Market Analysis in Banking Domain

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.

Analysis tasks to be done:

1. Load data and create a Spark data frame
2. Give marketing success rate (No. of people subscribed / total no. of entries) 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 for 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. **Upload file through FTP to local server, then copy it into HDFS and check if file exists:**

```
hadoop fs -put 'Bank.txt' /user/bnmwengmail/assessment/

hadoop fs -ls /user/bnmwengmail/assessment/
```

Import required packages and create a spark data frame:

```
import org.apache.spark.sql.DataFrame

import org.apache.spark.sql.SQLContext

val sqlContext = new org.apache.spark.sql.SQLContext(sc)

val df = sqlContext.read.format("com.databricks.spark.csv")

        .option("header","true").option("escape","\"").option("delimiter",";")

        .load("/user/bnmwengmail/assessment/Bank.txt")
```

```
df.show(20)
```

2. **Calculate marketing success rate:**

```
# No. of people subscribed
val subscribedCount = df.filter($”y”===”yes”).count().toDouble
# Total no. of entries
val totalCount = df.count().toDouble
var successRate = subscribedCount / totalCount
val failureRate = (totalCount – subscribedCount) / totalCount
```

Marketing success rate is 11.70%, failure rate is about 88.30%.

3. **Maximum, Minimun and Mean age of average targeted customer:**

```
df.select(max($”age”), min($”age”), avg($”age”)).show()
```

Among campaign population, max age is 95, min age is 18, mean age is 41.

4. **Check the quality of the customers by checking the balance:**

```
# create a temporary table
df.registerTempTable("bankdetails")
sqlContext.sql("select avg(balance) as average,percentile(balance,0.5) as
median from bankdetails").show()
```

5. **Check if age matters in marketing campaign:**

```
val ageMatt = sqlContext.sql(“select age, count(*) as number from bankdetails
where y = ‘yes’ group by age order by number desc”).show(30)
```

We see age actually matters. The age range 28-37 shows most promise.

6. **Check if marital matters:**

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

Marital status matters. The married couples subscribed the most, then single people, the divorced people subscribed the least.

7. **Check if age and marital matter:**

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

Single people around 30 show the most subscriptions.

8. **Feature engineering:**

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
```

Define a function which will groups age into 4 categories:

```
val ageRDD = sqlContext.udf.register("ageRDD", (age:Int) => {
if (age < 20) "Teenage"
else if (age > 20 && age <= 35) "Young Aged"
else if (age > 35 && age <= 55) "Middle Aged"
else "Old"
})
val df_new = df.withColumn("age", ageRDD(df("age")))
df_new.show()
df_new.registerTempTable(“bankdetails_new”)
val ageMatt1 = sqlContext.sql(“select age, count(*) as number from
bankdetails_new where y = ‘yes’ group by age order by number
desc”).show()
```

We can see “Middle Aged” group which subscribed the most.