JORDI PALOMA

Market Analysis in Banking Domain (Simplilearn BDHS Project 3)

DESCRIPTION

Background and Objective:

Your client, a 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, in order to assess if they would want to subscribe to the bank term deposit or not. You have to perform the marketing analysis of the data generated by this campaign.

Domain: Banking (Market Analysis)

Dataset Description

The data fields are as follows:

1.	age	numeric
2.	job	type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3.	marital	marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)
4.	education	(categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5.	default	has credit in default? (categorical: 'no', 'yes', 'unknown')
6.	housing:	has housing loan? (categorical: 'no', 'yes', 'unknown')
7.	loan	has a personal loan? (categorical: 'no', 'yes', 'unknown')

related to the last contact of the current campaign:

8.	contact	contact communication type (categorical: 'cellular', 'telephone')
9.	month	Month of last contact (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10.	day_of_week	last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11.	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.

other attributes:

12.	campaign	number of times a customer was contacted during the campaign (numeric, includes last contact)
13.	pdays:	number of days passed after the customer was last contacted from a previous campaign (numeric; 999 means customer was not previously contacted)
14.	previous	number of times the customer was contacted prior to (or before) this campaign (numeric)
15.	poutcome	outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')

#Output variable (desired target):

16 y has the customer subscribed a term deposit? (binary: 'yes', 'no')

Download the Sample Dataset.

Analysis tasks to be done:

The data size is huge and the marketing team has asked you to perform the below analysis-

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

1) Load data and create a Spark data frame.

```
val bank_df = spark.read.option("multiline",
"true").json("/user/jordipalomagmail/banking_code.json");
bank_df.show()
```

```
[jordipalomagmail@ip-10-0-31-151 ~]$ spark-shell
Setting default log level to "ERROR".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/11/14 16:29:27 WARN cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: Attempted to request executors before the AM has registered!
21/11/14 16:29:27 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_1636095248675_2505).
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.

scala> val bank_df = spark.read.option("multiline", "true").json("/user/jordipalomagmail/banking_code.json");
21/11/14 16:29:59 WARN lineage.LineageWriter: Lineage directory /var/log/spark/lineage doesn't exist or is not writable. Lineage for this application will be disabled.
bank_df: org.apache.spark.sql.DataFrame = [age: bigint, balance: bigint ... 15 more fields]

scala> bank_df.show()
+---+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
|age|balance|campaign|contact|day|default|duration|education|housing|         job|loan| marital|month|pdays|poutcome|previous|  y|
+---+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
| 58|   2143|       1|unknown|  5|     no|     261| tertiary|    yes|  management|  no| married|  may|   -1| unknown|       0| no|
| 44|     29|       1|unknown|  5|     no|     151|secondary|    yes|  technician|  no|  single|  may|   -1| unknown|       0| no|
| 33|      2|       1|unknown|  5|     no|      76|secondary|    yes|entrepreneur| yes| married|  may|   -1| unknown|       0| no|
| 47|   1506|       1|unknown|  5|     no|      92|  unknown|    yes| blue-collar|  no| married|  may|   -1| unknown|       0| no|
| 33|      1|       1|unknown|  5|     no|     198|  unknown|     no|     unknown|  no|  single|  may|   -1| unknown|       0| no|
| 35|    231|       1|unknown|  5|     no|     139| tertiary|    yes|  management|  no| married|  may|   -1| unknown|       0| no|
| 28|    447|       1|unknown|  5|     no|     217| tertiary|    yes|  management| yes|  single|  may|   -1| unknown|       0| no|
| 42|      2|       1|unknown|  5|    yes|     380| tertiary|    yes|entrepreneur|  no|divorced|  may|   -1| unknown|       0| no|
| 58|    121|       1|unknown|  5|     no|      50|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
| 43|    593|       1|unknown|  5|     no|      55|secondary|    yes|  technician|  no|  single|  may|   -1| unknown|       0| no|
| 41|    270|       1|unknown|  5|     no|     222|secondary|    yes|      admin.|  no|divorced|  may|   -1| unknown|       0| no|
| 29|    390|       1|unknown|  5|     no|     137|secondary|    yes|      admin.|  no|  single|  may|   -1| unknown|       0| no|
| 53|      6|       1|unknown|  5|     no|     517|secondary|    yes|  technician|  no| married|  may|   -1| unknown|       0| no|
| 58|     71|       1|unknown|  5|     no|      71|  unknown|    yes|  technician|  no| married|  may|   -1| unknown|       0| no|
| 57|    162|       1|unknown|  5|     no|     174|secondary|    yes|    services|  no| married|  may|   -1| unknown|       0| no|
| 51|    229|       1|unknown|  5|     no|     353|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
| 45|     13|       1|unknown|  5|     no|      98|  unknown|    yes|      admin.|  no|  single|  may|   -1| unknown|       0| no|
| 57|     52|       1|unknown|  5|     no|      38|  primary|    yes| blue-collar|  no| married|  may|   -1| unknown|       0| no|
| 60|     60|       1|unknown|  5|     no|     219|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
| 33|      0|       1|unknown|  5|     no|      54|secondary|    yes|    services|  no| married|  may|   -1| unknown|       0| no|
+---+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
only showing top 20 rows
```

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

```
bank_df.registerTempTable("bankTable");
```

```
val num_customers = spark.sql("select count(*) from bankTable").show()
```

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

scala> val num_customers = spark.sql("select count(*) from bankTable");
num_customers: org.apache.spark.sql.DataFrame = [count(1): bigint]

scala> val num_customers = spark.sql("select count(*) from bankTable").show()
+--------+
|count(1)|
+--------+
|   45211|
+--------+

num_customers: Unit = ()
```

```
val num_subscribers = spark.sql("select count(*) from bankTable where y='yes'");
num_subscribers.show()
```

```
scala> val num_subscribers = spark.sql("select count(*) from bankTable where y='yes'");
num_subscribers: org.apache.spark.sql.DataFrame = [count(1): bigint]

scala> num_subscribers.show()
+--------+
|count(1)|
+--------+
|    5289|
+--------+
```

```
scala> (5289.0/45211)*100
res7: Double = 11.698480458295547
```

Marketing success rate: `num_subscribers/num_customers = 5289.0/45211.0 = 11.69%`

a) Give marketing failure rate (No. of people not subscribed / total no. of entries).

```
val num_nosubscribers = spark.sql("select count(*) from bankTable where y='no'");
num_nosubscribers.show()
```

```
scala> val num_nosubscribers = spark.sql("select count(*) from bankTable where y='no'");
num_nosubscribers: org.apache.spark.sql.DataFrame = [count(1): bigint]

scala> num_nosubscribers.show()
+--------+
|count(1)|
+--------+
|   39922|
+--------+
```

```
scala> (39922.0/45211)*100
res9: Double = 88.30151954170445
```

Marketing failure rate: `num_nosubscribers/num_customers = 39922.0/45211.0 = 88.30%`

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

```
bank_df.select(max($"age")).show()
bank_df.select(min($"age")).show()
bank_df.select(avg($"age")).show()
```

or

```
bank_df.select("age").summary().show()
```

```
scala> bank_df.select("age").summary().show()
+-------+------------------+
|summary|               age|
+-------+------------------+
|  count|             45211|
|   mean| 40.93621021432837|
| stddev|10.618762040975405|
|    min|                18|
|    25%|                33|
|    50%|                39|
|    75%|                48|
|    max|                95|
+-------+------------------+
```

Max age: 95
Min age: 18
Average age: 40.96

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

```
bank_df.select("balance").summary().show()
```

```
scala> bank_df.select("balance").summary().show()
+-------+------------------+
|summary|           balance|
+-------+------------------+
|  count|             45211|
|   mean|1362.2720576850766|
| stddev|3044.7658291685257|
|    min|             -8019|
|    25%|                72|
|    50%|               448|
|    75%|              1427|
|    max|            102127|
+-------+------------------+
```

The average deposit is 1362.27$
The median deposit is 448$

5) Check if age matters in marketing subscription to deposit

```
val customers_by_age = spark.sql("select age, count(*) as number from bankTable where
y='yes' group by age order by number desc").show()
```

```
scala> val customers_by_age = spark.sql("select age, count(*) as number from bankTable where y='yes' group by age order by number desc").show()
+---+------+
|age|number|
+---+------+
| 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

customers_by_age: Unit = ()
```

Customers on their 30's have the highest number of subscriptions.

6) Check if marital status mattered for a subscription to deposit.

```
val customers_by_marital = spark.sql("select marital, count(*) as number from
bankTable where y='yes' group by marital order by number desc").show()
```

```
scala> val customers_by_marital = spark.sql("select marital, count(*) as number from bankTable where y='yes' group by marital order by number desc").show
+--------+------+
| marital|number|
+--------+------+
| married|  2755|
|  single|  1912|
|divorced|   622|
+--------+------+

customers_by_marital: Unit = ()
```

Married customers make more deposits than other groups.

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

```
val customers_by_agemarital = spark.sql("select age, marital, count(*) as number from
bankTable where y='yes' group by age, marital order by number desc").show()
```

```
scala> val customers_by_agemarital = spark.sql("select age, marital, count(*) as number from bankTable where y='yes' group by age, marital order by number desc").show()
+---+-------+------+
|age|marital|number|
+---+-------+------+
| 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| single|    97|
| 33|married|    97|
| 39|married|    87|
| 32|married|    87|
| 38|married|    86|
| 35| single|    84|
| 47|married|    83|
| 46|married|    80|
+---+-------+------+
only showing top 20 rows

customers_by_agemarital: Unit = ()
```

The combination of age and marital status shows that the highest amounts of deposits happen with single people in their 30's.

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

```
val age_levels = spark.udf.register("age_levels", (age:Int) => {
if (age <= 20)
"Teen"
else if (age > 20 && age <= 29)
"Young_adult"
else if (age > 29 && age <= 39)
"Adult"
else if (age > 39 && age < 49)
"Older_adult"
else if (age > 49 && age < 60)
"Young_senior"
else
"Senior"
})
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val age_levels = spark.udf.register("age_levels", (age:Int) => {
if (age <= 20)
"Teen"
else if (age > 20 && age <= 29)
"Young_adult"
else if (age > 29 && age <= 39)
"Adult"
else if (age > 39 && age < 49)
"Older_adult"
else if (age > 49 && age < 60)
"Young_senior"
else
"Senior"
})


// Exiting paste mode, now interpreting.

age_levels: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(IntegerType)))
```

```
val bank_df2 = bank_df.withColumn("age", age_levels(bank_df("age")))
bank_df2.show()
bank_df2.registerTempTable("bankTable2");
```

```
scala> bank_df2.registerTempTable("bankTable2");
warning: there was one deprecation warning; re-run with -deprecation for details

scala> val bank_df2 = bank_df.withColumn("age", age_levels(bank_df("age")))
bank_df2: org.apache.spark.sql.DataFrame = [age: string, balance: bigint ... 15 more fields]

scala> bank_df2.show()
+------------+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
|         age|balance|campaign|contact|day|default|duration|education|housing|         job|loan| marital|month|pdays|poutcome|previous|  y|
+------------+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
|Young_senior|   2143|       1|unknown|  5|     no|     261| tertiary|    yes|  management|  no| married|  may|   -1| unknown|       0| no|
| Older_adult|     29|       1|unknown|  5|     no|     151|secondary|    yes|  technician|  no|  single|  may|   -1| unknown|       0| no|
|       Adult|      2|       1|unknown|  5|     no|      76|secondary|    yes|entrepreneur| yes| married|  may|   -1| unknown|       0| no|
| Older_adult|   1506|       1|unknown|  5|     no|      92|  unknown|    yes| blue-collar|  no| married|  may|   -1| unknown|       0| no|
|       Adult|      1|       1|unknown|  5|     no|     198|  unknown|     no|     unknown|  no|  single|  may|   -1| unknown|       0| no|
|       Adult|    231|       1|unknown|  5|     no|     139| tertiary|    yes|  management|  no| married|  may|   -1| unknown|       0| no|
| Young_adult|    447|       1|unknown|  5|     no|     217| tertiary|    yes|  management| yes|  single|  may|   -1| unknown|       0| no|
| Older_adult|      2|       1|unknown|  5|    yes|     380| tertiary|    yes|entrepreneur|  no|divorced|  may|   -1| unknown|       0| no|
|Young_senior|    121|       1|unknown|  5|     no|      50|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
| Older_adult|    593|       1|unknown|  5|     no|      55|secondary|    yes|  technician|  no|  single|  may|   -1| unknown|       0| no|
| Older_adult|    270|       1|unknown|  5|     no|     222|secondary|    yes|      admin.|  no|divorced|  may|   -1| unknown|       0| no|
| Young_adult|    390|       1|unknown|  5|     no|     137|secondary|    yes|      admin.|  no|  single|  may|   -1| unknown|       0| no|
|Young_senior|      6|       1|unknown|  5|     no|     517|secondary|    yes|  technician|  no| married|  may|   -1| unknown|       0| no|
|Young_senior|     71|       1|unknown|  5|     no|      71|  unknown|    yes|  technician|  no| married|  may|   -1| unknown|       0| no|
|Young_senior|    162|       1|unknown|  5|     no|     174|secondary|    yes|    services|  no| married|  may|   -1| unknown|       0| no|
|Young_senior|    229|       1|unknown|  5|     no|     353|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
| Older_adult|     13|       1|unknown|  5|     no|      98|  unknown|    yes|      admin.|  no|  single|  may|   -1| unknown|       0| no|
|Young_senior|     52|       1|unknown|  5|     no|      38|  primary|    yes| blue-collar|  no| married|  may|   -1| unknown|       0| no|
|      Senior|     60|       1|unknown|  5|     no|     219|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
|       Adult|      0|       1|unknown|  5|     no|      54|secondary|    yes|    services|  no| married|  may|   -1| unknown|       0| no|
+------------+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
only showing top 20 rows
```

As predicted before, the age group with most deposits belongs to the "Adult" group, that is, people between 30 and 39 years old.