**Apache Spark—Real Time Project—Marketing Analysis**

**-By Kishan Shandilya**

Table of Contents:

1. Introduction
2. Required Tools
3. Problem Statement
4. Solution
5. Summary
6. Introduction

In the time it took you to read this sentence, terabytes of data have been collectively generated across the world — more data than any of us could ever hope to process, much less make sense of, on the machines we're using to read this notebook.

In response to this massive influx of data, the Big Data has come to the forefront in the past decade. Big Data tools such as Hadoop, sqoop, MapReduce, Hive, Pig, Spark etc., help process and store petabytes of data.

This document is a report on a mini-project completed for submission regarding marketing analysis for a Portuguese bank. The purpose is to gain a certification in the ‘Big Data Hadoop and Spark Developer’ course provided by Simplilearn.

1. Tools Required

We used the following tools for completing the project.

1. Hue – To load file from local folder to HDFS directory
2. HDFS- Storage directory
3. Hive- Tabular storage of dataset provided
4. Spark- For processing

The dataset was provided along with the problem statement in CSV format.

1. Problem Statement

Our client―a Portuguese banking institution—ran a marketing campaign to convince potential customers to invest in bank term deposit.

Information related to direct marketing campaigns of the bank are as follows.

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. The data fields are:

1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no', 'yes', 'unknown')

6 - housing: has housing loan? (categorical: 'no', 'yes', 'unknown')

7 - loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

# related to the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular', 'telephone')

9 - month: Month of last contact (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day\_of\_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (example, if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call “y” is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

12 - campaign: number of times a customer was contacted during the campaign (numeric, includes last contact)

13 - pdays: number of days passed after the customer was last contacted from a previous campaign (numeric; 999 means customer was not previously contacted)

14 - previous: number of times the customer was contacted prior to (or before) this campaign (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')

16 - y - has the customer subscribed a term deposit? (binary: 'yes', 'no')

The data size is huge and the Marketing team has asked to use Spark to help them get answers for the following questions:

1. Load data and create Spark data frame

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

2a Give marketing failure rate

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

4. Check 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 subscription to deposit.

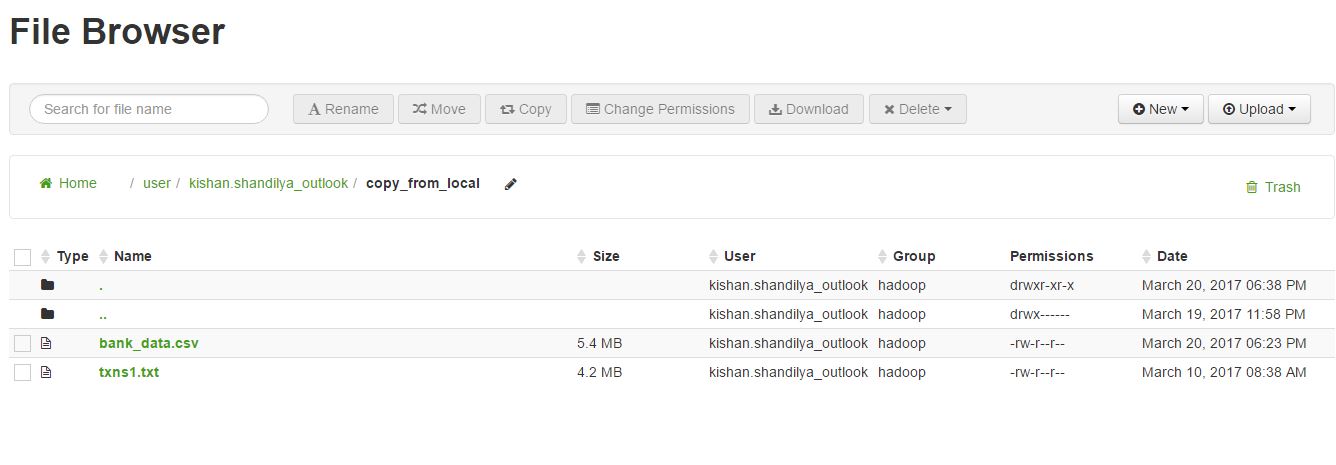
7. Check if age and marital status together mattered for subscription to deposit scheme

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

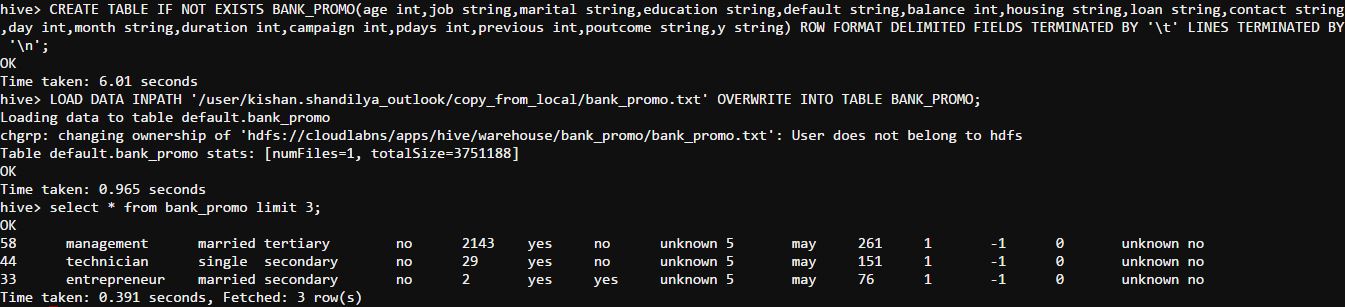
1. Solution

Pre-Processing –

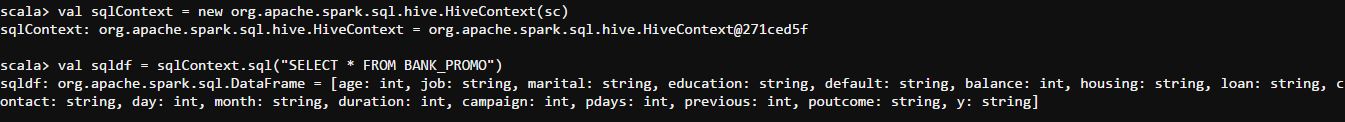
Upload dataset to HDFS using Hue browser:



Create Hive Table and Load the dataset into the table:



Create Spark Data Frame:



The Spark data frame was created using Sparksql (sqlContext) and querying the table created previously in hive:

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

val sqldf = sqlContext.sql("SELECT \* FROM BANK\_PROMO")

Success & Failure Rate:





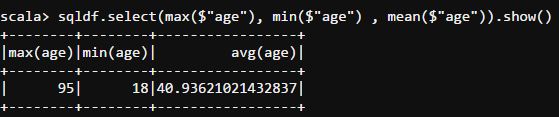
Success rate was calculated by doing a count of records with “y=yes” and dividing the resultant by total count of records.

Since the column “y” has binary value, i.e., either yes or no, simply subtracting success\_rate from 1 would give the failure rate.

val success\_rate = sqldf.filter(col("y").like("yes")).count().toDouble / sqldf.count().toDouble

val failure\_rate = 1 - success\_rate

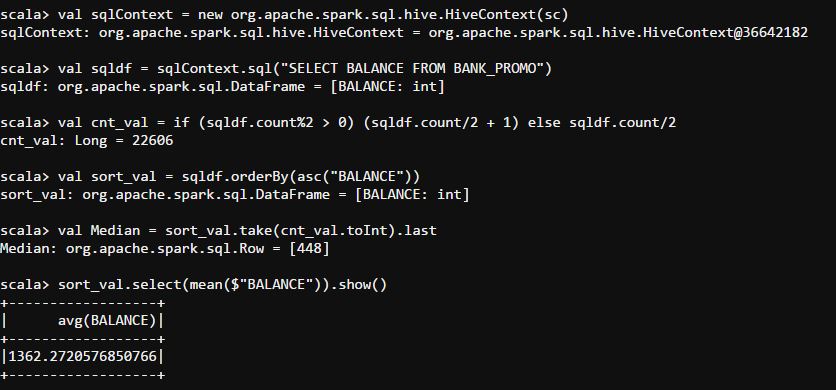
Maximum, Minimum & Mean Age of Targeted Customer:



This is be achieved by using the built-in max, min and mean functions in Spark on the data frame column(age)

sqldf.select(max($"age"), min($"age") , mean($"age")).show()

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



Here, we can see that the value of mean(1362.27) is much higher than that of median(448). Thus we can conclude that the dataset is skewed to the right.

We can also conclude that most of the customers have low incomes. However, some of the customers have very high income in comparison to the average.

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

val sqldf = sqlContext.sql("SELECT BALANCE FROM BANK\_PROMO")

val cnt\_val = if (sqldf.count%2 > 0) (sqldf.count/2 + 1) else sqldf.count/2

val sort\_val = sqldf.orderBy(asc("BALANCE"))

//Median Balance

val Median = sort\_val.take(cnt\_val.toInt).last

//Avg Balance

sort\_val.select(mean($"BALANCE")).show()

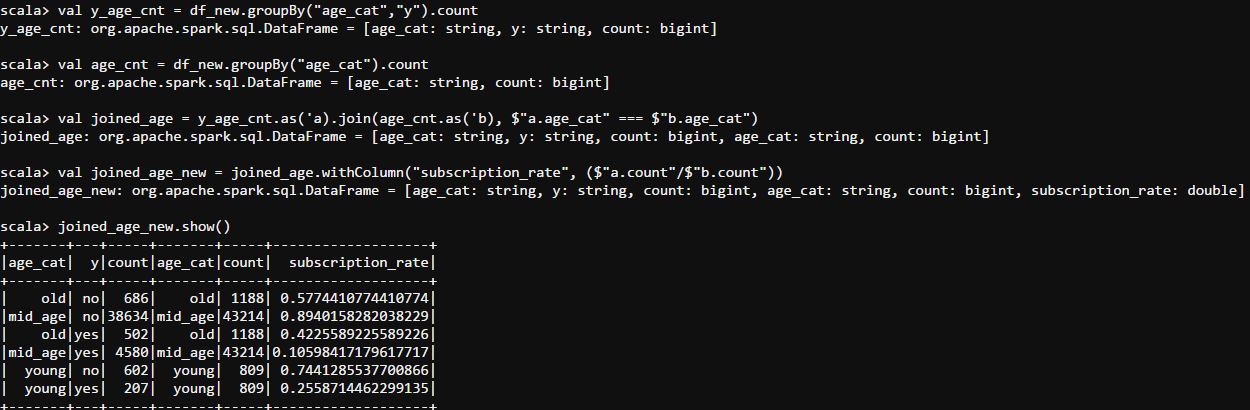
Check if age matters in marketing subscription for deposit:

Assumptions:

Age <25 – Young

Age>60 – Old

25>Age<60 – Mid Age



Here we can arrive at two situations:

1. The maximum number of customers who opted for the term deposit lie in the ‘mid age’ range.
2. We can also see that penetration is highest in customers who lie in the ‘old age’ range.

Thus we can conclude that even though the highest number of customers who opt for the subscription lie in the mid-age, there is much greater penetration in the old age range and that this group can be targeted more.

sqldf.groupBy("y").agg(avg($"age")).show()

val df\_new = sqldf.withColumn("age\_cat", when($"age" < 25 , "young").otherwise( when($"age" > 60 , "old").otherwise("mid\_age")))

df\_new.groupBy("age\_cat","y").count.show()

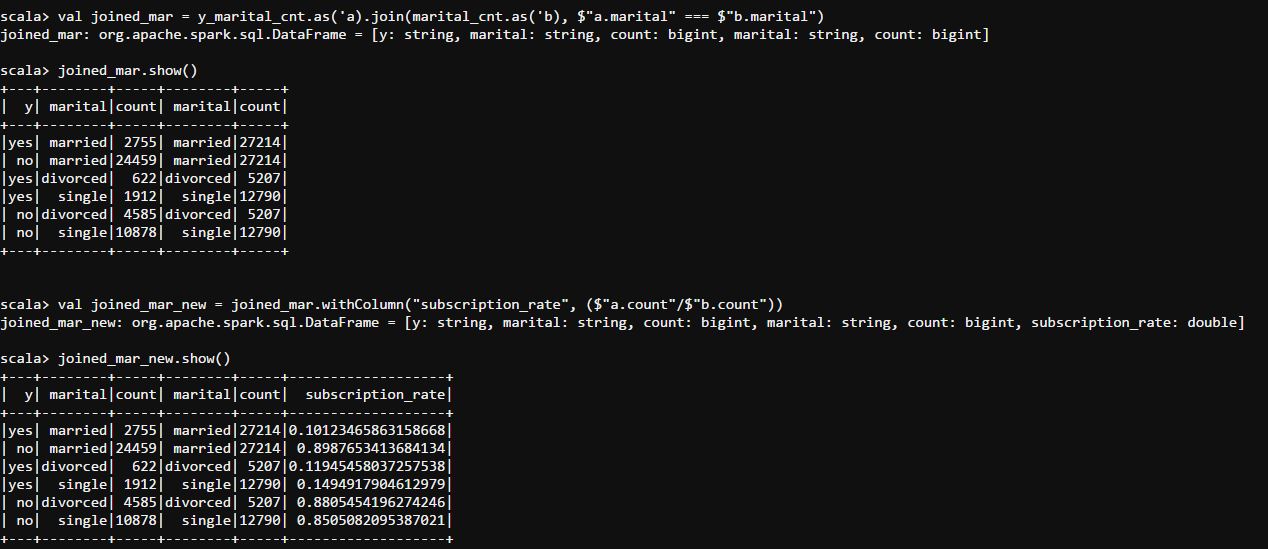
val y\_age\_cnt = df\_new.groupBy("age\_cat","y").count

val age\_cnt = df\_new.groupBy("age\_cat").count

val joined\_age = y\_age\_cnt.as('a).join(age\_cnt.as('b), $"a.age\_cat" === $"b.age\_cat")

val joined\_age\_new = joined\_age.withColumn("subscription\_rate", ($"a.count"/$"b.count"))

Check if marital status mattered for subscription to deposit



Here again we can see that that most customers who opted for the subscription are married. However, there is greater penetration in single customers.

However, the percentage penetration is only a little more (~4%). Therefore, we can safely conclude that it is in best interest to focus on customers who are married.

sqldf.groupBy("y","marital").count.show()

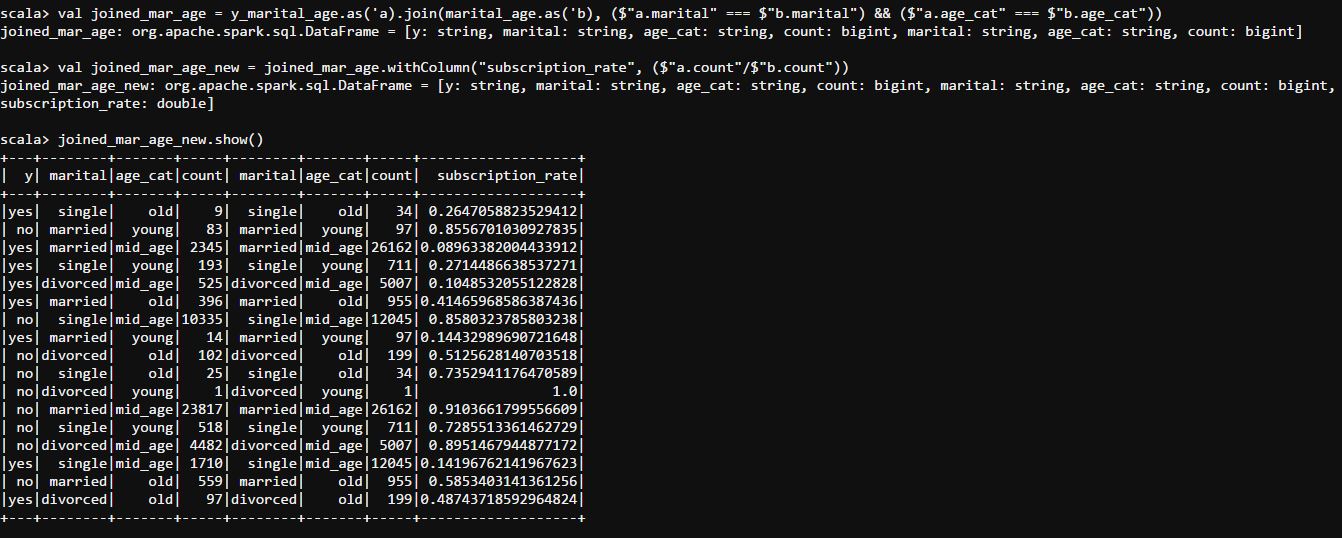
val y\_marital\_cnt = sqldf.groupBy("y","marital").count

val marital\_cnt = sqldf.groupBy("marital").count

val joined\_mar = y\_marital\_cnt.as('a).join(marital\_cnt.as('b), $"a.marital" === $"b.marital")

val joined\_mar\_new = joined\_mar.withColumn("subscription\_rate", ($"a.count"/$"b.count"))

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



Here, we can see that highest number of subscribers are married, mid-aged customers. However, the penetration is very low(~9%).

Penetration is greatest in old and married customers(~41.5%) and old and divorced customers(~49%). However, the number of old and divorced customers is low(~200).

Thus we can conclude that old, married customers are best suited here.

val df\_new = sqldf.withColumn("age\_cat", when($"age" < 25 , "young").otherwise( when($"age" > 60 , "old").otherwise("mid\_age")))

df\_new.groupBy("y","marital","age\_cat").count.show()

val y\_marital\_age = df\_new.groupBy("y","marital","age\_cat").count

val marital\_age = df\_new.groupBy("marital","age\_cat").count

val joined\_mar\_age = y\_marital\_age.as('a).join(marital\_age.as('b), ($"a.marital" === $"b.marital") && ($"a.age\_cat" === $"b.age\_cat"))

val joined\_mar\_age\_new = joined\_mar\_age.withColumn("subscription\_rate", ($"a.count"/$"b.count"))

Do feature engineering for column—age and find right age effect on campaign:

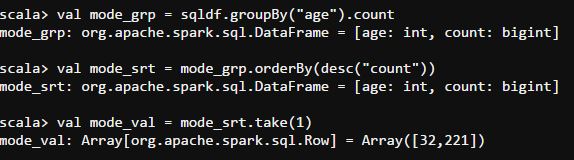
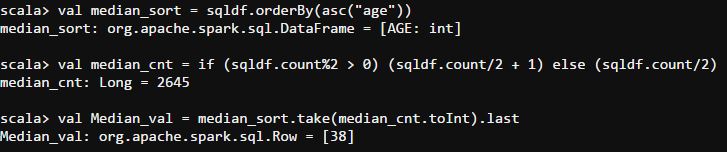
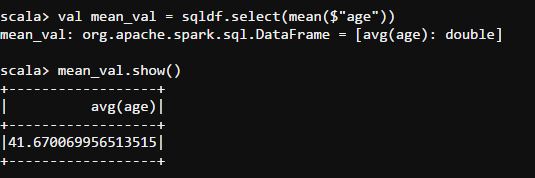
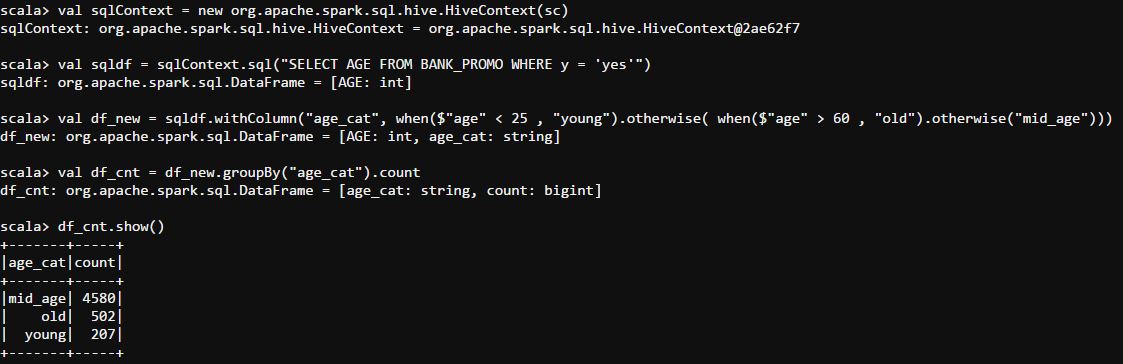
Here we opt for mean, median and mode technique of feature engineering.

Assumptions:

Age <25 – Young

Age>60 – Old

25>Age<60 – Mid Age



On evaluating the outputs, we can see that the mean (age~42), median (age 38) and mode (age 32) all lie in the mid-age range as per our assumption.

It can also be seen that most of the subscribers also belong to the mid-age range.

However, as deduced previously, penetration is much higher in customers from the old age range.

We can therefore conclude that mid-age range has the most impact in terms of quantity, but old age range is better in terms of penetration.

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

val sqldf = sqlContext.sql("SELECT AGE FROM BANK\_PROMO WHERE y = 'yes'")

val df\_new = sqldf.withColumn("age\_cat", when($"age" < 25 , "young").otherwise( when($"age" > 60 , "old").otherwise("mid\_age")))

val df\_cnt = df\_new.groupBy("age\_cat").count

val mean\_val = sqldf.select(mean($"age"))

val median\_sort = sqldf.orderBy(asc("age"))

val median\_cnt = if (sqldf.count%2 > 0) (sqldf.count/2 + 1) else (sqldf.count/2)

val Median\_val = median\_sort.take(median\_cnt.toInt).last

val mode\_grp = sqldf.groupBy("age").count

val mode\_srt = mode\_grp.orderBy(desc("count"))

val mode\_val = mode\_srt.take(1)

1. Summary

* Success Rate = 11.7% and Failure Rate = 88.3%
* Max Age = 95, Min Age = 18 and Avg. Age = 41
* Most of the customers have low incomes. However, some of the customers have very high income in comparison to the average.
* Even though the highest number of customers who opt for the subscription lie in the mid-age, there is much greater penetration in the old age range and that this group can also be targeted.
* It is in best interest to focus on customers who are married.
* Old, married people are best suited to be considered target customers for the term deposit.
* Mid-age range has the most impact in terms of quantity, but old age range is better in terms of penetration.