**Market Analysis in Banking Domain – Solution**

DESCRIPTION

**Background and 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.

**Domain**: Banking (Market Analysis)

**Dataset Description**

The data fields are as follows:

|  |  |  |
| --- | --- | --- |
| 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 a 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. |

# other attributes:

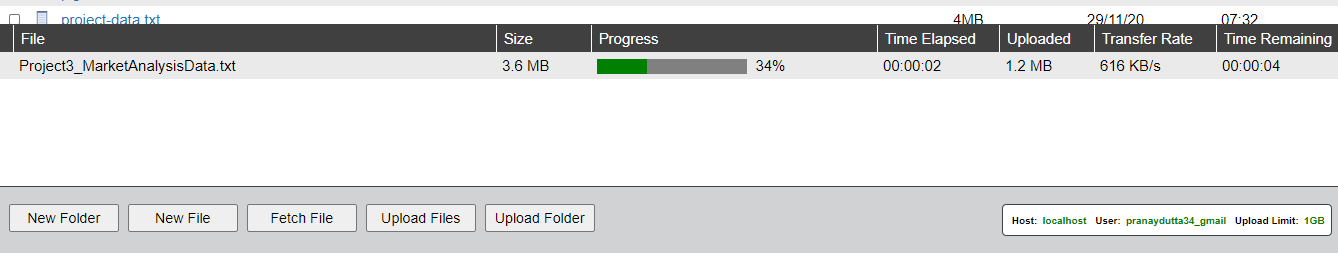
|  |  |  |
| --- | --- | --- |
| 12. | campaign | number of times a customer was contacted during the campaign (numeric, includes last contact) |
| 13. | pdays: | number of days passed after the customer was last contacted from a previous campaign (numeric; 999 means customer was not previously contacted) |
| 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') |

#Output variable (desired target):

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

Downloaded Sample dataset and uploaded into local directory via FTP

Filename: **Project3\_MarketAnalysisData.txt**



Placing file from local directory to hdfs

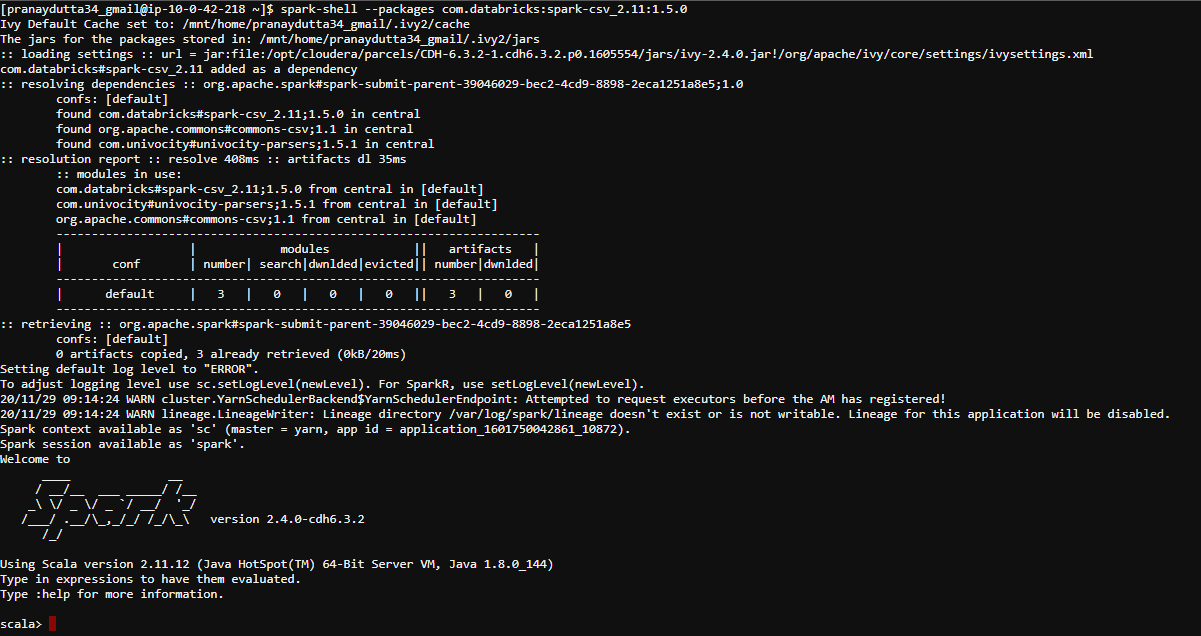
**hadoop dfs -put /mnt/home/pranaydutta34\_gmail/Project3\_MarketAnalysisData.txt /user/pranaydutta34\_gmail**



Entering into Spark shell :

code:

**spark-shell --packages com.databricks:spark-csv\_2.11:1.5.0**



**Analysis Tasks:**

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

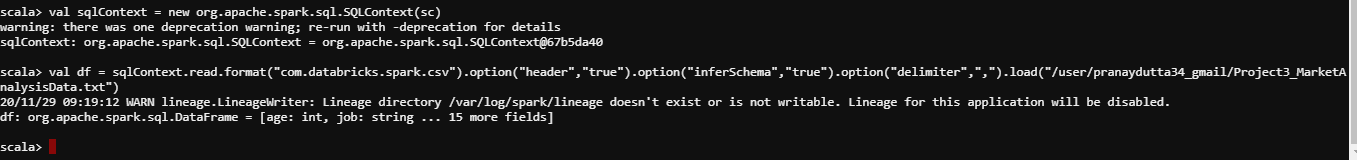
load the file as data frame using the spark-csv package

code:

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

**val df = sqlContext.read.format("com.databricks.spark.csv").option("header","true").option("inferSchema","true").option("delimiter",",").load("/user/pranaydutta34\_gmail/Project3\_MarketAnalysisData.txt.txt")**

output:



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

* **Give marketing failure rate**

To evaluate success rate and failure rate , we have calculated Total no of Entries and no of people subscribed –

No of Entries: **45211**

Code: **val totalcount = df.count().toDoubletotal**

No of Subscription: **5289**

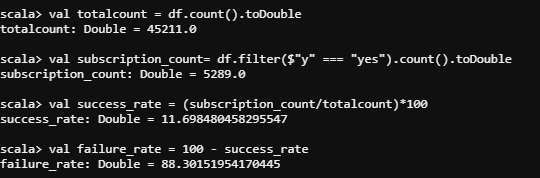
Code: **val subscription\_count= df.filter($"y" === "yes").count().toDouble**

Success rate: **11.69**

Code: **val success\_rate = (subscription\_count/totalcount)\*100**

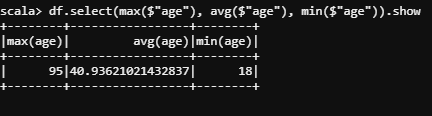
Failure rate: **88.30**

Code: **val failure\_rate = 100 – success\_rate**



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

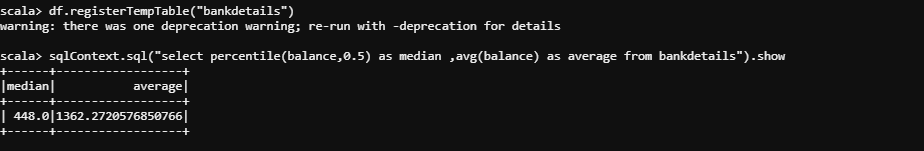
Code: **df.select(max($"age"), avg($"age"), min($"age")).show**



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

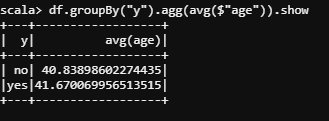
Code : **df.registerTempTable("bankdetails")**

**sqlContext.sql("select percentile(balance,0.5) as median ,avg(balance) as average from bankdetails").show**



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

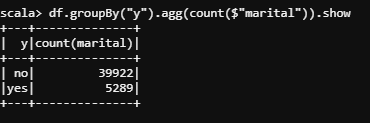
Code**: df.groupBy("y").agg(avg($"age")).show**

**’**

**Conclusion:** Age doesn’t matter in marketing subscriptions

**6.Check if marital status mattered for a subscription to deposit**

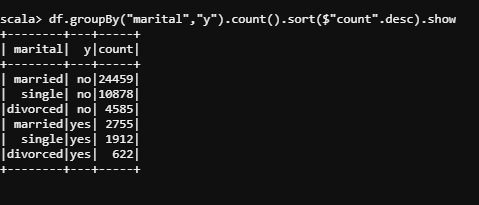
Code: **df.groupBy("y").agg(count($"marital")).show**



**Conclusion:** Marital status does matter in marketing subscriptions

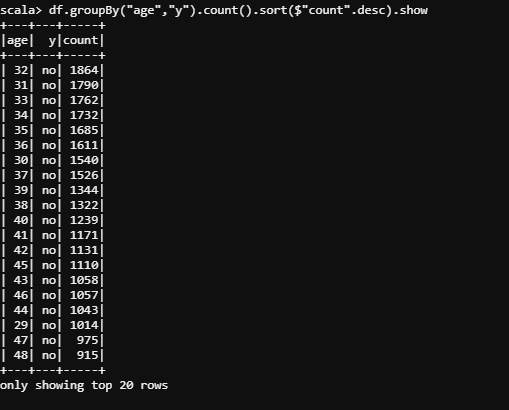
**7.Check if age and marital status together mattered for a subscription to deposit scheme**

Code: **df.groupBy("marital","y").count().sort($"count".desc).show**



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

Code: **df.groupBy("age","y").count().sort($"count".desc).show**



**df.groupBy("age","y").count().sort($"count".desc).count**



There are 148 combinations and age is between 18 and 95 and with possible subscriptions of yes and no , so it’s better to divide age category as three with 18-30 as young and 31 to 65 as mid and > 65 as old

Lets create an user defined function (UDF) for the conversion

Code: **import org.apache.spark.sql.functions.udf**



**Code :**

**def ageToCategory = udf((age:Int) => {**

**age match {**

**case t if t < 30 => "young"**

**case t if t > 65 => "old"**

**case \_ => "mid"**

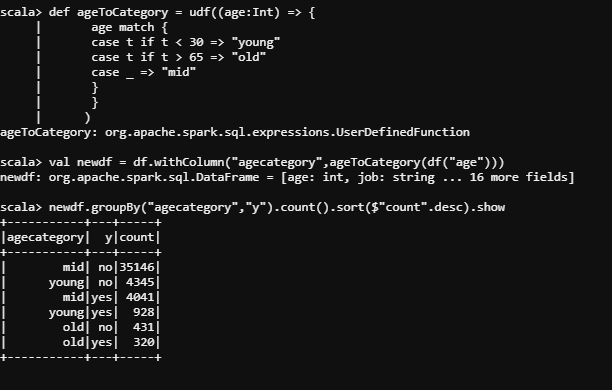
**}**

**}**

**)**

**val newdf = df.withColumn("agecategory",ageToCategory(df("age")))**

**newdf.groupBy("agecategory","y").count().sort($"count".desc).show**



**Conclusion**: Middle aged clients are much interested than other age groups