**Market Analysis in Banking Domain**

Submitted by Sampada Dagwar

**Analysis tasks**

**1: Load data and create a Spark data frame.**

val bank = spark.read.option("inferSchema","False").option("header","True").csv("Bank-full.csv")

bank.show

**Let’s convert to dataframe**

val bankdf = bank.toDF()

bankdf.registerTempTable("newtable")

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

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

* **Give marketing failure rate**

scala> val success = spark.sql("select (a.subscribed/b.total)\*100 as success\_percent from (select count(\*) as subscribed from newtable where y='yes') a,(select count(\*) as total from newtable) b").show()

Give marketing failure rate

scala> val failure = spark.sql("select (a.not\_subscribed/b.total)\*100 as failure\_percent from (select count(\*) as not\_subscribed from newtable where y='no') a,(select count(\*) as total from newtable) b").show()

**2.1: Give the maximum, mean, and minimum age of the average targeted customer.**

scala> bank.select(max($"age")).show()

scala> bank.select(min($"age")).show()

scala> bank.select(avg($"age")).show()

**2.2: Check the quality of customers by checking average balance, median balance of customers**

scala> bank.select(avg($"balance")).show()

scala> val median = spark.sql("SELECT percentile\_approx(balance, 0.5) FROM newtable").show()

**2.3: Check if age matters in marketing subscription for deposit**

scala> val agedata = spark.sql("select age, count(\*) as number from newtable where y='yes' group by age order by number desc").show()

**2.4: Check if marital status mattered for a subscription to deposit**

scala> val maritaldata = spark.sql("select marital, count(\*) as number from newtable where y='yes' group by marital order by number desc").show()

**2.5: Check if age and marital status together mattered for a subscription to deposit scheme**

scala> val ageandmaritaldata = spark.sql("select age, marital, count(\*) as number from newtable where y='yes' group by age,marital order by number desc").show()

**2.6: Do feature engineering for the bank and find the right age effect on the campaign.**

Defined a new UDF with which we will generate new features. We divide the age groups into 4 categories.

 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.withColumn("age",agedata(bank("age"))).show()

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

scala> val targetage = spark.sql("select age, count(\*) as number from newtable where y='yes' group by age order by number desc")

scala> targetage.show()

//pipelining with string Indexer

scala> import org.apache.spark.ml.feature.StringIndexer

val agedata2 = new StringIndexer().setInputCol("age").setOutputCol("ageindex")

//Fitting the model

var strindModel = agedata2.fit(banknewDF)

//assigns generated value of index of the column, by feature engineering

StringIndexerModel.transform() assigns the generated index to each value of the column in the

    given DataFrame.

Middle aged is the most frequent word in this data, so it is given index 0

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

After performing above Feature engineering , found that age between 33 and 55(middle age) who should be the targeted customers as they did subscribe the most.