Big Data Hadoop Spark Developer

Marketing Analysis Banking Domain Project

Christina Grys

Background and Objective: A client from Portuguese banking institution, ran a marketing campaign to convince potential customers to invest in a bank term deposit scheme. The marketing campaigns were based on phone calls. Often, the same customer was contacted more than once through phone, to assess if they would want to subscribe to the bank term deposit or not. Data engineer is to perform the marketing analysis of the data generated by this campaign.

Domain: Banking (Market Analysis)

Dataset Description: The data fields are as follows below.

Dataset	Categories
Age	numeric
Job	Type of job - admin, blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, student, technician, unemployed, unknown.
Marital	Marital status - married, divorced, single, unknown, note: divorced means divorced or widowed
Education	Level of education - (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate', 'professional.course', 'university.degree', 'unknown')
Default	has credit in default? (categorical: 'no', 'yes', 'unknown')
Housing	has housing loan? (categorical: 'no', 'yes', 'unknown')
Loan	has personal loan? (categorical: 'no', 'yes', 'unknown')
Contact	contact communication type (categorical: 'cellular', 'telephone')
Month	Month of last contact (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
Day of Week	last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
Duration	last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (example, if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call “y” is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
Campaign	number of times a customer was contacted during the campaign (numeric, includes last contact)
pDays	number of days passed after the customer was last contacted from a previous campaign (numeric; 999 means customer was not previously contacted)
Previous	number of times the customer was contacted prior to (or before) this campaign (numeric)
Poutcome	outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', ‘success’)
Employment variation rate	emp.var.rate: - quarterly indicator (numeric)
Consumer Price Index	cons.price.idx: - monthly indicator (numeric)
Consumer Confidence Index	cons.conf.idx: - monthly indicator (numeric)
Euribor 3-month rate	euribor3m: - daily indicator (numeric)
Number of employees	nr.employed: - quarterly indicator (numeric)
Output	Desired target
Y	has the customer subscribed a term deposit? (binary: 'yes', 'no')

Analysis Tasks

1. Load data and create Spark data frame
2. Give marketing success rate. (No. of people subscribed / total no. of entries)
    a. Give marketing failure rate
3. Maximum, Mean, and Minimum age of average targeted customer
4. Check quality of customers by checking average balance, median balance of customers
5. Check if age matters in marketing subscription for deposit
6. Check if marital status mattered for subscription to deposit.
7. Check if age and marital status together mattered for subscription to deposit scheme
8. Do feature engineering for column—age and find right age effect on campaign

Breakdown Report from using Hadoop Spark

Logged into personal account and access to spark-shell

```
ip-10-0-42-218 login: grys1cmgmail
Password:
Last login: Tue Feb 22 19:41:43 on pts/153
[grys1cmgmail@ip-10-0-42-218 ~]$ spark-shell --packages.com.databricks:spark-csv_2.11:1.5.0
Setting default log level to "ERROR".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
bad option: '--packages.com.databricks:spark-csv_2.11:1.5.0'
[grys1cmgmail@ip-10-0-42-218 ~]$ spark-shell
Setting default log level to "ERROR".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/02/22 20:31:56 WARN lineage.LineageWriter: Lineage directory /var/log/spark/lineage doesn't exist
or is not writable. Lineage for this application will be disabled.
Spark context available as 'sc' (master = yarn, app id = application_1640258093152_11073).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0-cdh6.3.2
      /_/

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.
```

Import Package: import org.apache.spark.sql.SQLContext

Initialize the sqlContext object: val sqlContext = new SQLContext(sc)

```
scala> import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.SQLContext

scala> val sqlContext = new 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@33acec5e
```

QUESTION 1: Load data and create Spark data frame

```
scala> val bankCamp_df = sqlContext.read.format("csv").option("header", "true").option("inferSchema", "true").option("se
p", ",").load("/user/grys1cmgmail/sparkproject.txt")
bankCamp_df: org.apache.spark.sql.DataFrame = [age: int, job: string ... 15 more fields]

scala> bankCamp_df.columns
res10: Array[String] = Array(age, job, marital, education, default, balance, housing, loan, contact, day, month, duratio
n, campaign, pdays, previous, poutcome, y)

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


scala> bankCamp_df.show(5)
+---+-----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+----
----+---+
|age|        job|marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|pout
come|  y|
+---+-----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+----
----+---+
| 58|  management|married| tertiary|     no|   2143|    yes|  no|unknown|  5|  may|     261|       1|   -1|       0| unk
nown| no|
| 44|  technician| single|secondary|     no|     29|    yes|  no|unknown|  5|  may|     151|       1|   -1|       0| unk
nown| no|
| 33|entrepreneur|married|secondary|     no|      2|    yes| yes|unknown|  5|  may|      76|       1|   -1|       0| unk
nown| no|
| 47| blue-collar|married|  unknown|     no|   1506|    yes|  no|unknown|  5|  may|      92|       1|   -1|       0| unk
nown| no|
| 33|     unknown| single|  unknown|     no|      1|     no|  no|unknown|  5|  may|     198|       1|   -1|       0| unk
nown| no|
+---+-----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+----
----+---+
only showing top 5 rows
```

QUESTION 2: Give marketing success rate. (No. of people subscribed / total no. of entries)

a. Give marketing failure rate

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

```
scala> sqlContext.sql("select * from Marketing_Analysis_Table limit 4").show();
+---+-----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
|age|        job|marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|  y|
+---+-----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
| 58|  management|married| tertiary|     no|   2143|    yes|  no|unknown|  5|  may|     261|       1|   -1|       0| unknown| no|
| 44|  technician| single|secondary|     no|     29|    yes|  no|unknown|  5|  may|     151|       1|   -1|       0| unknown| no|
| 33|entrepreneur|married|secondary|     no|      2|    yes| yes|unknown|  5|  may|      76|       1|   -1|       0| unknown| no|
| 47| blue-collar|married|  unknown|     no|   1506|    yes|  no|unknown|  5|  may|      92|       1|   -1|       0| unknown| no|
+---+-----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+


scala>
```

```
scala> val totalRecords = bankCamp_df.count()
totalRecords: Long = 45211

scala> val numSubscribed = bankCamp_df.filter(bankCamp_df("y") === "yes").count()
numSubscribed: Long = 5289

scala> val successRate = (numSubscribed.toFloat/totalRecords * 1000).round / 1000.toDouble
successRate: Double = 0.117

scala> println(successRate)
0.117

scala> val notSubscribed = bankCamp_df.filter(bankCamp_df("y") === "no").count()
notSubscribed: Long = 39922

scala> val failRate = (notSubscribed.toFloat/totalRecords * 1000).round / 1000.toDouble
failRate: Double = 0.883

scala> println(failRate)
0.883
```

QUESTION 3: Maximum, Mean, and Minimum age of average targeted customer

```
scala> val ageDescStat = sqlContext.sql("""select max(age) MAX_AGE, avg(age) AVERAGE_AGE, min(age) MIN_AGE from Marketing_Analysis_Table""")
ageDescStat: org.apache.spark.sql.DataFrame = [MAX_AGE: int, AVERAGE_AGE: double ... 1 more field]

scala> ageDescStat.show()
+-------+-----------------+-------+
|MAX_AGE|      AVERAGE_AGE|MIN_AGE|
+-------+-----------------+-------+
|     95|40.93621021432837|     18|
+-------+-----------------+-------+
```

QUESTION 4: Check quality of customers by checking average balance, median balance of customers

```
scala> val custQuality = sqlContext.sql("select avg(balance) Average_Balance, percentile_approx(balance, 0.5) Median_Balance from Marketing_Analysis_Table")
custQuality: org.apache.spark.sql.DataFrame = [Average_Balance: double, Median_Balance: int]

scala> custQuality.show
+-----------------+--------------+
|   Average_Balance|Median_Balance|
+-----------------+--------------+
|1362.2720576850766|           448|
+-----------------+--------------+
```

QUESTION 5: Check if age matters in marketing subscription for deposit

```
scala> val ageAndSubscrip = sqlContext.sql("select avg(age) Average_Age, y Subscribed from  Marketing_Analysis_Table  group by y")
ageAndSubscrip: org.apache.spark.sql.DataFrame = [Average_Age: double, Subscribed: string]

scala> ageAndSubscrip.show
+-----------------+----------+
|       Average_Age|Subscribed|
+-----------------+----------+
| 40.83898602274435|        no|
|41.670069956513515|       yes|
+-----------------+----------+
```

QUESTION 6: Check if marital status mattered for subscription to deposit.

```
scala> val marriageAndSubscrip = sqlContext.sql("select marital Marital_Status, y Subscription, count(marital) Subscription_Count from Marketing_Analysis_Table  group by ma
rital, y order by y")
marriageAndSubscrip: org.apache.spark.sql.DataFrame = [Marital_Status: string, Subscription: string ... 1 more field]

scala> marriageAndSubscrip.show
+--------------+------------+------------------+
|Marital_Status|Subscription|Subscription_Count|
+--------------+------------+------------------+
|      divorced|          no|              4585|
|        single|          no|             10878|
|       married|          no|             24459|
|      divorced|         yes|               622|
|       married|         yes|              2755|
|        single|         yes|              1912|
+--------------+------------+------------------+
```

QUESTION 7: Check if age and marital status together mattered for subscription to deposit scheme

```
scala> val ageAndMarital = sqlContext.sql("select marital Marital_Status, y Subscription, count(marital) Subsp
tn_By_Marriage, avg(age) Average_Age from Marketing_Analysis_Table group by marital, y order by y")
ageAndMarital: org.apache.spark.sql.DataFrame = [Marital_Status: string, Subscription: string ... 2 more field
s]

scala> ageAndMarital.show
+--------------+------------+------------------+------------------+
|Marital_Status|Subscription|Subsptn_By_Marriage|       Average_Age|
+--------------+------------+------------------+------------------+
|      divorced|          no|              4585| 45.31297709923664|
|        single|          no|             10878| 33.96258503401361|
|       married|          no|             24459| 43.05854695613067|
|      divorced|         yes|               622|49.247588424437296|
|       married|         yes|              2755| 46.51143375680581|
|        single|         yes|              1912| 32.22907949790795|
+--------------+------------+------------------+------------------+
```

QUESTION 8: Do feature engineering for column—age and find right age effect on campaign

```
scala> new_df.show
+----------+-----------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
|       age|        job| marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|  y|
+----------+-----------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
|       Old|  management| married| tertiary|     no|   2143|    yes|  no|unknown|  5|  may|     261|       1|   -1|       0| unknown| no|
|Middle Aged|  technician|  single|secondary|     no|     29|    yes|  no|unknown|  5|  may|     151|       1|   -1|       0| unknown| no|
|Middle Aged|entrepreneur| married|secondary|     no|      2|    yes| yes|unknown|  5|  may|      76|       1|   -1|       0| unknown| no|
|Middle Aged| blue-collar| married|  unknown|     no|   1506|    yes|  no|unknown|  5|  may|      92|       1|   -1|       0| unknown| no|
|Middle Aged|     unknown|  single|  unknown|     no|      1|     no|  no|unknown|  5|  may|     198|       1|   -1|       0| unknown| no|
|Middle Aged|  management| married| tertiary|     no|    231|    yes|  no|unknown|  5|  may|     139|       1|   -1|       0| unknown| no|
|      Young|  management|  single| tertiary|     no|    447|    yes| yes|unknown|  5|  may|     217|       1|   -1|       0| unknown| no|
|Middle Aged|entrepreneur|divorced| tertiary|    yes|      2|    yes|  no|unknown|  5|  may|     380|       1|   -1|       0| unknown| no|
|       Old|     retired| married|  primary|     no|    121|    yes|  no|unknown|  5|  may|      50|       1|   -1|       0| unknown| no|
|Middle Aged|  technician|  single|secondary|     no|    593|    yes|  no|unknown|  5|  may|      55|       1|   -1|       0| unknown| no|
|Middle Aged|      admin.|divorced|secondary|     no|    270|    yes|  no|unknown|  5|  may|     222|       1|   -1|       0| unknown| no|
|      Young|      admin.|  single|secondary|     no|    390|    yes|  no|unknown|  5|  may|     137|       1|   -1|       0| unknown| no|
|Middle Aged|  technician| married|secondary|     no|      6|    yes|  no|unknown|  5|  may|     517|       1|   -1|       0| unknown| no|
|       Old|  technician| married|  unknown|     no|     71|    yes|  no|unknown|  5|  may|      71|       1|   -1|       0| unknown| no|
|       Old|    services| married|secondary|     no|    162|    yes|  no|unknown|  5|  may|     174|       1|   -1|       0| unknown| no|
|Middle Aged|     retired| married|  primary|     no|    229|    yes|  no|unknown|  5|  may|     353|       1|   -1|       0| unknown| no|
|Middle Aged|      admin.|  single|  unknown|     no|     13|    yes|  no|unknown|  5|  may|      98|       1|   -1|       0| unknown| no|
|       Old| blue-collar| married|  primary|     no|     52|    yes|  no|unknown|  5|  may|      38|       1|   -1|       0| unknown| no|
|       Old|     retired| married|  primary|     no|     60|    yes|  no|unknown|  5|  may|     219|       1|   -1|       0| unknown| no|
|Middle Aged|    services| married|secondary|     no|      0|    yes|  no|unknown|  5|  may|      54|       1|   -1|       0| unknown| no|
+----------+-----------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+
only showing top 20 rows


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

scala> sqlContext.sql("select age, count(y) from Age_Transformed_Table group by age").show()
+----------+--------+
|       age|count(y)|
+----------+--------+
|Middle Aged|   29200|
|      Teen|      47|
|       Old|    4950|
|     Young|   11014|
+----------+--------+


scala> sqlContext.sql("select age, count(y) from Age_Transformed_Table group by age, y order by y").show()
+----------+--------+
|       age|count(y)|
+----------+--------+
|Middle Aged|   26389|
|      Teen|      29|
|     Young|    9475|
|       Old|    4029|
|     Young|    1539|
|Middle Aged|    2811|
|       Old|     921|
|      Teen|      18|
+----------+--------+
```

```
scala> (new_df.filter(new_df("age") === "Teen" && new_df("y") === "yes").count()).toFloat / (new_df.filter(new_df("age") === "Teen").count())
res27: Float = 0.38297874

scala> (new_df.filter(new_df("age") === "Old" && new_df("y") === "yes").count()).toFloat / (new_df.filter(new_df("age") === "Old").count())
res28: Float = 0.1860606

scala> (new_df.filter(new_df("age") === "Young" && new_df("y") === "yes").count()).toFloat / (new_df.filter(new_df("age") === "Young").count())
res29: Float = 0.13973126

scala> (new_df.filter(new_df("age") === "Middle Aged" && new_df("y") === "yes").count()).toFloat / (new_df.filter(new_df("age") === "Middle Aged").count())
res30: Float = 0.09626713

scala> Project #1 Completed by Christina Grys
```