Assignment 1

Report

Of

Deployment Steps & Implementation

BIG DATA

Team Members

Mostafa Fathi Mohamed

Teacher Assistant @ Nile University

Deployment Steps

1. Docker-compose

- **Namenode Service:**
  - ✓ Image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
  - ✓ Container Name: namenode
  - ✓ Ports: Expose port 9870 on the host, mapped to port 9870 in the container.
  - ✓ Volumes: Mount the hadoop_namenode volume to /hadoop/dfs/name in the container.
  - ✓ Environment: Set CLUSTER_NAME to test.
  - ✓ Environment File: Use the configurations from hadoop.env.
- **Datanode Service:**
  - ✓ Image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
  - ✓ Container Name: datanode
  - ✓ Ports: Expose port 9864 on the host, mapped to port 9864 in the container.
  - ✓ Volumes: Mount the hadoop_datanode volume to /hadoop/dfs/data in the container.
  - ✓ Environment: Set SERVICE_PRECONDITION to "namenode:9870".
  - ✓ Environment File: Use the configurations from hadoop.env.
- **Resourcemanager Service:**
  - ✓ Image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8
  - ✓ Container Name: resourcemanager
  - ✓ Ports: Expose port 8088 on the host, mapped to port 8088 in the container.
  - ✓ Environment: Set SERVICE_PRECONDITION to "namenode:9000 namenode:9870 datanode:9864".
  - ✓ Environment File: Use the configurations from hadoop.env.
- **Nodemanager Service:**
  - ✓ Image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8
  - ✓ Container Name: nodemanager
  - ✓ Ports: Expose port 8042 on the host, mapped to port 8042 in the container.
  - ✓ Environment: Set SERVICE_PRECONDITION to "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088".
  - ✓ Environment File: Use the configurations from hadoop.env.
- **Historyserver Service:**
  - ✓ Image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8
  - ✓ Container Name: historyserver
  - ✓ Ports: Expose port 8188 on the host, mapped to port 8188 in the container.

    - ✓ Volumes: Mount the hadoop_historyserver volume to /hadoop/yarn/timeline in the container.
    - ✓ Environment File: Use the configurations from hadoop.env.
- **Spark Master Service:**
    - ✓ Image: bde2020/spark-master:3.0.0-hadoop3.2
    - ✓ Container Name: spark-master
    - ✓ Depends On: namenode, datanode
    - ✓ Ports: Expose ports 8080 and 7077 on the host, mapped to 8080 and 7077 in the container.
    - ✓ Environment:
        - o Set INIT_DAEMON_STEP to setup_spark.
        - o Set CORE_CONF_fs_defaultFS to hdfs://namenode:9000.
- **Spark Worker 1 Service:**
    - ✓ Image: bde2020/spark-worker:3.0.0-hadoop3.2
    - ✓ Container Name: spark-worker-1
    - ✓ Depends On: spark-master
    - ✓ Ports: Expose port 8081 on the host, mapped to port 8081 in the container.
    - ✓ Environment:
        - o Set SPARK_MASTER to spark://spark-master:7077.
        - o Set CORE_CONF_fs_defaultFS to hdfs://namenode:9000.
- **Volumes:**
    - ✓ Define three named volumes: hadoop_namenode, hadoop_datanode, and hadoop_historyserver. These volumes are used to persist data in the respective containers.

2. Hadoop:

- Download and install Hadoop using bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8 image in the docker container.
- Configure Hadoop settings in hadoop-env.sh, core-site.xml, and hdfs-site.xml.
- Format the Hadoop Distributed File System (HDFS).
- Start Hadoop services using start-all.sh or individual commands (start-dfs.sh, start-yarn.sh).

3. Spark:

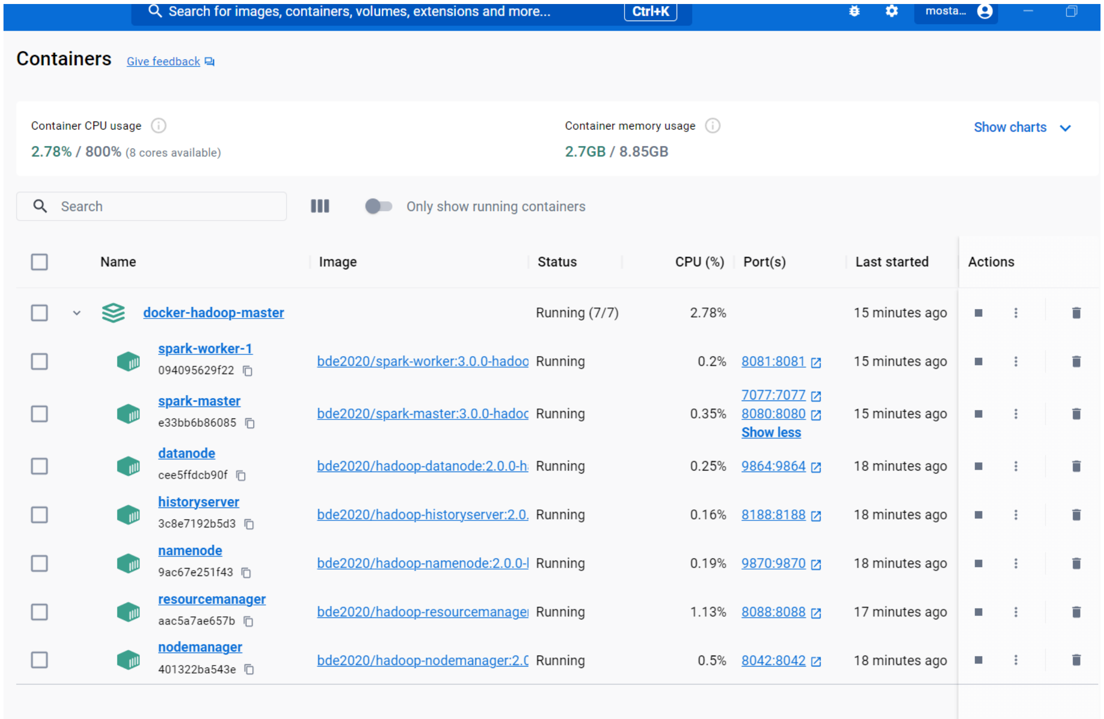
- Download and install Apache Spark using bde2020/spark-master:3.0.0-hadoop3.2 image in the docker container.
- Configure Spark settings in spark-env.sh.
- Set up Hadoop configuration in spark-defaults.conf.
- Start Spark services using start-master.sh and start-worker.sh.

4. YARN:

- Ensure Hadoop is correctly configured and running.
- Configure YARN settings in yarn-site.xml.
- Start the ResourceManager and NodeManagers using start-yarn.sh.

Implementations Steps

Using Docker-Compose to use spark, Hadoop and yarn: -



Search for images, containers, volumes, extensions and more...
Ctrl+K
mosta...

Containers Give feedback

Container CPU usage
2.78% / 800% (8 cores available)

Container memory usage
2.7GB / 8.85GB

Show charts

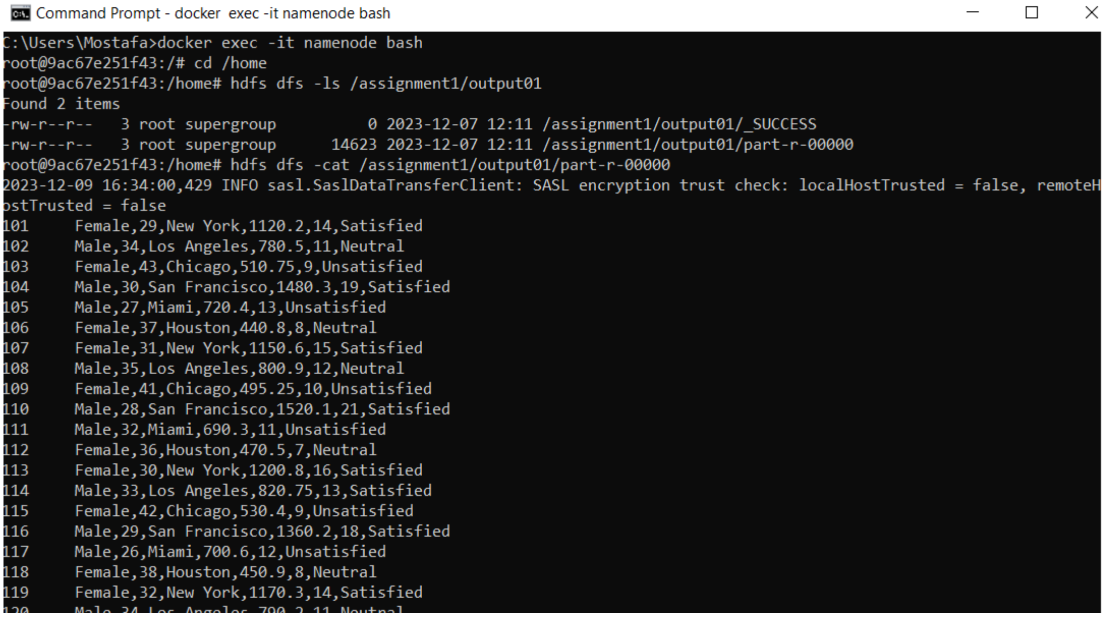
Search

Only show running containers

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
□	docker-hadoop-master		Running (7/7)	2.78%		15 minutes ago	
□	spark-worker-1 094095629f22	bde2020/spark-worker:3.0.0-hadoo	Running	0.2%	8081:8081	15 minutes ago	
□	spark-master e33bb6b86085	bde2020/spark-master:3.0.0-hadoc	Running	0.35%	7077:7077 8080:8080 Show less	15 minutes ago	
□	datanode cee5ffdcb90f	bde2020/hadoop-datanode:2.0.0-h	Running	0.25%	9864:9864	18 minutes ago	
□	historyserver 3c8e7192b5d3	bde2020/hadoop-historyserver:2.0.	Running	0.16%	8188:8188	18 minutes ago	
□	namenode 9ac67e251f43	bde2020/hadoop-namenode:2.0.0-l	Running	0.19%	9870:9870	18 minutes ago	
□	resourcemanager aac5a7ae657b	bde2020/hadoop-resourcemanage	Running	1.13%	8088:8088	17 minutes ago	
□	nodemanager 401322ba543e	bde2020/hadoop-nodemanager:2.0	Running	0.5%	8042:8042	18 minutes ago	

- **Hadoop -MapReduce**
  - Create MapReduce using java to select effective columns and Emit key-value pairs (customer Id as key, selected columns as value)
  - Store data in HDFs to be ready for processing from spark.

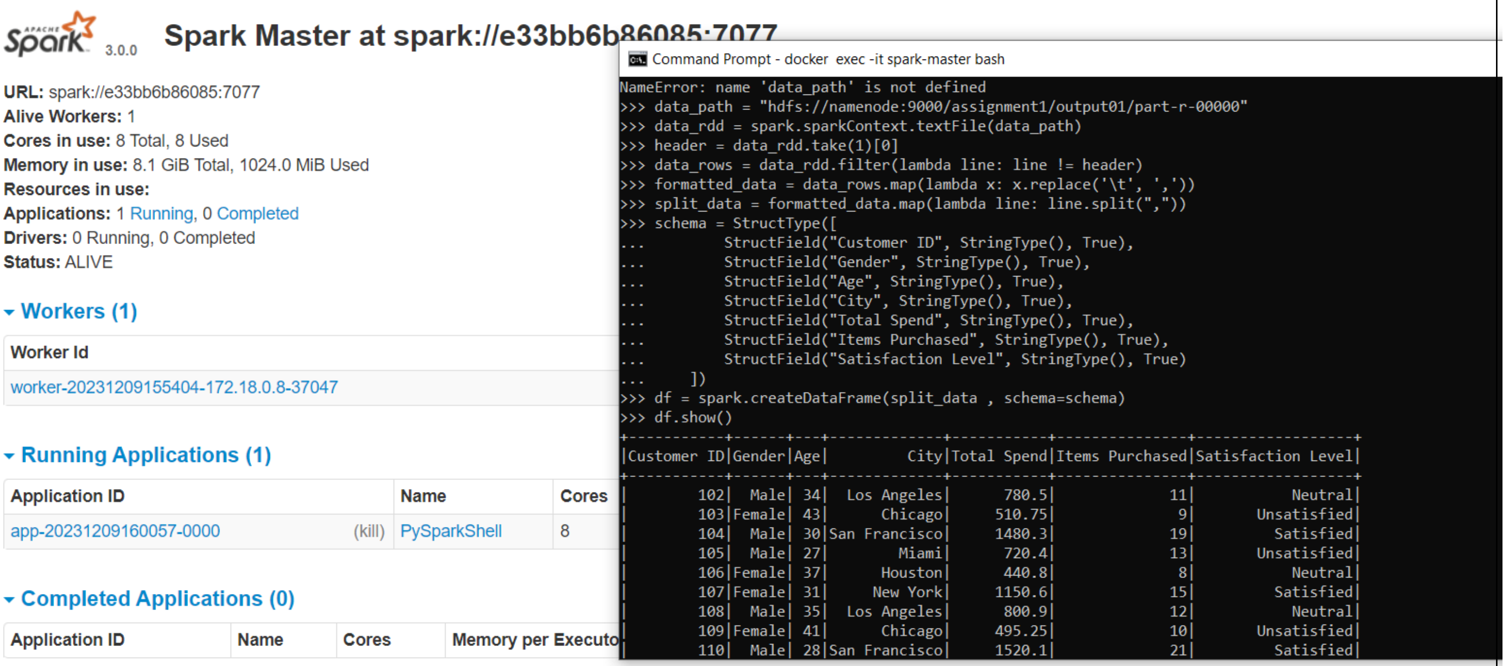
Now, Data In hdfs after preprocessing: -



```
C:\Users\Mostafa>docker exec -it namenode bash
root@9ac67e251f43:/# cd /home
root@9ac67e251f43:/home# hdfs dfs -ls /assignment1/output01
Found 2 items
-rw-r--r--   3 root supergroup          0 2023-12-07 12:11 /assignment1/output01/_SUCCESS
-rw-r--r--   3 root supergroup      14623 2023-12-07 12:11 /assignment1/output01/part-r-00000
root@9ac67e251f43:/home# hdfs dfs -cat /assignment1/output01/part-r-00000
2023-12-09 16:34:00,429 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteH
ostTrusted = false
101     Female,29,New York,1120.2,14,Satisfied
102     Male,34,Los Angeles,780.5,11,Neutral
103     Female,43,Chicago,510.75,9,Unsatisfied
104     Male,30,San Francisco,1480.3,19,Satisfied
105     Male,27,Miami,720.4,13,Unsatisfied
106     Female,37,Houston,440.8,8,Neutral
107     Female,31,New York,1150.6,15,Satisfied
108     Male,35,Los Angeles,800.9,12,Neutral
109     Female,41,Chicago,495.25,10,Unsatisfied
110     Male,28,San Francisco,1520.1,21,Satisfied
111     Male,32,Miami,690.3,11,Unsatisfied
112     Female,36,Houston,470.5,7,Neutral
113     Female,30,New York,1200.8,16,Satisfied
114     Male,33,Los Angeles,820.75,13,Satisfied
115     Female,42,Chicago,530.4,9,Unsatisfied
116     Male,29,San Francisco,1360.2,18,Satisfied
117     Male,26,Miami,700.6,12,Unsatisfied
118     Female,38,Houston,450.9,8,Neutral
119     Female,32,New York,1170.3,14,Satisfied
120     Male,34,Los Angeles,790.2,11,Neutral
```

- **Spark**
  - Start execute spark-master container and launch spark session: -
    1- docker exec -it spark-master bash
    2- /spark/bin/pyspark --master spark://spark-master:7077
  - Determine data path from hdfs: -
    data_path = "hdfs://namenode:9000/assignment1/output01/part-r-00000"
  - Convert Our data To RDD to start to work in it:-
    data_rdd = spark.sparkContext.textFile(data_path)

Apply some of data processing to reformate data from (key, value) to be data frame like this: -



Spark 3.0.0 **Spark Master at spark://e33bb6b86085:7077**

URL: spark://e33bb6b86085:7077
Alive Workers: 1
Cores in use: 8 Total, 8 Used
Memory in use: 8.1 GiB Total, 1024.0 MiB Used
Resources in use:
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▾ **Workers (1)**

Worker Id
worker-20231209155404-172.18.0.8-37047

▾ **Running Applications (1)**

Application ID	Name	Cores
app-20231209160057-0000 (kill)	PySparkShell	8

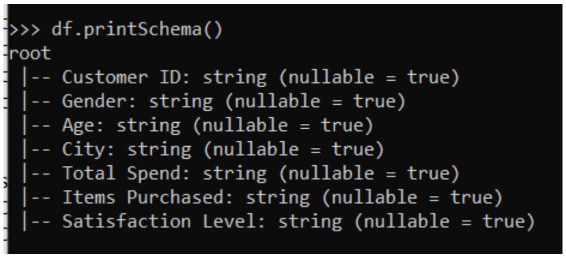
▾ **Completed Applications (0)**

Application ID	Name	Cores	Memory per Executo

Command Prompt - docker exec -it spark-master bash

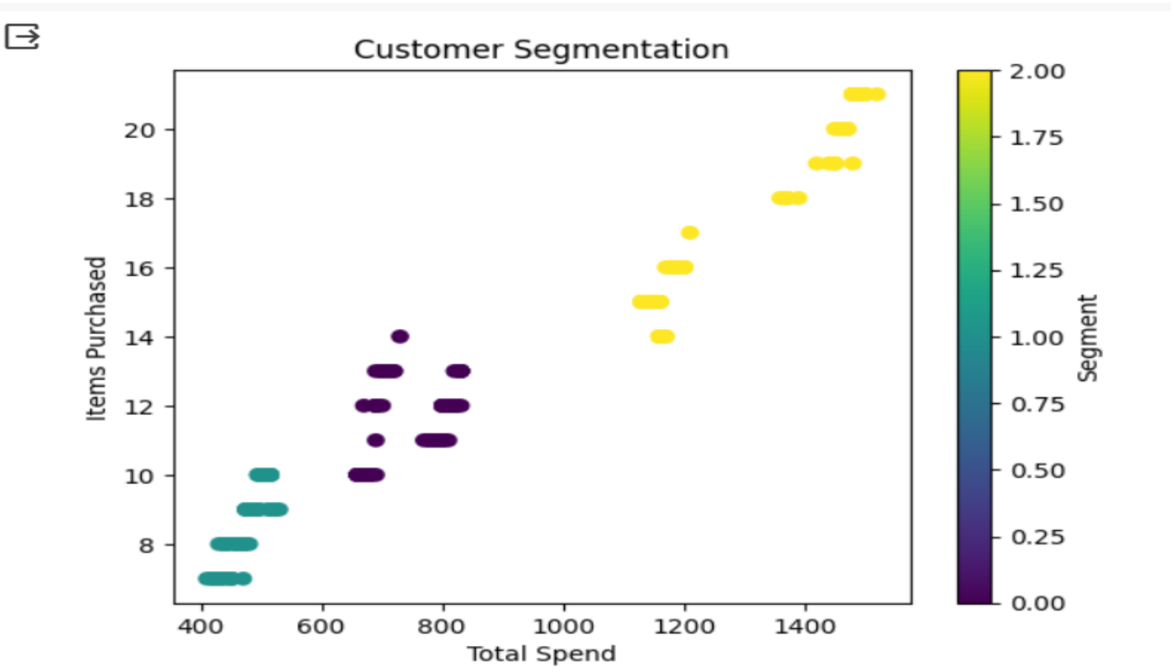
```
NameError: name 'data_path' is not defined
>>> data_path = "hdfs://namenode:9000/assignment1/output01/part-r-00000"
>>> data_rdd = spark.sparkContext.textFile(data_path)
>>> header = data_rdd.take(1)[0]
>>> data_rows = data_rdd.filter(lambda line: line != header)
>>> formatted_data = data_rows.map(lambda x: x.replace('\t', ','))
>>> split_data = formatted_data.map(lambda line: line.split(","))
>>> schema = StructType([
...         StructField("Customer ID", StringType(), True),
...         StructField("Gender", StringType(), True),
...         StructField("Age", StringType(), True),
...         StructField("City", StringType(), True),
...         StructField("Total Spend", StringType(), True),
...         StructField("Items Purchased", StringType(), True),
...         StructField("Satisfaction Level", StringType(), True)
...     ])
>>> df = spark.createDataFrame(split_data , schema=schema)
>>> df.show()
+-----------+------+---+-------------+-----------+---------------+------------------+
|Customer ID|Gender|Age|         City|Total Spend|Items Purchased|Satisfaction Level|
+-----------+------+---+-------------+-----------+---------------+------------------+
|        102|  Male| 34|  Los Angeles|      780.5|             11|           Neutral|
|        103|Female| 43|      Chicago|     510.75|              9|       Unsatisfied|
|        104|  Male| 30|San Francisco|     1480.3|             19|         Satisfied|
|        105|  Male| 27|        Miami|      720.4|             13|       Unsatisfied|
|        106|Female| 37|      Houston|      440.8|              8|           Neutral|
|        107|Female| 31|     New York|     1150.6|             15|         Satisfied|
|        108|  Male| 35|  Los Angeles|      800.9|             12|           Neutral|
|        109|Female| 41|      Chicago|     495.25|             10|       Unsatisfied|
|        110|  Male| 28|San Francisco|     1520.1|             21|         Satisfied|
```

- After Appling Preprocessing and some of Data Engineering using spark, we stored output in hdfs:-
  output_path = "hdfs://namenode:9000/assignment1/output_csv"
  csv_data = split_data.map(lambda line: ",".join(line))
  csv_data.saveAsTextFile(output_path)
- We are using collab to apply spark ml and use the output from the previous operation which is data stored in HDFS.
- Now, our data contain the following columns: -



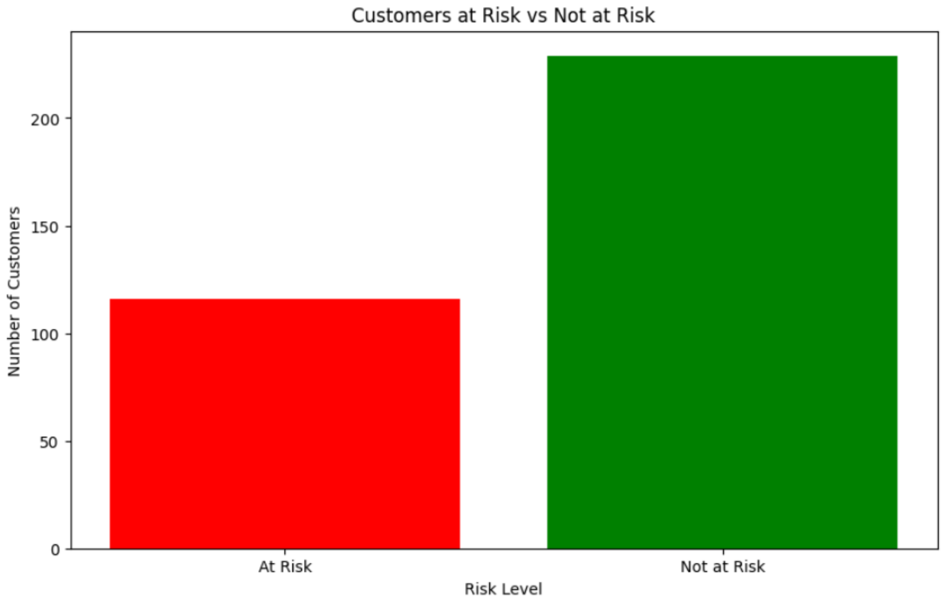
```
>>> df.printSchema()
root
 |-- Customer ID: string (nullable = true)
 |-- Gender: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- City: string (nullable = true)
 |-- Total Spend: string (nullable = true)
 |-- Items Purchased: string (nullable = true)
 |-- Satisfaction Level: string (nullable = true)
```

- Start our work to answer 3 Questions: -
  1. Can we segment customers based on their demographic information (Age, Gender, City) and shopping behaviors (Total Spend, Number of Items Purchased, Membership Type)?
     ➔ To answer this question, we applied KMeans algorithm and using scatter plot to visualize our results.



2. Which customers are at risk of not making future purchases based on their Days?

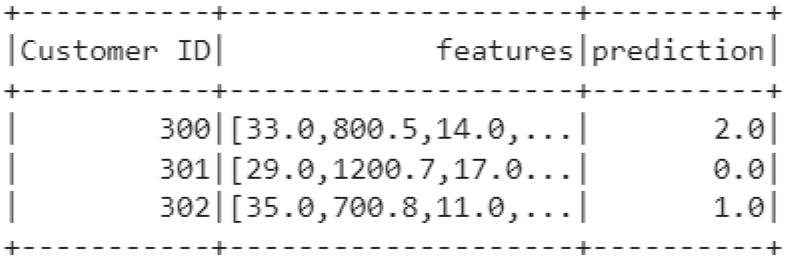
   ➔ To answer this question, we used RandomForestClassifier to classify customers with some features specially “Days Since Last Purchase”, “Items Purchased” and “Total Spend”



3. Can we predict a customer’s Satisfaction Level based on their demographic and purchase history data?
    - ➔ To answer this question, we use DecisionTreeClassifier to classify our customers by using feature like "Age", "Total Spend", "Items Purchased", "GenderIndex", "CityIndex" and use MulticlassClassificationEvaluator to evaluate our model and test model by trying to predict new data and get result as following:-

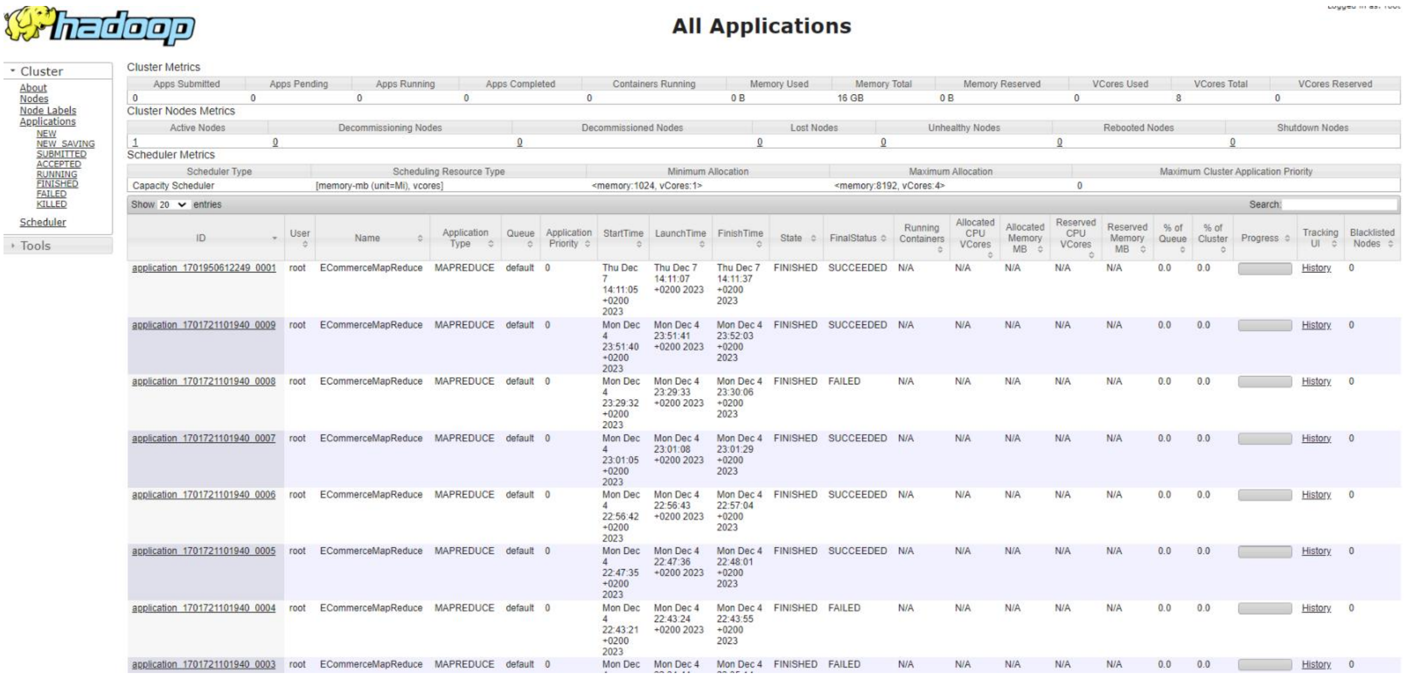
```
new_data_samples = [
   (300, "Male", 33, "Chicago", 800.5, 14),
   (301, "Female", 29, "New York", 1200.7, 17),
   (302, "Male", 35, "Los Angeles", 700.8, 11)
]
```

✓ **The output**



```
+-----------+--------------------+----------+
|Customer ID|            features|prediction|
+-----------+--------------------+----------+
|        300|[33.0,800.5,14.0,...|       2.0|
|        301|[29.0,1200.7,17.0...|       0.0|
|        302|[35.0,700.8,11.0,...|       1.0|
+-----------+--------------------+----------+
```

- **Yarn**
    - Using as resource management layer for Apache Hadoop allows Spark and other applications to efficiently share and allocate resources in a Hadoop cluster. And the following is screenshot from resource manager for all application and screenshot for specific application: -



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	16 GB	0 B	0	8	0

Cluster Nodes Metrics

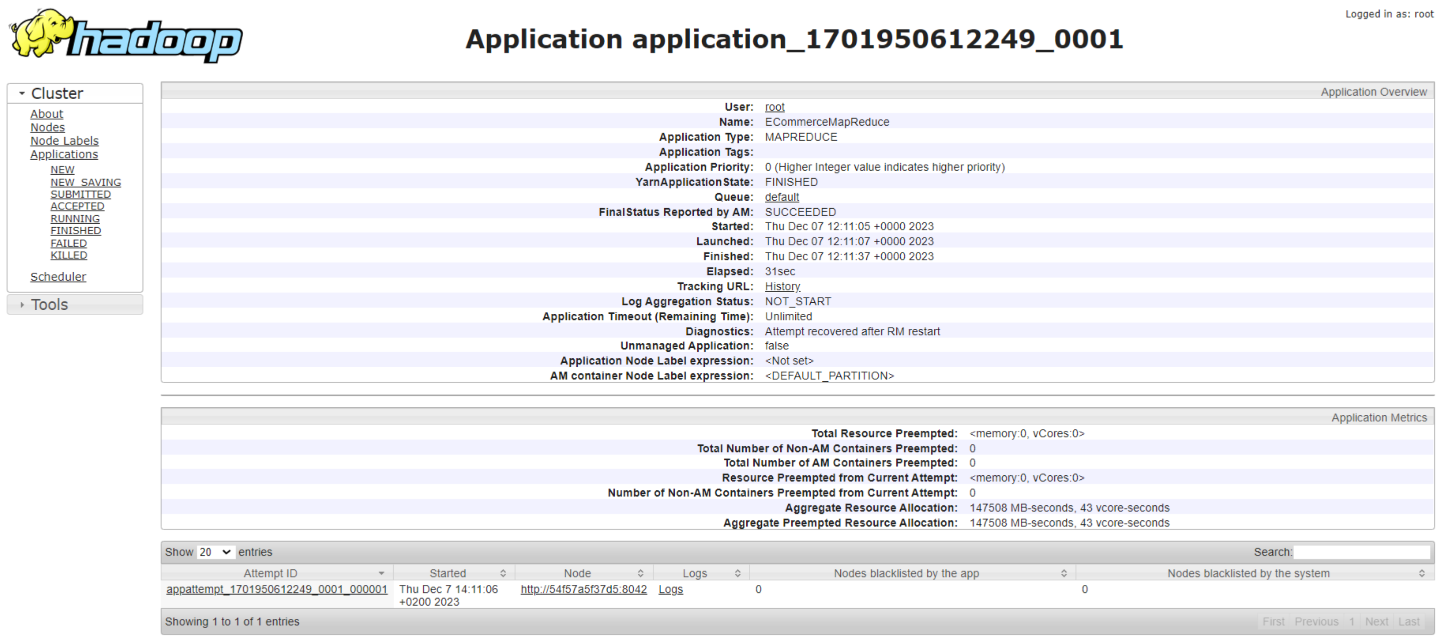
Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1701950612249_0001	root	ECommerceMapReduce	MAPREDUCE	default	0	Thu Dec 7 14:11:05 +0200 2023	Thu Dec 7 14:11:07 +0200 2023	Thu Dec 7 14:11:37 +0200 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0009	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec 4 23:51:40 +0200 2023	Mon Dec 4 23:51:41 +0200 2023	Mon Dec 4 23:52:03 +0200 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0008	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec 4 23:29:32 +0200 2023	Mon Dec 4 23:29:33 +0200 2023	Mon Dec 4 23:30:06 +0200 2023	FINISHED	FAILED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0007	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec 4 23:01:05 +0200 2023	Mon Dec 4 23:01:08 +0200 2023	Mon Dec 4 23:01:29 +0200 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0006	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec 4 22:56:42 +0200 2023	Mon Dec 4 22:56:43 +0200 2023	Mon Dec 4 22:57:04 +0200 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0005	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec 4 22:47:35 +0200 2023	Mon Dec 4 22:47:36 +0200 2023	Mon Dec 4 22:48:01 +0200 2023	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0004	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec 4 22:43:21 +0200 2023	Mon Dec 4 22:43:24 +0200 2023	Mon Dec 4 22:43:55 +0200 2023	FINISHED	FAILED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0
application_1701721101940_0003	root	ECommerceMapReduce	MAPREDUCE	default	0	Mon Dec	Mon Dec 4	Mon Dec 4	FINISHED	FAILED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0



Logged in as: root

Application application_1701950612249_0001

Application Overview

User:	root
Name:	ECommerceMapReduce
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Thu Dec 07 12:11:05 +0000 2023
Launched:	Thu Dec 07 12:11:07 +0000 2023
Finished:	Thu Dec 07 12:11:37 +0000 2023
Elapsed:	31sec
Tracking URL:	History
Log Aggregation Status:	NOT_START
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	Attempt recovered after RM restart
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	147508 MB-seconds, 43 vcore-seconds
Aggregate Preempted Resource Allocation:	147508 MB-seconds, 43 vcore-seconds

Show 20 entries

Search:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1701950612249_0001_000001	Thu Dec 7 14:11:06 +0200 2023	http://54f57a5f37d5:8042	Logs	0	0

Showing 1 to 1 of 1 entries