Project3 - Market Analysis in Banking Domain

Author - Arka Prava Panda

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.

Q1> Load data and create a Spark data frame

At the first step we are uploading the csv in test format and partitioning it into 15 partitions

```
val bank_data = sc.textFile("FileStore/Project_1_dataset_bank_full__2_-1.csv",15)
```

```
bank_data: org.apache.spark.rdd.RDD[String] = FileStore/Project_1_dataset_bank_full__2_-1.csv M
apPartitionsRDD[1] at textFile at command-2758534730523416:1
```

```
bank_data.take(2)
```

```
res0: Array[String] = Array("age;""job"";""marital"";""education"";""default"";""balance"";""ho
using"";""loan"";""contact"";""day"";""month"";""duration"";""campaign"";""pdays"";""previou
s"";""poutcome"";""y""", "58;""management"";""married"";""tertiary"";""no"";2143;""yes"";""n
o"";""unknown"";5;""may"";261;1;-1;0;""unknown"";""no""")
```

```
val bank_data1 = bank_data.map(x => x.replace("\"",""))
```

```
bank_data1: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at map at command-2758534730
523418:1
```

```
val header = bank_data1.first()
```

```
header: String = age;job;marital;education;default;balance;housing;loan;contact;day;month;durat
ion;campaign;pdays;previous;poutcome;y
```

```
val bank_data2 = bank_data1.filter(x => x != header)

bank_data2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at filter at command-2861593
367221660:1
```

```
bank_data2.take(2)

res1: Array[String] = Array(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;n
o)
```

```
val bank_data3 = bank_data2.toDF()

bank_data3: org.apache.spark.sql.DataFrame = [value: string]
```

Now we are writing this file as a csv in hdfs

```
bank_data3.write.csv("bankcsv3.csv")

  AnalysisException: path dbfs:/bankcsv3.csv already exists.
```

Then we are again loading this csv designating proper syntax

```
val bank_data4 = spark.read.format("csv").option("delimiter",";").load("/bankcsv3.csv")

bank_data4: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 15 more fields]
```

Assigning the headers manually and saving it into a file as spark dataframe

```
val bank_data5 =
bank_data4.toDF("age","job","marital","education","default","balance","housing","loan","contact
","day","month","duration","campaign","pdays","previous","poutcome","y")

bank_data5: org.apache.spark.sql.DataFrame = [age: string, job: string ... 15 more fields]
```

Checking the the newly created dataframe if everything is in order

```
bank_data5.show(10)

+---+------------+--------+---------+-------+-------+-------+----+---------+---+-----+--------+
--------+-----+--------+--------+---+
|age|         job| marital|education|default|balance|housing|loan|  contact|day|month|duration|
campaign|pdays|previous|poutcome|  y|
+---+------------+--------+---------+-------+-------+-------+----+---------+---+-----+--------+
--------+-----+--------+--------+---+
| 43| blue-collar|divorced|  primary|     no|    219|     no|  no|telephone| 13|  nov|     138|
1|   -1|       0| unknown| no|
| 33|      admin.|  single|secondary|     no|    206|     no|  no|telephone| 13|  nov|     168|
1|   -1|       0| unknown| no|
| 40| blue-collar| married|secondary|     no|    718|    yes| yes|telephone| 13|  nov|       72|
1|   -1|       0| unknown| no|
| 59|   housemaid| married| tertiary|     no|      0|     no|  no|telephone| 13|  nov|      494|
1|   -1|       0| unknown|yes|
| 30|  management|  single| tertiary|     no|   1243|    yes|  no|telephone| 13|  nov|       86|
1|  174|       1| failure| no|
| 32|  management| married| tertiary|     no|     62|    yes|  no|telephone| 13|  nov|       45|
1|   -1|       0| unknown| no|
| 32| blue-collar|  single| tertiary|     no|    700|     no|  no|  unknown| 13|  nov|       90|
1|   -1|       0| unknown|yes|
```

Q2> Give marketing success rate (No. of people subscribed / total no. of entries) and Give marketing failure rate

Creating a temporary view in order to use it with SparkSQL

```
bank_data5.createOrReplaceTempView("bankdf")
```

Since the subscription status is in string format I am using a case option in the below sql query to take the yes as 1 and the no as 0 and also the data has no null values, also I have rounded off the result to 2 decimal places

```
spark.sql(""" select round(sum(sub)*100/count(sub),2) as success_rate from (select *, case when
y = "yes" then 1 else 0 end as sub from bankdf) bnkdf """).show
```

```
+------------+
|success_rate|
+------------+
|        11.7|
+------------+
```

In the above query one can see that the success rate of subscription is 11.69 % and hence the failure rate will be 100-11.69 ie 88.31%

Q3>Give the maximum, mean, and minimum age of the average targeted customer

```
spark.sql("""select max(age) MAX_AGE,round(mean(age),2) AVG_AGE,min(age) MIN_AGE from
bankdf""").show()
```

```
+-------+-------+-------+
|MAX_AGE|AVG_AGE|MIN_AGE|
+-------+-------+-------+
|     95|  40.94|     18|
+-------+------+-------+
```

from the above output we can say that the maximum age, average age and minimum age is 95 years, 40.94 years and 18 years respectively

Q4> Check the quality of customers by checking average balance, median balance of customers

```
spark.sql("""select round(mean(balance),2) as AVG_BALANCE, percentile_approx(balance, 0.5) as
MEDIAN_BALANCE from bankdf""").show
```

```
+-----------+--------------+
|AVG_BALANCE|MEDIAN_BALANCE|
+-----------+--------------+
|    1362.27|         448.0|
+-----------+--------------+
```

From the above query we can observe that the average balance is 1362.27 and the median balance is 448.0

Q5> Check if age matters in marketing subscription for deposit

Now we will create a table which will classify the age into groups

```
val case_table_tot = spark.sql("""select case
when age>=18 and age<=30 then '18-30'
when age>=31 and age<=40 then '31-40'
when age>=41 and age<=50 then '41-50'
when age>=51 and age<=60 then '51-60'
when age>=61 and age<=70 then '61-70'
when age>=71 and age<=80 then '71-80'
when age>=81 and age<=95 then '81-95'
end as age_category_yrs,
count(*) as total_counts
from bankdf
group by
age_category_yrs
order by age_category_yrs""")
```

```
case_table_tot: org.apache.spark.sql.DataFrame = [age_category_yrs: string, total_counts: bigin
t]
```

```
/*checking the table*/
case_table_tot.show
```

```
+----------------+------------+
|age_category_yrs|total_counts|
+----------------+------------+
|           18-30|        7030|
|           31-40|       17687|
|           41-50|       11239|
|           51-60|        8067|
|           61-70|         701|
|           71-80|         388|
|           81-95|          99|
+----------------+------------+
```

now we will create a output which will show how many people subscribed per age category

```
val case_table_sub = spark.sql("""select case
when age>=18 and age<=30 then '18-30'
when age>=31 and age<=40 then '31-40'
when age>=41 and age<=50 then '41-50'
when age>=51 and age<=60 then '51-60'
when age>=61 and age<=70 then '61-70'
when age>=71 and age<=80 then '71-80'
when age>=81 and age<=95 then '81-95'
end as age_category_yrs,
count(*) as total_sub_counts
from bankdf
where y = 'yes'
group by
age_category_yrs
order by age_category_yrs""")
```

```
case_table_sub: org.apache.spark.sql.DataFrame = [age_category_yrs: string, total_sub_counts: b
igint]
```

```
case_table_sub.show
```

```
+----------------+----------------+
|age_category_yrs|total_sub_counts|
+----------------+----------------+
|           18-30|            1145|
|           31-40|            1812|
|           41-50|            1019|
|           51-60|             811|
|           61-70|             284|
|           71-80|             175|
|           81-95|              43|
+----------------+----------------+
```

As the table has been created and stored in a variable called case_table we will create a temporary view out of it in order to query it with spark sql

```
case_table_tot.createOrReplaceTempView("case_table_total_df")
case_table_sub.createOrReplaceTempView("case_table_sub_df")
```

Now we will join the two tables and compare their subscription rates

```
spark.sql("""select s.age_category_yrs,s.total_sub_counts,t.total_counts,
round(s.total_sub_counts*100/t.total_counts,2) as per_sum
from case_table_total_df t
inner join case_table_sub_df s on
t.age_category_yrs = s.age_category_yrs
order by t.age_category_yrs""").show

+----------------+---------------+------------+-------+
|age_category_yrs|total_sub_counts|total_counts|per_sum|
+----------------+---------------+------------+-------+
|           18-30|           1145|        7030|  16.29|
|           31-40|           1812|       17687|  10.24|
|           41-50|           1019|       11239|   9.07|
|           51-60|            811|        8067|  10.05|
|           61-70|            284|         701|  40.51|
|           71-80|            175|         388|   45.1|
|           81-95|             43|          99|  43.43|
+----------------+---------------+------------+-------+
```

from the above table we can observe that people in the age range of 61-95 yrs has a higher subscription rate but lower number of total subscriptions

Now we will query what is the age_category and how much percentage of the total subscription it contributes to

```
spark.sql("""select *,
round(sum(total_sub_counts) over(order by age_category_yrs) * 100/sum(total_sub_counts) over
(),2) cumsum
from case_table_sub_df""").show

+----------------+---------------+------+
|age_category_yrs|total_sub_counts|cumsum|
+----------------+---------------+------+
|           18-30|           1145| 21.65|
|           31-40|           1812| 55.91|
|           41-50|           1019| 75.17|
|           51-60|            811| 90.51|
|           61-70|            284| 95.88|
|           71-80|            175| 99.19|
|           81-95|             43| 100.0|
+----------------+---------------+------+
```

From the above table we can see the rolling percentage sum in the per_sum column and can also infer the following

1) The age_categories of 18-60 years contributes to over 90% of the successful subscription

2) Also the age_categories of 18-50 years contribute to over 75% of the successful subscription

3) Finally the age_categories of 18-40 years contribute to over 55% of the successful subscription

Q5> Check if marital status mattered for a subscription to deposit

```
spark.sql("""with subs as (select marital, count(*) tot_sub_count from bankdf where y = 'yes'
group by marital),
tot as (select marital, count(*) tot_count from bankdf group by marital)
select subs.marital, subs.tot_sub_count, tot.tot_count,
round(subs.tot_sub_count*100/tot.tot_count,2) per_sub
from tot
inner join subs on
subs.marital = tot.marital""").show
```

```
+--------+-------------+---------+-------+
| marital|tot_sub_count|tot_count|per_sub|
+--------+-------------+---------+-------+
|divorced|          622|     5207|  11.95|
| married|         2755|    27214|  10.12|
|  single|         1912|    12790|  14.95|
+--------+-------------+---------+-------+
```

As from the above query we can see total number respondents who subscribed, total count of the respondents and the percentage of them who subscribed

We can also infer the following

There is no significant difference among the percentage of respondents who subscribed across all the marital statuses although the subscription rate of the marital status: single is a little higher

Also among all the subscribers the single and the married respondents contribute to the major part of the subscription

Q6> Check if age and marital status together mattered for a subscription to deposit scheme

```
val age_cat_table = spark.sql("""select*, case
when age>=18 and age<=30 then '18-30'
when age>=31 and age<=40 then '31-40'
when age>=41 and age<=50 then '41-50'
when age>=51 and age<=60 then '51-60'
when age>=61 and age<=70 then '61-70'
when age>=71 and age<=80 then '71-80'
when age>=81 and age<=95 then '81-95'
end as age_category_yrs
from bankdf
order by age_category_yrs""").show(10)

+---+-------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--
------+-----+--------+--------+---+----------------+
|age|          job|marital|education|default|balance|housing|loan|contact|day|month|duration|ca
mpaign|pdays|previous|poutcome|  y|age_category_yrs|
+---+-------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--
------+-----+--------+--------+---+----------------+
| 29|  blue-collar|married|secondary|     no|    185|    yes| yes|unknown|  5|  jun|     471|
3|   -1|       0| unknown| no|           18-30|
| 26|     services| single|secondary|     no|    512|    yes| yes|unknown|  5|  jun|      75|
3|   -1|       0| unknown| no|           18-30|
| 29|self-employed|married|secondary|     no|    425|    yes|  no|unknown|  5|  jun|     562|
15|   -1|       0| unknown|yes|           18-30|
| 28|       admin.| single|secondary|     no|     25|    yes|  no|unknown|  5|  jun|     109|
2|   -1|       0| unknown| no|           18-30|
| 30| entrepreneur| single| tertiary|     no|    734|    yes|  no|unknown|  5|  jun|     113|
3|   -1|       0| unknown| no|           18-30|
| 30|   technician|married|secondary|    yes|  -1019|    yes|  no|unknown|  5|  jun|     955|
4|   -1|       0| unknown| no|           18-30|
| 29|   technician| single|secondary|     no|   3313|    yes|  no|unknown|  5|  jun|      18|
3|   -1|       0| unknown| no|           18-30|
| 30|     services|married|secondary|     no|    311|    yes| yes|unknown|  5|  jun|      94|

age_cat_table.createOrReplaceTempView("age_cat_table_df")
```

```
val age_marriage = spark.sql("""select '18-30' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '18-30' group by marital
union
select '31-40' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '31-40' group by marital
union
select '41-50' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '41-50' group by marital
union
select '51-60' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '51-60' group by marital
union
select '61-70' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '61-70' group by marital
union
select '71-80' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '71-80' group by marital
union
select '81-95' age_cat,marital,count(y) from age_cat_table_df
where y='yes' and age_category_yrs = '81-95' group by marital
""")
```

```
age_marriage: org.apache.spark.sql.DataFrame = [age_cat: string, marital: string ... 1 more fie
ld]
```

```
age_marriage.createOrReplaceTempView("age_marriage_df")
```

```
spark.sql("""select *  from age_marriage_df where marital='single'""").show
```

```
+-------+-------+--------+
|age_cat|marital|count(y)|
+-------+-------+--------+
|  18-30| single|     945|
|  31-40| single|     722|
|  41-50| single|     183|
|  51-60| single|      53|
|  61-70| single|       6|
|  71-80| single|       1|
|  81-95| single|       2|
+-------+-------+--------+
```

```
spark.sql("""select * from age_marriage_df where marital='divorced'""").show
```

```
+-------+--------+--------+
|age_cat| marital|count(y)|
+-------+--------+--------+
|  18-30|divorced|      18|
|  31-40|divorced|     163|
|  41-50|divorced|     174|
|  51-60|divorced|     170|
|  61-70|divorced|      47|
|  71-80|divorced|      36|
|  81-95|divorced|      14|
+-------+--------+--------+
spark.sql("""select * from age_marriage_df where marital='married'""").show
```

```
+-------+-------+--------+
|age_cat|marital|count(y)|
+-------+-------+--------+
|  18-30|married|     182|
|  31-40|married|     927|
|  41-50|married|     662|
|  51-60|married|     588|
|  61-70|married|     231|
|  71-80|married|     138|
|  81-95|married|      27|
+-------+-------+--------+
```

The above three queries show the following:

age category (18-30 yrs) followed by (31-40 yrs) and (41-50 yrs) in the 'single' marital category contributed to the major subscription

age category (31-40 yrs) followd by (41-50 yrs) and (51-60 yrs) in the 'married' marital categrory contribute to the major subscription

subscriptions in the 'divorced' marital status is significantly low than the 'single' and 'married' categories.

We can say that age along with marital status mattered for a subscription to deposit scheme.

Q7> Do feature engineering for the bank and find the right age effect on the campaign

```
spark.sql("""select *, case
when age>=18 and age<=30 then '18-30'
when age>=31 and age<=40 then '31-40'
when age>=41 and age<=50 then '41-50'
when age>=51 and age<=60 then '51-60'
when age>=61 and age<=70 then '61-70'
when age>=71 and age<=80 then '71-80'
when age>=81 and age<=95 then '81-95'
end as age_category_yrs
from bankdf
order by age_category_yrs""").show(10)
```

```
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--
------+-----+--------+--------+---+----------------+
|age|         job|marital|education|default|balance|housing|loan|contact|day|month|duration|ca
mpaign|pdays|previous|poutcome|  y|age_category_yrs|
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--
------+-----+--------+--------+---+----------------+
| 29|  blue-collar|married|secondary|     no|    185|    yes| yes|unknown|  5|  jun|     471|
3|   -1|       0| unknown| no|           18-30|
| 26|     services| single|secondary|     no|    512|    yes| yes|unknown|  5|  jun|      75|
3|   -1|       0| unknown| no|          18-30|
| 29|self-employed|married|secondary|     no|    425|    yes|  no|unknown|  5|  jun|     562|
15|   -1|       0| unknown|yes|          18-30|
| 28|       admin.| single|secondary|     no|     25|    yes|  no|unknown|  5|  jun|     109|
2|   -1|       0| unknown| no|          18-30|
| 30| entrepreneur| single| tertiary|     no|    734|    yes|  no|unknown|  5|  jun|     113|
3|   -1|       0| unknown| no|          18-30|
| 30|   technician|married|secondary|    yes|  -1019|    yes|  no|unknown|  5|  jun|     955|
4|   -1|       0| unknown| no|          18-30|
| 29|   technician| single|secondary|     no|   3313|    yes|  no|unknown|  5|  jun|      18|
3|   -1|       0| unknown| no|          18-30|
```

```
spark.sql("""select age_category_yrs, count(*) as subs from age_cat_table_df where y ='yes'
group by age_category_yrs order by age_category_yrs""").show
```

```
+----------------+----+
|age_category_yrs|subs|
+----------------+----+
|           18-30|1145|
|           31-40|1812|
|           41-50|1019|
|           51-60| 811|
|           61-70| 284|
|           71-80| 175|
|           81-95|  43|
+----------------+----+
```

We can see that the age_categories of (18-30 yrs),(31-40 yrs),(41-50 yrs),(51-60 yrs) consists of highest subscribers.

With highest subscriptions coming from the age_category of (31 - 40 yrs)