Market Analysis in Banking Domain

1. Load data and create a Spark data frame

```
[erabhishektiwari04gmail@ip-10-0-31-29 ~]$ spark-shell
Setting default log level to "ERROR".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/08/01 09:29:03 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_1622117371245_24686).
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>
```

Fig: Opening spark shell

- Importing libraries:

```
import scala.reflect.runtime.universe
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.feature.Bucketizer
import org.apache.spark.ml.feature.Normalizer
import org.apache.spark.ml.feature.StringIndexer
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.functions.mean
import org.apache.spark.sql.hive.HiveContext
```

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


import scala.reflect.runtime.universe
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.feature.Bucketizer
import org.apache.spark.ml.feature.Normalizer
import org.apache.spark.ml.feature.StringIndexer
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.functions.mean
import org.apache.spark.sql.hive.HiveContext
```

Fig: Importing libraries:

-Loading data in Spark data frame

```
val bank_people_data =
spark.read.option("multiline","true").json("/user/erabhishektiwari04gmail/simplilearn/
project3_bank.json");
bank_people_data.show()
```

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

val bank_people_data = spark.read.option("multiline","true").json("/user/erabhishektiwari04gmail/simplilearn/project3_bank.json");
bank_people_data.show()

// Exiting paste mode, now interpreting.

21/08/01 09:46:03 WARN lineage.LineageWriter: Lineage directory /var/log/spark/lineage doesn't exist or is not writable. Lineage for this application will be disabled.
+---+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
|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

bank_people_data: org.apache.spark.sql.DataFrame = [age: bigint, balance: bigint ... 15 more fields]
```

Fig: Loading data in Spark data frame

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

Marketing success rate:

```
val sub_count = bank_people_data .filter($"y"=="yes").count()
val totalcount = bank_people_data .count().toDouble
val success_rate = (sub_count/totalcount)*100
```

Sucess rate = 11.698480458295547 (approx: 11.70%)

```
scala> val sub_count = bank_people_data .filter($"y"=="yes").count()
sub_count: Long = 5289

scala> val totalcount = bank_people_data .count().toDouble
totalcount: Double = 45211.0

scala> val success_rate = (sub_count/totalcount)*100
success_rate: Double = 11.698480458295547
```

Fig : Marketing success rate

Give marketing failure rate:

```
val fail_count = bank_people_data.filter($"y"=="no").count().toDouble
val failure_rate = (fail_count/totalcount)*100
```

Failure rate = 88.30151954170445 (approx: 88.3%)

```
scala> val fail_count = bank_people_data.filter($"y"=="no").count().toDouble
fail_count: Double = 39922.0

scala> val failure_rate = (fail_count/totalcount)*100
failure_rate: Double = 88.30151954170445

scala>
```

Fig : Failure rate

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

- Maximum age:

```
bank_people_data.select(max($"age")).show()
```

- Mean age:

```
bank_people_data.select(min($"age")).show()
```

- Minimum Age:

```
bank_people_data.select(avg($"age")).show()
```

```
scala> bank_people_data.select(max($"age")).show()
+--------+
|max(age)|
+--------+
|      95|
+--------+


scala> bank_people_data.select(min($"age")).show()
+--------+
|min(age)|
+--------+
|      18|
+--------+


scala> bank_people_data.select(avg($"age")).show()
+-----------------+
|         avg(age)|
+-----------------+
|40.93621021432837|
+-----------------+


scala>
```

Fig: maximum, mean, and minimum age of the average targeted customer.

Max age: 95

Min Age: 18

Average Age : 40.93621021432837

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

```
sql("select avg(balance), percentile_approx(balance, 0.5) from erabhishektiwari.datanewtable").show
```

```
scala> sql("select avg(balance), percentile_approx(balance, 0.5) from erabhishektiwari.datanewtable").show
+------------------+-----------------------------------------------------+
|      avg(balance)|percentile_approx(balance, CAST(0.5 AS DOUBLE), 10000)|
+------------------+-----------------------------------------------------+
|1362.2720576850766|                                                  448|
+------------------+-----------------------------------------------------+


scala>
```

Fig: quality of customers by checking average balance, median balance of customers

Average Balance: 1362.2720576850766

Median Balance: 448

---

5. Check if age matters in marketing subscription for deposit

```
sql("select age, count(*) as Age_Count from erabhishektiwari.datanewtable where y = 'yes' group by age order by Age_count desc").show
```

```
scala> sql("select age, count(*) as Age_Count from erabhishektiwari.datanewtable where y = 'yes' group by age order by Age_count desc").show
+---+---------+
|age|Age_Count|
+---+---------+
| 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
```

Fig: age matters in marketing subscription for deposit

Here, we can see that most of the subscription are brought by the people between the age of 30-36

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

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

maritaldata.show()
```

```
scala> val maritaldata = spark.sql("select marital, count(*) as number from erabhishektiwari.datanewtable where y='yes' group by marital order by number desc")
maritaldata: org.apache.spark.sql.DataFrame = [marital: string, number: bigint]

scala> maritaldata.show()
+--------+------+
| marital|number|
+--------+------+
| married|  2755|
|  single|  1912|
|divorced|   622|
+--------+------+


scala>
```

Fig: marital status mattered for a subscription to deposit.

We can clearly see that married people have more number of subscription (2755) as compared to single (1912) and divorced (622).

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

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

ageandmaritaldata.show()
```

```
scala> ageandmaritaldata.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|
| 32|married|    87|
| 39|married|    87|
| 38|married|    86|
| 35| single|    84|
| 47|married|    83|
| 31|married|    80|
+---+-------+------+
only showing top 20 rows
```

Fig: age and marital status

As we can see that most number of subscription are made by Single peoples, also we can see that most of the singles are in the age group of 26-32.

We can also see that married people in the age group of 31 - 37 have been subscribed.

Specifically married people in the age limit of 34 - 36 years.

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

The main objective of this feature engineering is that which age group is more important for subscriptions

```
val agedata = spark.udf.register("agedata",(age:Int) => {
if (age < 20)
"Teen"
else if (age > 20 && age <= 32)
"Young"
else if (age > 33 && age <= 55)
"Middle Aged"
else
"old"
})

//Replacing the old age column with the new age column


val banknewDF = bank_people_data.withColumn("age",agedata(bank_people_data("age")))
banknewDF.show()

banknewDF.registerTempTable("banknewtable")

//which age group subscribed the most

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

```
// Exiting paste mode, now interpreting.

warning: there was one deprecation warning; re-run with -deprecation for details
21/08/01 11:24:01 WARN analysis.SimpleFunctionRegistry: The function agedata replaced a previously registered function.
+-----------+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
|        age|balance|campaign|contact|day|default|duration|education|housing|         job|loan| marital|month|pdays|poutcome|previous|  y|
+-----------+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
|        old|   2143|       1|unknown|  5|     no|     261| tertiary|    yes|  management|  no| married|  may|   -1| unknown|       0| no|
|Middle Aged|     29|       1|unknown|  5|     no|     151|secondary|    yes|  technician|  no|  single|  may|   -1| unknown|       0| no|
|        old|      2|       1|unknown|  5|     no|      76|secondary|    yes|entrepreneur| yes| married|  may|   -1| unknown|       0| no|
|Middle Aged|   1506|       1|unknown|  5|     no|      92|  unknown|    yes| blue-collar|  no| married|  may|   -1| unknown|       0| no|
|        old|      1|       1|unknown|  5|     no|     198|  unknown|     no|     unknown|  no|  single|  may|   -1| unknown|       0| no|
|Middle Aged|    231|       1|unknown|  5|     no|     139| tertiary|    yes|  management|  no| married|  may|   -1| unknown|       0| no|
|      Young|    447|       1|unknown|  5|     no|     217| tertiary|    yes|  management| yes|  single|  may|   -1| unknown|       0| no|
|Middle Aged|      2|       1|unknown|  5|    yes|     380| tertiary|    yes|entrepreneur|  no|divorced|  may|   -1| unknown|       0| no|
|        old|    121|       1|unknown|  5|     no|      50|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
|Middle Aged|    593|       1|unknown|  5|     no|      55|secondary|    yes|  technician|  no|  single|  may|   -1| unknown|       0| no|
|Middle Aged|    270|       1|unknown|  5|     no|     222|secondary|    yes|      admin.|  no|divorced|  may|   -1| unknown|       0| no|
|      Young|    390|       1|unknown|  5|     no|     137|secondary|    yes|      admin.|  no|  single|  may|   -1| unknown|       0| no|
|Middle Aged|      6|       1|unknown|  5|     no|     517|secondary|    yes|  technician|  no| married|  may|   -1| unknown|       0| no|
|        old|     71|       1|unknown|  5|     no|      71|  unknown|    yes|  technician|  no| married|  may|   -1| unknown|       0| no|
|        old|    162|       1|unknown|  5|     no|     174|secondary|    yes|    services|  no| married|  may|   -1| unknown|       0| no|
|Middle Aged|    229|       1|unknown|  5|     no|     353|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
|Middle Aged|     13|       1|unknown|  5|     no|      98|  unknown|    yes|      admin.|  no|  single|  may|   -1| unknown|       0| no|
|        old|     52|       1|unknown|  5|     no|      38|  primary|    yes| blue-collar|  no| married|  may|   -1| unknown|       0| no|
|        old|     60|       1|unknown|  5|     no|     219|  primary|    yes|     retired|  no| married|  may|   -1| unknown|       0| no|
|        old|      0|       1|unknown|  5|     no|      54|secondary|    yes|    services|  no| married|  may|   -1| unknown|       0| no|
+-----------+-------+--------+-------+---+-------+--------+---------+-------+------------+----+--------+-----+-----+--------+--------+---+
only showing top 20 rows
```

Fig: banknewDF.show()

```
+------------+------+
|         age|number|
+------------+------+
|Middle Aged|  2601|
|      Young|  1539|
|        old|  1131|
|       Teen|    18|
+------------+------+
```

Fig:targetage.show()

As we can see here, most of the subscriber are in the Middle age between the age group 33-55.

End of Assessment