Apache Spark Marketing Analysis Project

Your client—a Portuguese banking institution—ran a marketing campaign to convince potential customers to invest in bank term deposit. Information related to direct marketing campaigns of the bank are as follows. The marketing campaigns were based on phone calls.

Author: Roshan Kumar Jha
Github: https://github.com/contactmeroshan
Linkedin: https://www.linkedin.com/in/roshan-kumar-jha-24891a219/

Load and Create Spark Data Frame

```
val df =
sqlContext.read.format("com.databricks.spark.csv").option("header","true").option("inferSchema","true").option("delimiter",";").load("/FileStor
e/tables/bank_full-bd3df.csv")
```

```
df.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|
| 33|     unknown|  single|  unknown|     no|      1|     no|  no|unknown|  5|  may|     198|       1|   -1|       0| unknown| no|
| 35|  management| married| tertiary|     no|    231|    yes|  no|unknown|  5|  may|     139|       1|   -1|       0| unknown| no|
| 28|  management|  single| tertiary|     no|    447|    yes| yes|unknown|  5|  may|     217|       1|   -1|       0| unknown| no|
| 42|entrepreneur|divorced| tertiary|    yes|      2|    yes|  no|unknown|  5|  may|     380|       1|   -1|       0| unknown| no|
| 58|     retired| married|  primary|     no|    121|    yes|  no|unknown|  5|  may|      50|       1|   -1|       0| unknown| no|
| 43|  technician|  single|secondary|     no|    593|    yes|  no|unknown|  5|  may|      55|       1|   -1|       0| unknown| no|
| 41|      admin.|divorced|secondary|     no|    270|    yes|  no|unknown|  5|  may|     222|       1|   -1|       0| unknown| no|
| 29|      admin.|  single|secondary|     no|    390|    yes|  no|unknown|  5|  may|     137|       1|   -1|       0| unknown| no|
| 53|  technician| married|secondary|     no|      6|    yes|  no|unknown|  5|  may|     517|       1|   -1|       0| unknown| no|
| 58|  technician| married|  unknown|     no|     71|    yes|  no|unknown|  5|  may|      71|       1|   -1|       0| unknown| no|
| 57|    services| married|secondary|     no|    162|    yes|  no|unknown|  5|  may|     174|       1|   -1|       0| unknown| no|
| 51|     retired| married|  primary|     no|    229|    yes|  no|unknown|  5|  may|     353|       1|   -1|       0| unknown| no|
| 45|      admin.|  single|  unknown|     no|     13|    yes|  no|unknown|  5|  may|      98|       1|   -1|       0| unknown| no|
| 57| blue-collar| married|  primary|     no|     52|    yes|  no|unknown|  5|  may|      38|       1|   -1|       0| unknown| no|
```

Marketing Success Rate

```
val suc = df.filter($"y" === "yes").count.toFloat / df.count.toFloat *100
```

```
suc: Float = 11.698481
```

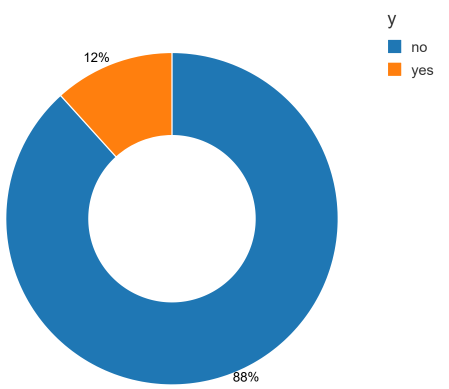
Marketing Fail Rate

```
val fail = df.filter($"y" === "no").count.toFloat / df.count.toFloat *100
```

```
fail: Float = 88.30152
```

Marketing success/failure Rate

```
display(df)
```



Max, Min, Min, age of average target customer

```
import org.apache.spark.sql.functions.{min, max, avg}

df.agg(max($"age"),min($"age"), avg($"age")).show()
```

```
+--------+--------+-----------------+
|max(age)|min(age)|         avg(age)|
+--------+--------+-----------------+
|      95|      18|40.93621021432837|
+--------+--------+-----------------+

import org.apache.spark.sql.functions.{min, max, avg}
```

The same descriptive statistics can also be computed with Spark SQL

```
import org.apache.commons.math3.stat.descriptive

df.createOrReplaceTempView("sample")
val med = sql("SELECT max(age) as max, min(age) as min, avg(age) as average, percentile_approx(age, 0.5) as median  FROM sample");

med.show()
```

```
+---+---+-----------------+------+
|max|min|          average|median|
+---+---+-----------------+------+
| 95| 18|40.93621021432837|    39|
+---+---+-----------------+------+

import org.apache.commons.math3.stat.descriptive
med: org.apache.spark.sql.DataFrame = [max: int, min: int ... 2 more fields]
```

Quality of clients by checking average balance, median balance of clients

```
val medBal = sql("SELECT max(balance) as max, min(balance) as min, avg(balance) as average, percentile_approx(balance, 0.5) as median  FROM
sample");

medBal.show()
```

```
+------+-----+------------------+------+
|   max|  min|           average|median|
+------+-----+------------------+------+
|102127|-8019|1362.2720576850766|   448|
+------+-----+------------------+------+

medBal: org.apache.spark.sql.DataFrame = [max: int, min: int ... 2 more fields]
```

Did marital status mattered for subscription to deposit?

```
df.groupBy($"y".alias("Did the customer Subscribed")).agg(count($"marital").alias("Marital Count")).show
```

```
+---------------------------+-------------+
|Did the customer Subscribed|Marital Count|
+---------------------------+-------------+
|                         no|        39922|
|                        yes|         5289|
+---------------------------+-------------+
```

Did age and marital status together mattered for subscription to deposit scheme?

```
df.groupBy("marital","y").count.sort($"count").show
```

```
+--------+---+-----+
| marital|  y|count|
+--------+---+-----+
|divorced|yes|  622|
|  single|yes| 1912|
| married|yes| 2755|
|divorced| no| 4585|
|  single| no|10878|
| married| no|24459|
+--------+---+-----+
```

Feature engineering for age column and find right age effect on campaign

```
import org.apache.spark.sql.functions.udf

def ageToCategory = udf((age:Int) => {
      age match {
      case t if t < 30 => "young"
      case t if t > 65 => "Old"
      case _ => "mid"
      }
      }
     )

val newdf = df.withColumn("agecat",ageToCategory(df("age"))) // create newcolumn
newdf.groupBy("agecat","y").count().sort($"count".desc).show
```

```
+------+---+-----+
|agecat|  y|count|
+------+---+-----+
|   mid| no|35146|
| young| no| 4345|
|   mid|yes| 4041|
| young|yes|  928|
|   Old| no|  431|
|   Old|yes|  320|
+------+---+-----+

import org.apache.spark.sql.functions.udf
ageToCategory: org.apache.spark.sql.expressions.UserDefinedFunction
newdf: org.apache.spark.sql.DataFrame = [age: int, job: string ... 16 more fields]
```

The same feature engineering can also be done by using aggregate functions

```
val bank_f = df.filter($"y" === "yes")
val age_cat = bank_f.withColumn("age_cat", when($"age" < 25, "young").otherwise(when($"age" > 60, "old").otherwise("mid")))

val result = age_cat.groupBy("age_cat").count()
result.show()

val resultWithMarital= age_cat.groupBy("age_cat", "marital").count().sort($"count".desc)
resultWithMarital.show()
```

```
+-------+-----+
|age_cat|count|
+-------+-----+
|    mid| 4580|
|    old|  502|
|  young|  207|
+-------+-----+

+-------+--------+-----+
|age_cat| marital|count|
+-------+--------+-----+
|    mid| married| 2345|
|    mid|  single| 1710|
|    mid|divorced|  525|
|    old| married|  396|
|  young|  single|  193|
|    old|divorced|   97|
|  young| married|   14|
|    old|  single|    9|
+-------+--------+-----+
```

Correlation Analysis

Convert "y" (If the person subscribed after the marketing strategy) to dummy

```
import org.apache.spark.sql.functions._ // for `when`

// Dummy column will be called "bin"
val newDF = df.withColumn("bin", when($"y" === "yes" , 1).otherwise(0))

newDF.show(5)
```

```
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+---+
|age|         job|marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|  y|bin|
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+---+
| 58|  management|married| tertiary|     no|   2143|    yes|  no|unknown|  5|  may|     261|       1|   -1|       0| unknown| no|  0|
| 44|  technician| single|secondary|     no|     29|    yes|  no|unknown|  5|  may|     151|       1|   -1|       0| unknown| no|  0|
| 33|entrepreneur|married|secondary|     no|      2|    yes| yes|unknown|  5|  may|      76|       1|   -1|       0| unknown| no|  0|
```

```
| 47| blue-collar|married|  unknown|     no|   1506|    yes|  no|unknown|  5|  may|      92|       1|   -1|       0| unknown| no|  0|
| 33|     unknown| single|  unknown|     no|      1|     no|  no|unknown|  5|  may|     198|       1|   -1|       0| unknown| no|  0|
+---+------------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+---+
only showing top 5 rows

import org.apache.spark.sql.functions._
newDF: org.apache.spark.sql.DataFrame = [age: int, job: string ... 16 more fields]
```

Prepare Data for Correlation Analysis

```
import org.apache.spark.ml.linalg.{Matrix, Vectors}
import org.apache.spark.ml.stat.Correlation
import org.apache.spark.sql.Row
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.linalg.Vectors


val corrDF = newDF.select($"age",$"bin")

//There is no support for spearman correlation



val assembler = new VectorAssembler()
  .setInputCols(Array("age", "bin"))
  .setOutputCol("features")

val output = assembler.transform(corrDF)
```

```
import org.apache.spark.ml.linalg.{Matrix, Vectors}
import org.apache.spark.ml.stat.Correlation
import org.apache.spark.sql.Row
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.linalg.Vectors
corrDF: org.apache.spark.sql.DataFrame = [age: int, bin: int]
assembler: org.apache.spark.ml.feature.VectorAssembler = vecAssembler_22e4280e7633
output: org.apache.spark.sql.DataFrame = [age: int, bin: int ... 1 more field]
```

Correlation between People that subscribed to the bank and the age of the marke

```
println("Spearman Correlation: " + "\n", Correlation.corr(output, "features", "spearman").head())

println("Pearson Correlation: " + corrDF.stat.corr("age","bin"))
```

```
(Spearman Correlation:
,[1.0                    -0.008749987770931828
-0.008749987770931828  1.0                    ])
Pearson Correlation: 0.025155017088386994
```

KMeans Analysis

Kmeans Implementation

```
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.ml.feature.VectorAssembler


val featureco1 = Array("age","bin")
val assembler = new VectorAssembler().setInputCols(featureco1).setOutputCol("features")


val df2 = assembler.transform(newDF)
df2.show()
```

```
+---+------------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+---+--------
--+
|age|         job| marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|  y|bin|  featur
es|
+---+------------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+---+---+--------
--+
| 58|  management| married| tertiary|     no|   2143|    yes|  no|unknown|  5|  may|     261|       1|   -1|       0| unknown| no|  0|[58.0,0.
0]|
| 44|  technician|  single|secondary|     no|     29|    yes|  no|unknown|  5|  may|     151|       1|   -1|       0| unknown| no|  0|[44.0,0.
0]|
| 33|entrepreneur| married|secondary|     no|      2|    yes| yes|unknown|  5|  may|      76|       1|   -1|       0| unknown| no|  0|[33.0,0.
```

```
0]|
| 47| blue-collar| married|  unknown|     no|   1506|    yes|   no|unknown|  5|  may|      92|       1|   -1|       0| unknown| no|  0|[47.0,0.
0]|
| 33|     unknown|  single|  unknown|     no|      1|     no|   no|unknown|  5|  may|     198|       1|   -1|       0| unknown| no|  0|[33.0,0.
0]|
| 35|  management| married| tertiary|     no|    231|    yes|   no|unknown|  5|  may|     139|       1|   -1|       0| unknown| no|  0|[35.0,0.
0]|
| 28|  management|  single| tertiary|     no|    447|    yes|  yes|unknown|  5|  may|     217|       1|   -1|       0| unknown| no|  0|[28.0,0.
0]|
```

Train Test Split

```
val Array(trainData,testData) = df2.randomSplit(Array(.7, .3))

trainData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [age: int, job: string ... 17 more fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [age: int, job: string ... 17 more fields]
```

Train Test Split Description

```
println("Total rows of complete DF: " + df2.count() + "\n" +
"Train Data: " + trainData.count() + "\n" +
"Test Data: " + testData.count())

Total rows of complete DF: 45211
Train Data: 31650
Test Data: 13561
```

KMeans Algorithm

```
import org.apache.spark.ml.clustering.KMeans
//import org.apache.spark.mllib.linalg.Vectors

val kmeans = new KMeans().setK(20).setFeaturesCol("features").setMaxIter(3)

import org.apache.spark.ml.clustering.KMeans
kmeans: org.apache.spark.ml.clustering.KMeans = kmeans_e1fea8a89ab4
```

Fit KNN Model

```
//kmeansModel = KMeans.train(parsedData, numCluster, maxIter)


// This works with ml.Kmeans
val model = kmeans.fit(trainData)

model: org.apache.spark.ml.clustering.KMeansModel = kmeans_e1fea8a89ab4
```

Predict Test Data

```
val categories = model.transform(testData)

categories: org.apache.spark.sql.DataFrame = [age: int, job: string ... 18 more fields]

categories.show()
```

```
+---+-----------+-------+---------+-------+-------+-------+----+---------+---+-----+--------+--------+-----+--------+--------+---+---+--------
--+----------+
|age|        job|marital|education|default|balance|housing|loan|  contact|day|month|duration|campaign|pdays|previous|poutcome|  y|bin|  featur
es|prediction|
+---+-----------+-------+---------+-------+-------+-------+----+---------+---+-----+--------+--------+-----+--------+--------+---+---+--------
--+----------+
| 19|    student| single|secondary|     no|    626|     no|  no|telephone| 15|  apr|     117|       1|   -1|       0| unknown| no|  0|[19.0,0.
0]|         4|
| 19|    student| single|secondary|     no|   1803|     no|  no| cellular| 23|  jun|     124|       1|  105|       1| failure| no|  0|[19.0,0.
0]|         4|
| 19|    student| single|  unknown|     no|    779|     no|  no| cellular|  1|  apr|     184|       4|   -1|       0| unknown|yes|  1|[19.0,1.
0]|         4|
| 20|     admin.| single|secondary|     no|     66|    yes|  no|  unknown| 19|  jun|      75|       2|   -1|       0| unknown| no|  0|[20.0,0.
0]|         4|
| 20|blue-collar| single|secondary|     no|    129|    yes| yes|  unknown| 13|  may|     190|       1|   -1|       0| unknown| no|  0|[20.0,0.
0]|         4|
| 20|    student| single|  primary|     no|   6991|     no|  no| cellular| 12|  aug|     178|       2|   -1|       0| unknown|yes|  1|[20.0,1.
0]|         4|
| 20|    student| single|secondary|     no|   -322|    yes|  no|  unknown| 20|  jun|      73|       4|   -1|       0| unknown| no|  0|[20.0,0.
0]|         4|
| 20|    student| single|secondary|     no|    130|     no|  no|telephone| 11|  aug|      88|       1|   99|       3| failure| no|  0|[20.0,0.
```

Now we need can visualize each age per cluster

```
+---+--------+
|age|clusters|
+---+--------+
| 18|       4|
```

```
| 19|       4|
| 20|       4|
| 21|       4|
| 22|       4|
| 23|       4|
| 24|       4|
| 25|      11|
| 26|      11|
| 27|      11|
| 28|      17|
| 29|      17|
| 30|      17|
| 31|       0|
| 32|       0|
| 33|       0|
| 34|      19|
|   |        |
```