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 mydf =
spark.read.format("csv").option("header","true").option("delimiter",";").load("/user/sureshmecad_gmail/02Nov2019_SureshA/banking.csv")
```

```
scala> val mydf = spark.read.format("csv").option("header","true").option("delimiter",";").load("/user/sureshmev")
19/12/07 04:12:17 WARN lineage.LineageWriter: Lineage directory /var/log/spark2/lineage doesn't exist or is not
n will be disabled.
mydf: org.apache.spark.sql.DataFrame = ["age;""job"": string, ""marital"": string ... 14 more fields]
```

mydf.printSchema

```
scala> mydf.printSchema
root
 |-- age: string (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: string (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- day: string (nullable = true)
 |-- month: string (nullable = true)
 |-- duration: string (nullable = true)
 |-- campaign: string (nullable = true)
 |-- pdays: string (nullable = true)
 |-- previous: string (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- y: string (nullable = true)
```

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

```
val market_success =

(mydf.filter($"y"==="yes").count.toDouble)/(mydf.count.toDouble)*100
```

```
scala> val market_success=(mydf.filter($"y"==="yes").count.toDouble)/(mydf.count.toDouble)*100
market_success: Double = 11.698480458295547
```

2a. Give marketing failure rate

```
val market_failure1 =

(mydf.filter($“y”===“no”).count.toDouble)/(mydf.count.toDouble)*100
```

```
scala> val market_failure1=(mydf.filter($"y"==="no").count.toDouble)/(mydf.count.toDouble)*100
market_failure1: Double = 88.30151954170445
```

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

```
scala> mydf.createOrReplaceTempView("banking")

scala> sql("select min(age),max(age),avg(age) from banking").show
+--------+--------+-------------------------+
|min(age)|max(age)|avg(CAST(age AS DOUBLE))|
+--------+--------+-------------------------+
|      18|      95|       40.93621021432837|
+--------+--------+-------------------------+
```

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

```
scala> dataframe1.createOrReplaceTempView("banking")

scala> sql("select percentile_approx(balance,0.5) as median , avg(balance) from banking").show
+------+----------------------------+
|median|avg(CAST(balance AS DOUBLE))|
+------+----------------------------+
| 448.0|          1362.2720576850766|
+------+----------------------------+
```

5. Check if age matters in marketing subscription for deposit

```
scala> sql("select age, count(age) from banking where y = 'yes' group by age order by count(age) desc").show
```

```
+---+----------+
|age|count(age)|
+---+----------+
| 32|       221|
| 30|       217|
| 33|       210|
| 35|       209|
| 31|       206|
| 34|       198|
| 36|       195|
| 29|       171|
| 37|       170|
| 28|       162|
| 38|       144|
| 39|       143|
| 27|       141|
| 26|       134|
| 41|       120|
| 46|       118|
| 40|       116|
| 25|       113|
| 47|       113|
| 42|       111|
+---+----------+
only showing top 20 rows
```

Age doesn't matter in marketing subscription

6. 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

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

```
scala> mydf.select("marital","age").filter('y==="yes").groupBy('age,'marital).count.sort('count.desc).show
```

```
+---+-------+-----+
|age|marital|count|
+---+-------+-----+
| 30| single|  151|
| 28| single|  138|
| 29| single|  133|
| 32| single|  124|
| 26| single|  121|
| 34|married|  118|
| 31| single|  111|
| 27| single|  110|
| 35|married|  101|
| 36|married|  100|
| 25| single|   99|
| 37|married|   98|
| 33|married|   97|
| 33| single|   97|
| 39|married|   87|
| 32|married|   87|
| 38|married|   86|
| 35| single|   84|
| 47|married|   83|
| 31|married|   80|
+---+-------+-----+
only showing top 20 rows
```

Age and Marital status together doesn't matter in marketing subscription pattern

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

```
scala> val newdf = mydf.withColumn("category",when('age < 25,"young").otherwise(when('age>60,"old").otherwise("mid_age")))
newdf: org.apache.spark.sql.DataFrame = [age: string, job: string ... 16 more fields]
```

```
scala> newdf.groupBy('category,'y).count.sort('count.desc).show
+--------+---+-----+
|category|  y|count|
+--------+---+-----+
| mid_age| no|38634|
| mid_age|yes| 4580|
|     old| no|  686|
|   young| no|  602|
|     old|yes|  502|
|   young|yes|  207|
+--------+---+-----+
```