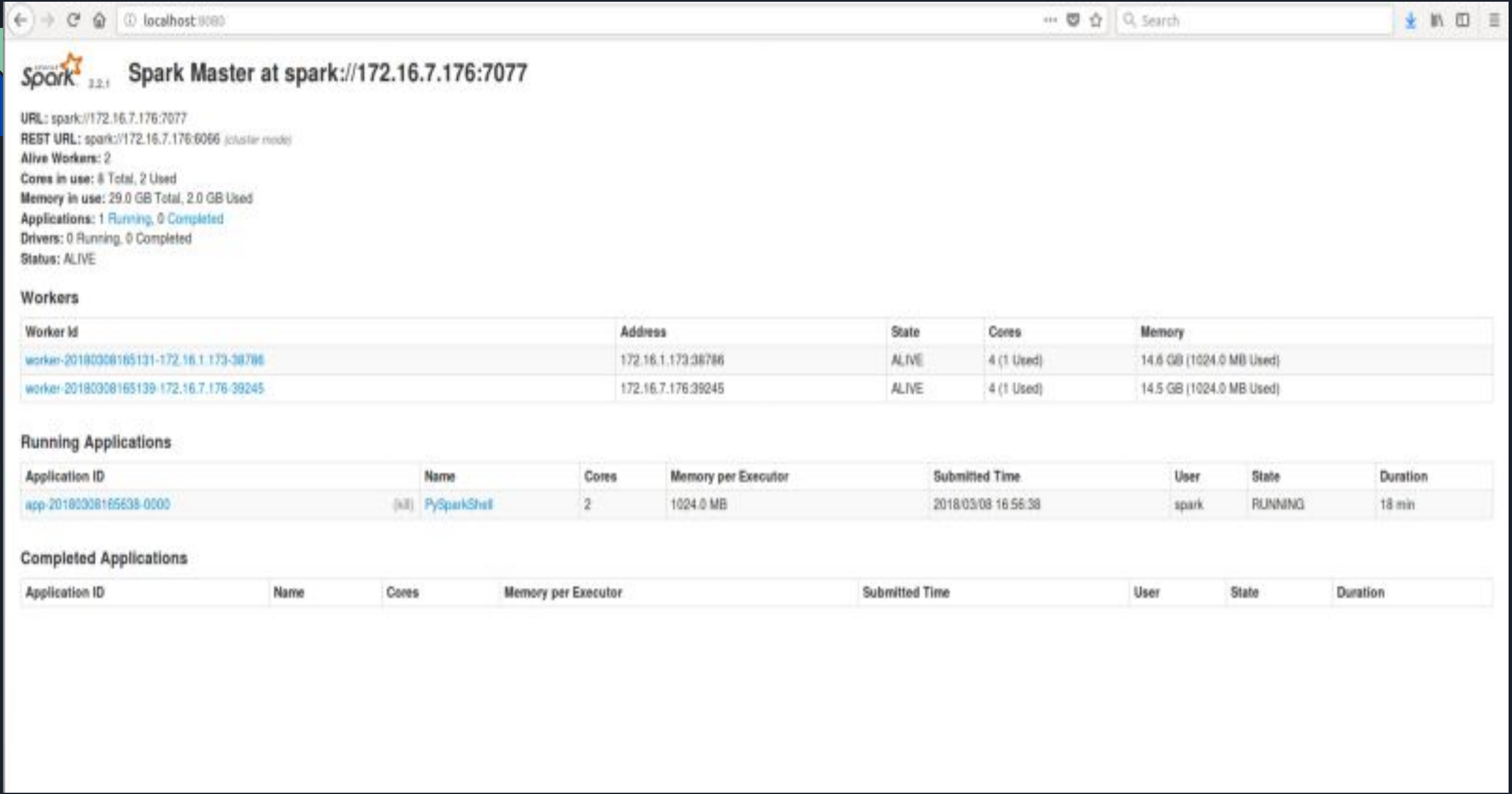
1) **Add entries in hosts file (master and slaves)**

2) **Install Java 7 (master and slaves)**

3) **Install Scala (master and slaves)**

4) **Configure SSH (only master)**

5) **Install Spark**



Spark Master at spark://172.16.7.176:7077

URL: spark://172.16.7.176:7077
REST URL: spark://172.16.7.176:6066 *(cluster mode)*
Alive Workers: 2
Cores in use: 8 Total, 2 Used
Memory in use: 29.0 GB Total, 2.0 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20180308165131-172.16.1.173-38786	172.16.1.173:38786	ALIVE	4 (1 Used)	14.6 GB (1024.0 MB Used)
worker-20180308165139-172.16.7.176-39245	172.16.7.176:39245	ALIVE	4 (1 Used)	14.5 GB (1024.0 MB Used)

Running Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20180308165638-0000 (kill)	PySparkShell	2	1024.0 MB	2018/03/08 16:56:38	spark	RUNNING	18 min

Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

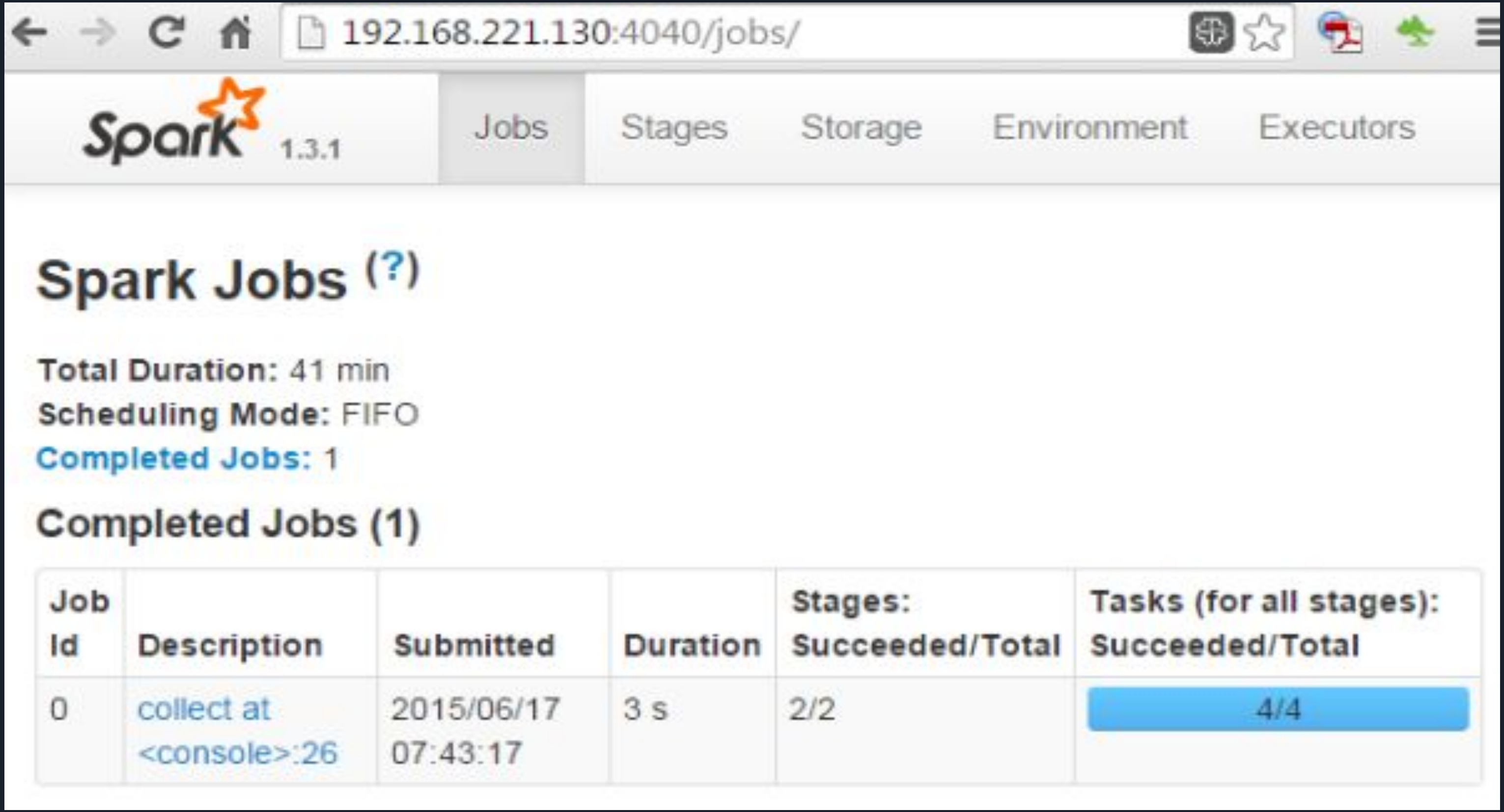
Spark Cluster

- ❖ Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the *driver program*).
- ❖ Specifically, to run on a cluster, the SparkContext can connect to several types of *cluster managers* (either Spark’s own standalone cluster manager or Mesos/YARN), which allocate resources across applications.
- ❖ Once connected, Spark acquires *executors* on nodes in the cluster, which are processes that run computations and store data for your application.
- ❖ Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends *tasks* for the executors to run.

Useful Things About Spark Cluster

- Each application gets its own executor processes, which stay up for the duration of the whole application and run tasks in multiple threads. This has the benefit of isolating applications from each other, on both the scheduling side (each driver schedules its own tasks) and executor side (tasks from different applications run in different JVMs).
- Spark is agnostic to the underlying cluster manager. As long as it can acquire executor processes, and these communicate with each other, it is relatively easy to run it even on a cluster manager that also supports other applications (e.g. Mesos/YARN).
- Because the driver schedules tasks on the cluster, it should be run close to the worker nodes, preferably on the same local area network. If you’d like to send requests to the cluster remotely, it’s better to open an RPC to the driver and have it submit operations from nearby than to run a driver far away from the worker nodes.

Spark Job Running in Spark Cluster



Spark Jobs (?)

Total Duration: 41 min
Scheduling Mode: FIFO
Completed Jobs: 1

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <console>:26	2015/06/17 07:43:17	3 s	2/2	4/4

Spark Mlib

- ❑ Apache Spark offers a Machine Learning API called **MLlib**. PySpark has this machine learning API in Python as well. It supports different kind of algorithms, which are mentioned below

1. **Mllib.classification**
2. **Mllib.clustering**
3. **Mllib.regression**
4. **Mllib.recommendation**
5. **mllib.linalg**

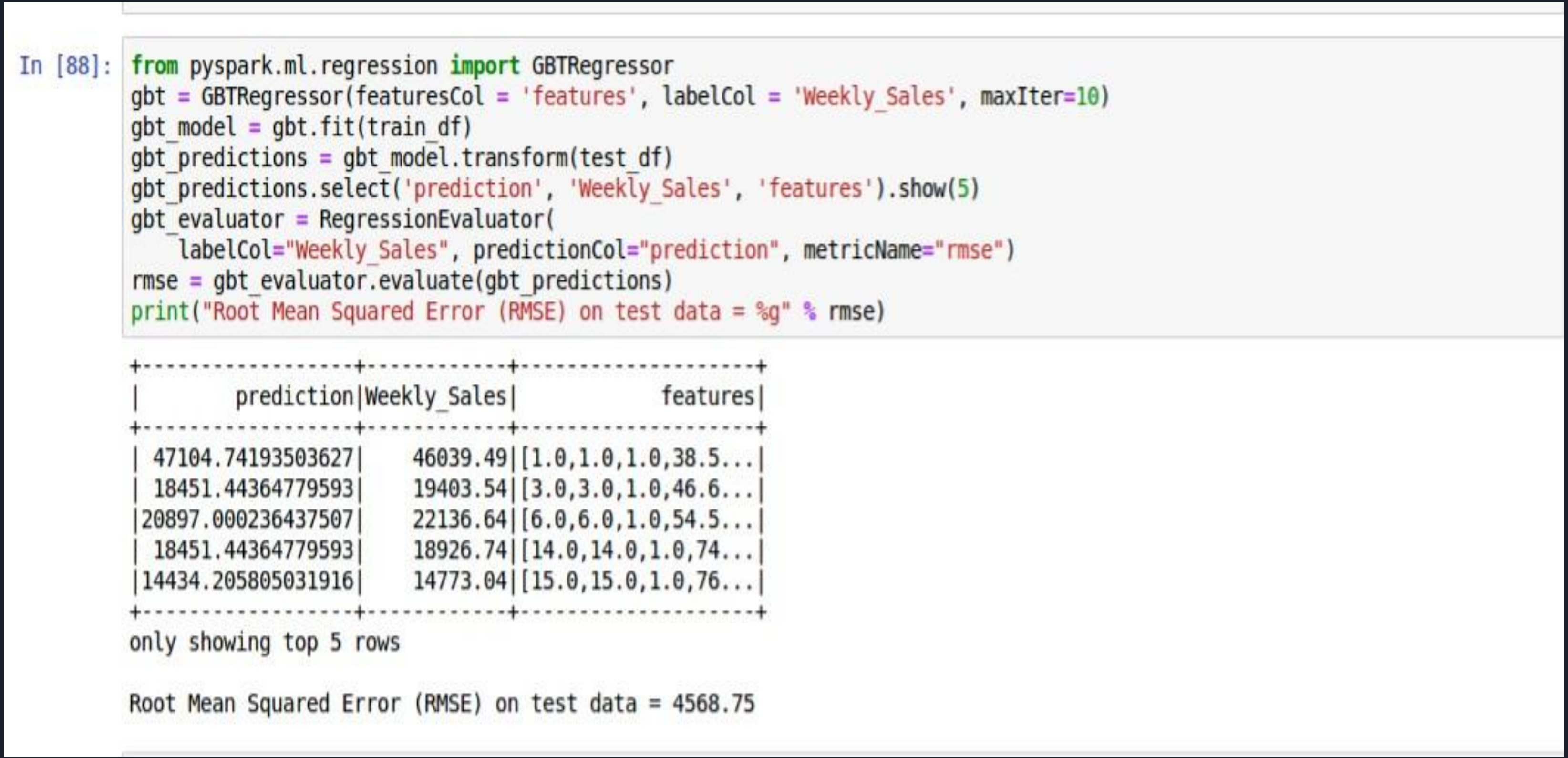
Implementation

- ➔ I have used Four types of regression in order to calculate prediction.
- ➔ Simple Linear regression
- ➔ Gradient Boost Regression
- ➔ Logistic Regression
- ➔ Decision Tree Regression

Implementation

- ❏ The tools and techniques used for this work includes the collection of Huge Walmart sales datasets stored in CSV format. We used Apache Spark with a build version of Hadoop leveraging HDFS as a data storage option.
- ❏ After we configured our environment, our first task was to load the files as spark dataframes. Dataframe is a distributed collection of data organized into named columns which is equivalent to tables in RDBMS . The spark dataframe API was designed to make big data processing simple for a wider audience and also it supports distributed data processing in general purpose programing languages like Scala, Python and Java.

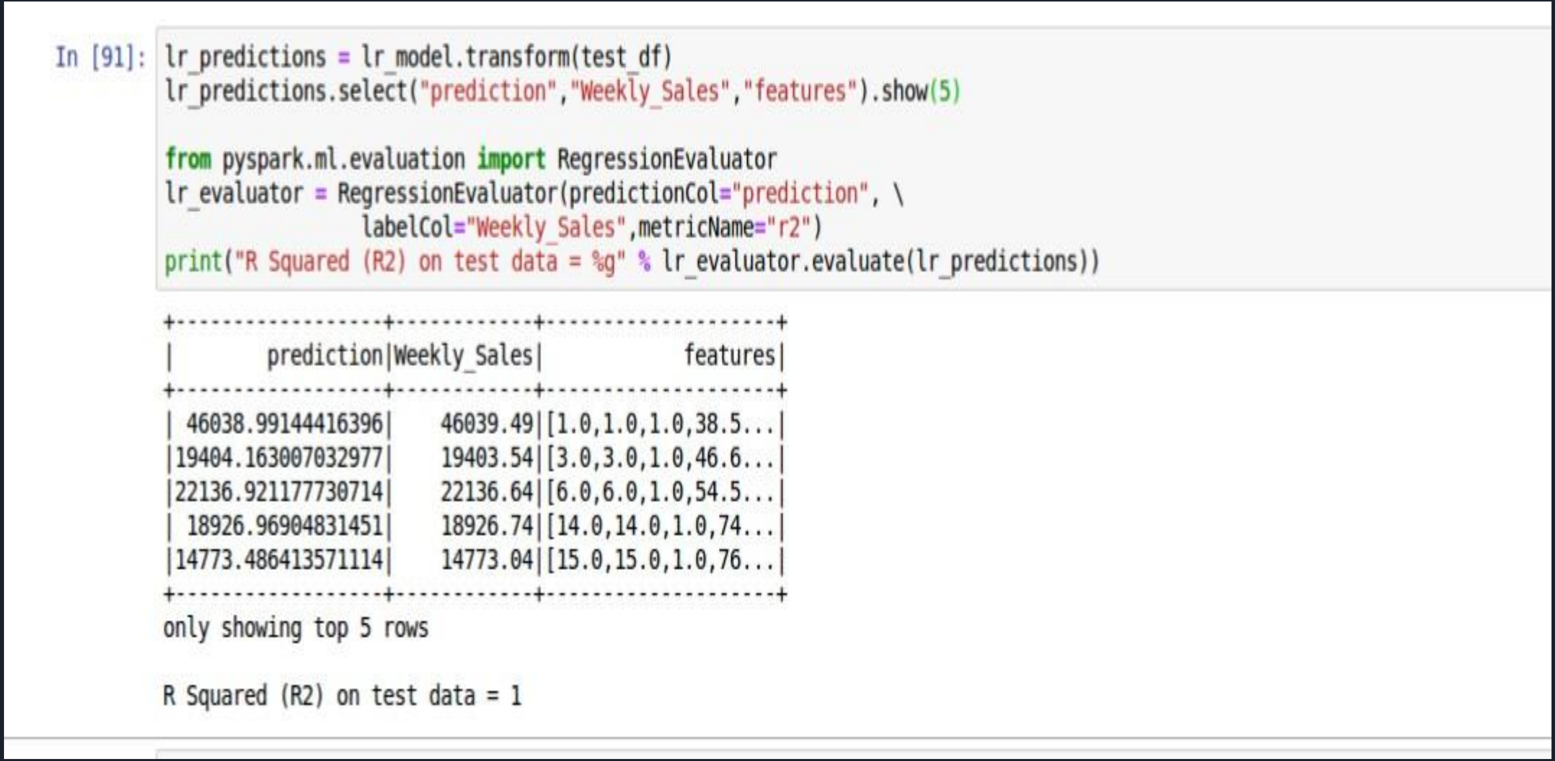
Final Output(Gradient Boosting Regression)



In [88]:
```
from pyspark.ml.regression import GBTRegressor
gbt = GBTRegressor(featuresCol = 'features', labelCol = 'Weekly_Sales', maxIter=10)
gbt_model = gbt.fit(train_df)
gbt_predictions = gbt_model.transform(test_df)
gbt_predictions.select('prediction', 'Weekly_Sales', 'features').show(5)
gbt_evaluator = RegressionEvaluator(
    labelCol="Weekly_Sales", predictionCol="prediction", metricName="rmse")
rmse = gbt_evaluator.evaluate(gbt_predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
+------------------+------------+--------------------+
|        prediction|Weekly_Sales|            features|
+------------------+------------+--------------------+
| 47104.74193503627|    46039.49|[1.0,1.0,1.0,38.5...|
| 18451.44364779593|    19403.54|[3.0,3.0,1.0,46.6...|
|20897.000236437507|    22136.64|[6.0,6.0,1.0,54.5...|
| 18451.44364779593|    18926.74|[14.0,14.0,1.0,74...|
|14434.205805031916|    14773.04|[15.0,15.0,1.0,76...|
+------------------+------------+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 4568.75
```

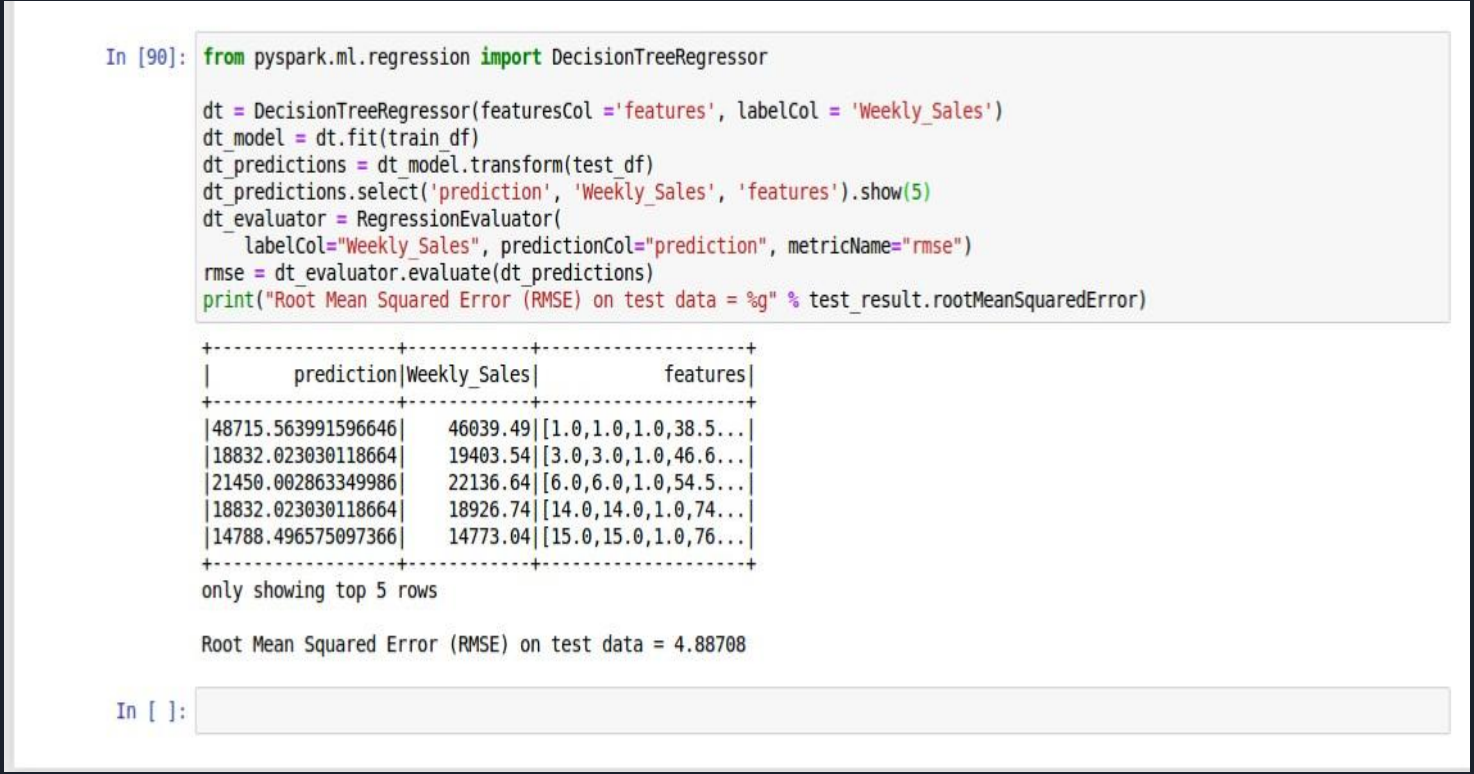
Simple Linear Regression



```
In [91]: lr_predictions = lr_model.transform(test_df)
         lr_predictions.select("prediction","Weekly_Sales","features").show(5)

         from pyspark.ml.evaluation import RegressionEvaluator
         lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                          labelCol="Weekly_Sales",metricName="r2")
         print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))
```

```
+-----------------+------------+--------------------+
|       prediction|Weekly_Sales|            features|
+-----------------+------------+--------------------+
| 46038.99144416396|    46039.49|[1.0,1.0,1.0,38.5...|
|19404.163007032977|    19403.54|[3.0,3.0,1.0,46.6...|
|22136.921177730714|    22136.64|[6.0,6.0,1.0,54.5...|
| 18926.96904831451|    18926.74|[14.0,14.0,1.0,74...|
|14773.486413571114|    14773.04|[15.0,15.0,1.0,76...|
+-----------------+------------+--------------------+
only showing top 5 rows

R Squared (R2) on test data = 1
```

Decision Tree Regression

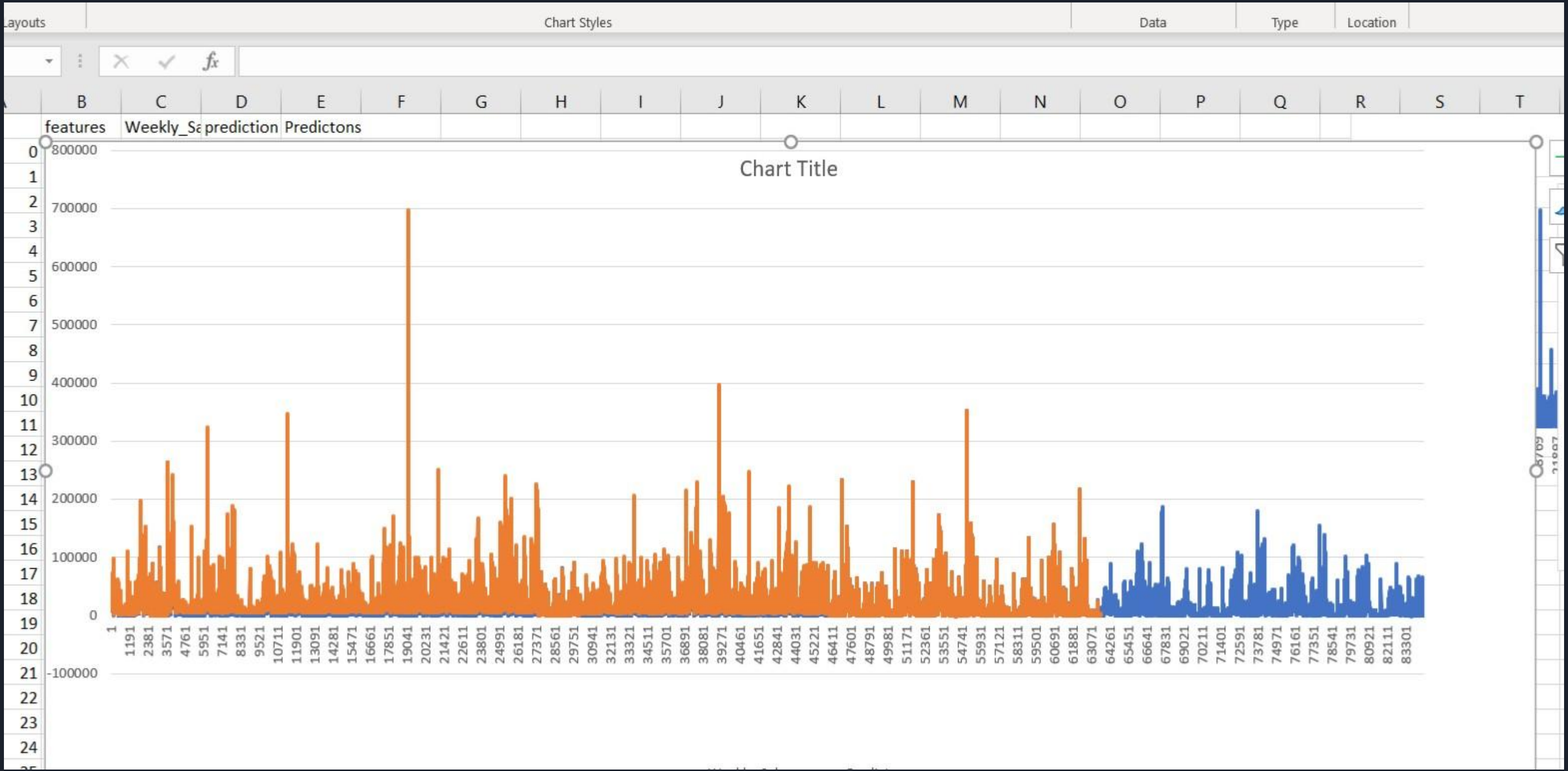


```
In [90]: from pyspark.ml.regression import DecisionTreeRegressor

         dt = DecisionTreeRegressor(featuresCol ='features', labelCol = 'Weekly_Sales')
         dt_model = dt.fit(train_df)
         dt_predictions = dt_model.transform(test_df)
         dt_predictions.select('prediction', 'Weekly_Sales', 'features').show(5)
         dt_evaluator = RegressionEvaluator(
             labelCol="Weekly_Sales", predictionCol="prediction", metricName="rmse")
         rmse = dt_evaluator.evaluate(dt_predictions)
         print("Root Mean Squared Error (RMSE) on test data = %g" % test_result.rootMeanSquaredError)
```

```
+-----------------+------------+--------------------+
|       prediction|Weekly_Sales|            features|
+-----------------+------------+--------------------+
|48715.563991596646|    46039.49|[1.0,1.0,1.0,38.5...|
|18832.023030118664|    19403.54|[3.0,3.0,1.0,46.6...|
|21450.002863349986|    22136.64|[6.0,6.0,1.0,54.5...|
|18832.023030118664|    18926.74|[14.0,14.0,1.0,74...|
|14788.496575097366|    14773.04|[15.0,15.0,1.0,76...|
+-----------------+------------+--------------------+
only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 4.88708
```

```
In [ ]:
```

Graphical Representation(prediction vs Weekly Sales)



Conclusion

- In conclusion, WalMart is the number one retailer in the USA and it also operates in many other countries all around the world and is moving into new countries as years pass by. There, are other companies who are constantly rising as well and would give Walmart a tough competition in the future if Walmart does not stay to the top of their game.
- In order to do so, they will need to understand their business trends, the customer needs and manage the resources wisely.
- In this era when the technologies are reaching out to new levels, Big Data is taking over the traditional method of managing and analyzing data. These technologies are constantly used to understand complex datasets in a matter of time with beautiful visual representations.
- Through observing the history of the company’s datasets, clearer ideas on the sales for the previous years was realized which will be very helpful to the company on its own.

Contd..

- ➢ Additionally, seasonality trend and randomness and future forecasts will help to analyse sale drops which the companies can avoid by using a more focused and efficient tactics to minimize the sale drop and maximize the profit and remain in competition.

References

- ➔ A. S. Harsoor and A. Patil, “Forecast of sales of walmart store using Big Data application,” International Journal of Research in Engineering and Technology, vol. 4, p. 6, June 2015.
- ➔ M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. J.Franklin, S. Shenker, and I. Stoica, “Fast and interactive analytics over Hadoop data with Spark,” Usenix - The Advanced Computing Systems Association, 2012.
- ➔ J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters. Association for Computing Machinery, 2008.

Contd..

- ➢ A. Katal, M. Wazid, and R. H. Goudar, Big Data: Issues,Challenges, Tools and Good Practices, 2013.
- ➢ P. Chouksey and A. S. Chauhan, A Review of Weather Data Analytics using Big Data. International Journal of Advanced Research in Computer and Communication Engineering, 2017.
- ➢ T. A.S. Foundation, “Lightning-fast cluster computing,”[Online].Available: [Accessed Sept, vol. 2017. [Online].Available:{Spark}.apache.org/
- ➢ M. Sharma, V. Chauhan, and K. Kishore, “A review: MapReduce and Spark for Big Data analysis,” in 5th International Conference on Recent Innovations in Science. 5: Engineering and Management, June 2016.