Big Data Hadoop and Spark Developer

Project 4 : Market Analysis in Banking Domain

Suresh Anuganti

STEP 01: Load data into hdfs / spark using FTP

hdfs dfs -put banking.csv /user/sureshmecad_gmail/02Nov2019_SureshA

STEP 02: Start Spark Shell including packages

```
[sureshmecad_gmail@ip-10-0-1-10 ~]$ spark2-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42001. Attempting port 42002.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42002. Attempting port 42003.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42003. Attempting port 42004.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42004. Attempting port 42005.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42005. Attempting port 42006.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42006. Attempting port 42007.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42007. Attempting port 42008.
19/12/02 15:16:02 WARN util.Utils: Service 'SparkUI' could not bind on port 42008. Attempting port 42009.
19/12/02 15:16:07 WARN lineage.LineageWriter: Lineage directory /var/log/spark2/lineage doesn't exist or i
n will be disabled.
Spark context Web UI available at http://ip-10-0-1-10.ec2.internal:42009
Spark context available as 'sc' (master = yarn, app id = application_1567150833346_23501).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0.cloudera2
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_144)
Type in expressions to have them evaluated.
Type :help for more information.
```

1. Load data and create a Spark data frame

```
val sqlContext = new org.apache.spark.sql.SQLContext(sc);
        import sqlContext.implicits._;


        val input = sc.textFile("banking.csv")
        input.collect()
        input.take(3)
```

```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc);
warning: there was one deprecation warning; re-run with -deprecation for details
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@72259917

scala> import sqlContext.implicits._;
import sqlContext.implicits._

scala> val input = sc.textFile("banking.csv")
input: org.apache.spark.rdd.RDD[String] = banking.csv MapPartitionsRDD[1] at textFile at <console>:29
```

```
scala> input.collect()
res0: Array[String] = Array("age;""job"";""marital"";""education"";""default"";""bala
ion"";""campaign"";""pdays"";""previous"";""poutcome"";""y""", "58;""management"";""m
"may"";261;1;-1;0;""unknown"";""no""", "44;""technician"";""single"";""secondary"";""
n"";""no""", "33;""entrepreneur"";""married"";""secondary"";""no"";2;""yes"";""yes"";
collar"";""married"";""unknown"";""no"";1506;""yes"";""no"";""unknown"";5;""may"";92;
"";""no"";1;""no"";""no"";""unknown"";5;""may"";198;...
scala> input.take(3)
res1: Array[String] = Array("age;""job"";""marital"";""education"";""default"";""bala
ion"";""campaign"";""pdays"";""previous"";""poutcome"";""y""", "58;""management"";""m
"may"";261;1;-1;0;""unknown"";""no""", "44;""technician"";""single"";""secondary"";""
n"";""no""")
```

```
val dataDF =
input.map(x=>x.replace("\"","").split(';')).map(y=>(y(0),y(1),y(2),y(3),y(4),y(5),y(6),y(7),y(8),y(
9),y(10),y(11),y(12),y(13),y(14),y(15),y(16))).toDF("age","job","marital","education","default",
"balance","housing","loan","contact","month","day_of_week","duration","campaign","pdays
","previous","poutcome","y");


        val newDataDF = dataDF.where(col("age") !== "age")
```

```
scala> val dataDF = input.map(x=>x.replace("\"","").split(';')).map(y=>(y(0),y(1),y(2
4),y(15),y(16))).toDF("age","job","marital","education","default","balance","housing"
days","previous","poutcome","y");
19/12/02 15:39:04 WARN lineage.LineageWriter: Lineage directory /var/log/spark2/linea
n will be disabled.
dataDF: org.apache.spark.sql.DataFrame = [age: string, job: string ... 15 more fields

scala> val newDataDF = dataDF.where(col("age") !== "age")
warning: there was one deprecation warning; re-run with -deprecation for details
newDataDF: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [age: string, job
```

2. Give marketing success rate (No. of people y / total no. of entries)

```
val totalcount = newDataDF.count().toDouble
val deposit_issued = newDataDF.groupBy("y").agg(count("y").as("count"))
deposit_issued.show()
val success_count:Float = deposit_issued.where($"y" === "yes").select("count").first().getLo
 val success_rate = (success_count)/(totalcount)*100
```

```
scala> val totalcount = newDataDF.count().toDouble
totalcount: Double = 45211.0
```

```
scala> val deposit_issued = newDataDF.groupBy("y").agg(count("y").as("count"))
deposit_issued: org.apache.spark.sql.DataFrame = [y: string, count: bigint]

scala> deposit_issued.show()
+---+-----+
|  y|count|
+---+-----+
| no|39922|
|yes| 5289|
+---+-----+
```

```
scala>val success_count:Float = deposit_issued.where($"y" === "yes").select("count").f
success_count: Float = 5289.0

scala> val success_rate = (success_count)/(totalcount)*100
success_rate: Double = 11.698480458295547
```

3. Give marketing failure rate

```
val failure_count:Float = deposit_issued.where($"y" === "no").select("count").first().getLong

val failure_rate = (failure_count)/(totalcount)*100
```

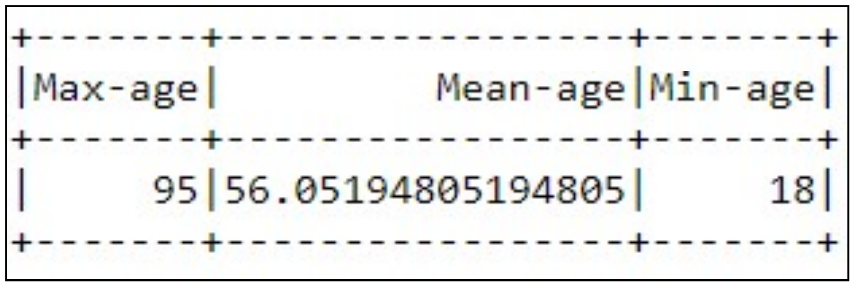
```
scala> val failure_count:Float = deposit_issued.where($"y" === "no").select("count").first().getLong(0)
failure_count: Float = 39922.0

scala> val failure_rate = (failure_count)/(totalcount)*100
failure_rate: Double = 88.30151954170445
```

4. Give the maximum, mean, and minimum age of the average targeted customer

```
newDataDF.groupBy("age").count().agg(max("age").as("Max-age"),avg("age").as("Mean-age"),min("age").as("Min-age")).show()
```

```
scala> newDataDF.groupBy("age").count().agg(max("age").as("Max-age"),avg("age").as("Mean-age"),min("age").as("Min-age")).show()
```



Max-age	Mean-age	Min-age
95	56.05194805194805	18

5. Check the quality of customers by checking average balance, median balance of customers

```
newDataDF.registerTempTable("bankdata");

sqlContext.sql("select avg(balance) as average , percentile(balance,0.5) as median from bankdata").show()
```

```
scala> newDataDF.registerTempTable("bankdata");
warning: there was one deprecation warning; re-run with -deprecation for details

scala> sqlContext.sql("select avg(balance) as average , percentile(balance,0.5) as median from bankdata").show()
+------------------+------+
|           average|median|
+------------------+------+
|1362.2720576850766| 448.0|
+------------------+------+
```

6. Check if age matters in marketing subscription for deposit

```
newDataDF.groupBy("age","y").count().where($"y" === "yes").show()
```

```
scala> newDataDF.groupBy("age","y").count().where($"y" === "yes").show()
+---+---+-----+
|age|  y|count|
+---+---+-----+
| 59|yes|   88|
| 49|yes|  101|
| 52|yes|   85|
| 26|yes|  134|
| 86|yes|    4|
| 90|yes|    2|
| 79|yes|   10|
| 61|yes|   57|
| 25|yes|  113|
| 35|yes|  209|
| 32|yes|  221|
| 18|yes|    7|
| 48|yes|   82|
| 95|yes|    1|
| 53|yes|   85|
| 45|yes|  106|
| 20|yes|   15|
| 55|yes|   76|
| 63|yes|   30|
| 19|yes|   11|
+---+---+-----+
only showing top 20 rows
```

Age doesn't matter in marketing subscription

7. Check if marital status mattered for a subscription to deposit

```
newDataDF.groupBy("marital","y").count().where($"y" === "yes").show()
```

```
scala> newDataDF.groupBy("marital","y").count().where($"y" === "yes").show()
+--------+---+-----+
| marital|  y|count|
+--------+---+-----+
|divorced|yes|  622|
|  single|yes| 1912|
| married|yes| 2755|
+--------+---+-----+
```

Marital status doesn't matter in marketing subscription pattern

8. Check if age and marital status together mattered for a subscription to deposit scheme

```
newDataDF.groupBy("age","marital","y").count().where($"y" === "yes").show()
```

```
scala> newDataDF.groupBy("age","marital","y").count().where($"y" === "yes").show()
+---+--------+---+-----+
|age| marital|  y|count|
+---+--------+---+-----+
| 43| married|yes|   62|
| 24| married|yes|   10|
| 73|  single|yes|    1|
| 54|  single|yes|    8|
| 22|  single|yes|   40|
| 21|  single|yes|   21|
| 33|  single|yes|
| 29|  single|yes|
| 25|  single|yes|   99|
| 77| married|yes|   19|
| 19|  single|yes|   11|
| 28|  single|yes|  138|
| 55|divorced|yes|   26|
| 28| married|yes|   20|
| 85| married|yes|    3|
| 43|divorced|yes|   15|
| 69|  single|yes|    1|
| 50|  single|yes|    4|
| 42| married|yes|   70|
| 36| married|yes|  100|
+---+--------+---+-----+
only showing top 20 rows
```

Age and Marital status together doesn't matter in marketing subs

9. Do feature engineering for the bank and find the right age effect on the campaign

```
val dfWithAgeCat = newDataDF.withColumn("age-cat",when (col("age") < 20 , "1 to 19").when (col("age") >= 20 && col("age") < 30 , "20 to 29").when (col("age") >= 30 && col("age") < 40 , "30 to 39").when (col("age") >= 40 && col("age") < 50 , "40 to 49").when (col("age") >= 50 && col("age") < 60 , "50 to 59").when (col("age") >= 60, "60 and above"))

dfWithAgeCat.groupBy("age-cat","y").count().where($"y" === "yes").orderBy("age-cat").show()
```

```
scala> val dfWithAgeCat = newDataDF.withColumn("age-cat",when (col("age") < 20 , "1 to 19").when (col("age") >= 20 && col("age") < 30 , "20 to 29").w
hen (col("age") >= 30 && col("age") < 40 , "30 to 39").when (col("age") >= 40 && col("age") < 50 , "40 to 49").when (col("age") >= 50 && col("age") <
 60 , "50 to 59").when (col("age") >= 60, "60 and above"))
dfWithAgeCat: org.apache.spark.sql.DataFrame = [age: string, job: string ... 16 more fields]
```

```
scala> dfWithAgeCat.groupBy("age-cat","y").count().where($"y" === "yes").orderBy("age-cat").show()
+------------+---+-----+
|     age-cat|  y|count|
+------------+---+-----+
|     1 to 19|yes|   18|
|    20 to 29|yes|  910|
|    30 to 39|yes| 1913|
|    40 to 49|yes| 1063|
|    50 to 59|yes|  785|
|60 and above|yes|  600|
+------------+---+-----+
```