Project 3 Simplilearn
Market analysis banking domain

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**
1. **Give the maximum, mean, and minimum age of the average targeted customer**
2. **Check the quality of customers by checking average balance, median balance of customers**
3. **Check if age matters in marketing subscription for deposit**
4. **Check if marital status mattered for a subscription to deposit**
5. **Check if age and marital status together mattered for a subscription to deposit scheme**
6. **Do feature engineering for the bank and find the right age effect on the campaign.**

Importing the required 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

val bank_people_data =
spark.read.option("multiline","true").json("/user/harrshavardhanangalakudur/bank_edited.json");


bank_people_data.show()

bank_people_data.registerTempTable("datanewtable")

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

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

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

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

val median = spark.sql("SELECT percentile_approx(balance, 0.5) FROM datanewtable").show()

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

agedata.show()
```

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

maritaldata.show()

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

ageandmaritaldata.show()

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()


//pipelining with string Indexer

Need to import the library

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

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