Market Analysis in Banking Domain

(Source Code)

// hdfs dfs -copyToLocal /user/gkesapragadateksystems/Project/dataset_bank-full.csv ./

// spark-shell

// 1. Load data and create Spark data frame

```
val lines = sc.textFile("/user/gkesapragadateksystems/Project/dataset_bank-full.csv")

val bank = lines.map(x => x.split(";"))

val bfields = bank.mapPartitionsWithIndex { (idx, iter) => if (idx == 0) iter.drop(1) else iter }
```

//Define Class for the schema

```
case class Bank(age:Int, job:String, marital:String, education:String, dft:String, balance:Int,

housing:String, loan:String, contact:String, day:Int, month: String, duration:Int, campaign:Int, pdays:Int,

previous:Int, poutcome:String, y:String)


val bankrdd = bifields.map(

x => Bank(x(0).replaceAll("\"","").toInt, x(1).replaceAll("\"",""), x(2).replaceAll("\"",""), x(3).replaceAll("\"",""), x(4).replaceAll("\"",""), x(5).toInt, x(6).replaceAll("\"",""),

x(7).replaceAll("\"",""), x(8).replaceAll("\"",""), x(9).toInt, x(10).replaceAll("\"",""), x(11).toInt, x(12).toInt, x(13).toInt, x(14).toInt, x(15).replaceAll("\"",""),
x(16).replaceAll("\"","")))


val df = bankrdd.toDF()

df.registerTempTable("bank")


//spark.conf.set("spark.sql.crossJoin.enabled", "true")
```

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

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

val success = sqlContext.sql("select (a.subscribed/b.total)*100 as success_percent
from (select count(*) as subscribed from bank where y='yes') a,(select count(*) as
total from bank) b").show()
```

// 2a Give marketing failure rate

```
val failure = sqlContext.sql("select (a.not_subscribed/b.total)*100 as
failure_percent from (select count(*) as not_subscribed from bank where y='no')
a,(select count(*) as total from bank) b").show()
```

// 3. Maximum, Mean, and Minimum age of average targeted customer

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

df.select(min($"age")).show()

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

// df.select("age").summary().show()

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

df.select(avg($"balance")).show()

val median = sqlContext.sql("SELECT percentile_approx(balance, 0.5) FROM bank").show()

// 5. Check if age matters in marketing subscription for deposit

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

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

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

//df.groupBy($"y".alias("Did the customer subscribed")).agg(count($"marital").alias("marital count")).show()

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

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

// 8. Do feature engineering for column—age and find right age effect on campaign

```
import scala.reflect.runtime.universe
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.functions.mean

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

//Replacing old “age” column with new “age” column

```
val banknewDF = df.withColumn("age",ageRDD(df("age")))
banknewDF.registerTempTable("bank_new")
```

//Running a query to see the age group which subscribed the most. We see it’s ‘Middle-Aged’

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

//Pipeline

```
import org.apache.spark.ml.feature.StringIndexer

val ageInd = new StringIndexer().setInputCol("age").setOutputCol("ageIndex")

//Fitting the model

var strIndModel = ageInd.fit(banknewDF)



strIndModel.transform(banknewDF).select("age","ageIndex").show(5)
```

So we can conclude from the Feature Engineering that It is the ‘Middle Aged’ people between age 33 and 55 who should be the targeted customers as they subscribe the most