Marketing for financial services

Group members : Anamika Mishra, Gaurav Bhatia, Shardul Khot, Kartik Vashist, Vanshika Bhavnani

TASK 2.2 Data Analysis Using Big Data Tools.

Big data technologies like HDFS,Hive and Pyspark have been used as the historical data increase in size. As part of this task the following activities have been done.

Developing a Pyspark Application to load data to spark dataframes

In [1]:
```
import os
import sys
os.environ["SPARK_HOME"] = "/usr/local/spark"
os.environ["HIVE_HOME"] = "/usr/local/spark"
os.environ["PYLIB"] = os.environ["SPARK_HOME"] + "/python/lib"
```

In [2]:
```
import findspark
findspark.init('/usr/local/spark')
findspark.find()
```

Out[2]: '/usr/local/spark'

In [3]:
```
import pyspark
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
```

In [4]:
```
settings=[("hive.exec.dynamic.partition","true"),
          ("hive.exec.dynamic.partition.mode","nonstrict"),
          ("spark.sql.orc.filterPushdown","true"),
          ("hive.msck.path.validation","ignore"),
          ("spark.sql.caseSensitive","true"),
          ("spark.speculation","false"),
          ("spark.sql.warehouse.dir","/user/hive/warehouse")]
spark = SparkConf()\
        .setAppName("PythonPi")\
        .setAll(settings)

spark=SparkSession.builder\
        .enableHiveSupport()\
        .config(conf=spark)\
        .config(conf=spark)\
        .getOrCreate()
```

In [5]:
```
spark
```

Out[5]:

SparkSession - hive

SparkContext

[Spark UI (http://192.168.5.128:4040)](http://192.168.5.128:4040)

Version

`v2.2.2`

Master

`local[*]`

AppName

`PythonPi`

Perform profiling of the data to pyspark and ensuring that it is migrated correctly, wherever the source is RDBMS. Creating user defined schema for the given datasets and importing the required datasets.

In [6]:
```
from pyspark.sql.types import StructType,StructField, StringType,In
tegerType,DoubleType


state_masterSchema = StructType([
    StructField('State_code_x',StringType(),True),
    StructField('State_name', StringType(),True),
    StructField('Region_code_x', IntegerType(), True)
])
```

In [7]:
```
df1=spark.read.schema(state_masterSchema).csv('file:///home/hduser/
Desktop/capstone/State_Master.csv',header=True)
```

In [8]:
```
# df1.printSchema()
```

In [9]:
```
# spark.sql("show databases").show()
```

In [10]:
```
Region_masterSchema = StructType([
    StructField('Region_name',StringType(),True),
    StructField('Region_code_y', IntegerType(), True)
])
```

In [11]:
```
df2=spark.read.schema(Region_masterSchema).csv('file:///home/hduse
r/Desktop/capstone/Region_code_master.csv',header=True)
# df2.show()
```

In [12]:
```
social_economicSchema = StructType([
    StructField('Customer_id',IntegerType(),True),
    StructField('emp_var_rate', DoubleType(),True),
    StructField('cons_price_idx', DoubleType(), True),
    StructField('cons_conf_idx', DoubleType(), True),
    StructField('euribor3m', DoubleType(), True),
    StructField('nr_employed', DoubleType(), True)
])
```

In [13]:
```
df3=spark.read.schema(social_economicSchema).csv('file:///home/hduser/Desktop/capstone/Customer_social_economic_data_p1.csv',header=True)
# df3.show()
```

In [14]:
```
Customer_response = StructType([
    StructField('Customer_id',IntegerType(),True),
    StructField('response',StringType(),True)
])
```

In [15]:
```
df4=spark.read.schema(Customer_response).csv('file:///home/hduser/Desktop/capstone/Customer_Response_data_p1.csv',header=True)
# df4.show()
```

In [16]:
```
postal_code = StructType([
    StructField('Customer_id',IntegerType(),True),
    StructField('Postal_code',IntegerType(),True)
])
```

In [17]:
```
df5=spark.read.schema(postal_code).csv('file:///home/hduser/Desktop/capstone/Customer_Postal_Code_details.csv',header=True)
# df5.show()
```

In [18]:
```
customer_camp = StructType([
    StructField('Customer_id',IntegerType(),True),
    StructField('Contact',StringType(),True),
    StructField('Month',StringType(),True),
    StructField('Day_of_week',StringType(),True),
    StructField('Duration',IntegerType(),True),
    StructField('Campaign',IntegerType(),True),
    StructField('Pdays',IntegerType(),True),
    StructField('Previous',IntegerType(),True),
    StructField('Poutcome',StringType(),True)

])
```

In [19]:
```
df6=spark.read.schema(customer_camp).csv('file:///home/hduser/Desktop/capstone/Customer_campaign_details_p1.csv',header=True)
# df6.show()
```

In [20]:
```
bank_clientSchema = StructType([
    StructField('Customer_id',IntegerType(),True),
    StructField('Age',IntegerType(),True),
    StructField('jobs',StringType(),True),
    StructField('marital',StringType(),True),
    StructField('education',StringType(), True),
    StructField('default_credit',StringType(),True),
    StructField('housing',StringType(),True),
    StructField('loan',StringType(),True),
    StructField('Region_code',StringType(),True),
    StructField('State_code',StringType(), True),
    StructField('City_code',StringType(), True)
])
```

In [21]:
```
df7=spark.read.schema(bank_clientSchema).csv('/home/hduser/Desktop/
capstone/Customer_and_bank details_p1.csv',header=True)
# df7.show()
```

In [22]:
```
city_mast = StructType([
    StructField('City_code_z',StringType(),True),
    StructField('City_name',StringType(),True),
    StructField('State_code_z',StringType(),True),
])
```

In [23]:
```
df8=spark.read.schema(city_mast).csv('file:///home/hduser/Desktop/c
apstone/City_Master.csv',header=True)
# df8.show()
```

Joining and merging all the datasets into a single dataset.

In [24]:
```
j1=df7.join(df6,["Customer_id"])
```

In [25]:
```
j2=j1.join(df3,["Customer_id"])
```

In [26]:
```
j3=j2.join(df5,["Customer_id"])
```

In [27]:
```
j4=j3.join(df4,["Customer_id"])
```

In [28]:
```
j5=j4.join(df2,j4.Region_code ==  df2.Region_code_y,"inner")
```

In [29]:
```
j6=j5.join(df1,j5.State_code == df1.State_code_x,"inner")
```

In [30]:
```
j6.count()
```

Out[30]:
```
37024
```

In [31]:
```
j7=j6.join(df8,j5.City_code ==  df8.City_code_z,"inner")
```

In [32]:
```
j7.count()
```

Out[32]:
```
37024
```

In [33]:
```
j7.printSchema()
```

```
root
 |-- Customer_id: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- jobs: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default_credit: string (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- Region_code: string (nullable = true)
 |-- State_code: string (nullable = true)
 |-- City_code: string (nullable = true)
 |-- Contact: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Day_of_week: string (nullable = true)
 |-- Duration: integer (nullable = true)
 |-- Campaign: integer (nullable = true)
 |-- Pdays: integer (nullable = true)
 |-- Previous: integer (nullable = true)
 |-- Poutcome: string (nullable = true)
 |-- emp_var_rate: double (nullable = true)
 |-- cons_price_idx: double (nullable = true)
 |-- cons_conf_idx: double (nullable = true)
 |-- euribor3m: double (nullable = true)
 |-- nr_employed: double (nullable = true)
 |-- Postal_code: integer (nullable = true)
 |-- response: string (nullable = true)
 |-- Region_name: string (nullable = true)
 |-- Region_code_y: integer (nullable = true)
 |-- State_code_x: string (nullable = true)
 |-- State_name: string (nullable = true)
 |-- Region_code_x: integer (nullable = true)
 |-- City_code_z: string (nullable = true)
 |-- City_name: string (nullable = true)
 |-- State_code_z: string (nullable = true)
```

Writing Pyspark routines to cleanse the data , prepare the data to handle missing values and data transformation.

In [34]:
```
j7=j7.drop(*['Region_code_y','State_code_x','Region_code_x','City_code_z','State_code_z'])
```

In [35]:
```
j7.printSchema()
```

```
root
 |-- Customer_id: integer (nullable = true)
 |-- Age: integer (nullable = true)
 |-- jobs: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default_credit: string (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- Region_code: string (nullable = true)
 |-- State_code: string (nullable = true)
 |-- City_code: string (nullable = true)
 |-- Contact: string (nullable = true)
 |-- Month: string (nullable = true)
 |-- Day_of_week: string (nullable = true)
 |-- Duration: integer (nullable = true)
 |-- Campaign: integer (nullable = true)
 |-- Pdays: integer (nullable = true)
 |-- Previous: integer (nullable = true)
 |-- Poutcome: string (nullable = true)
 |-- emp_var_rate: double (nullable = true)
 |-- cons_price_idx: double (nullable = true)
 |-- cons_conf_idx: double (nullable = true)
 |-- euribor3m: double (nullable = true)
 |-- nr_employed: double (nullable = true)
 |-- Postal_code: integer (nullable = true)
 |-- response: string (nullable = true)
 |-- Region_name: string (nullable = true)
 |-- State_name: string (nullable = true)
 |-- City_name: string (nullable = true)
```

In [36]:
```
from pyspark.sql.functions import col,when
j7=j7.withColumn("default_credit", \
       when(col("default_credit")=="unknown" ,None) \
          .otherwise(col("default_credit")))
```

In [37]:
```
j7=j7.withColumn("education", \
       when(col("education")=="unknown" ,None) \
          .otherwise(col("education")))
```

In [38]:
```
j7=j7.withColumn("housing", \
       when(col("housing")=="unknown" ,None) \
          .otherwise(col("housing")))
```

In [39]:
```
j7=j7.withColumn("loan", \
       when(col("loan")=="unknown" ,None) \
          .otherwise(col("loan")))
```

In [40]:
```
j7=j7.withColumn("Region_code", \
       when(col("Region_code")=="na" ,None) \
          .otherwise(col("Region_code")))
```

In [41]:
```
# j7.show()
```

In [42]:
```
from pyspark.sql.functions import col,isnan, when, count
j7.select(count(col('education'))).show()
```

```
+----------------+
|count(education)|
+----------------+
|           35480|
+----------------+
```

In [43]:
```
j7=j7.drop('default_credit')
```

In [44]:
```
j7=j7.na.drop()
```

In [45]:

```
j7.show()
```

```
+----------+---+-----------+-------+-------------------+-------+--
--+-----------+----------+---------+---------+-----+-----------+---
-----+--------+-----+--------+-----------+------------+-----------
--+-------------+---------+-----------+-----------+--------+-------
----+--------------+--------------+
|Customer_id|Age|       jobs|marital|          education|housing|lo
an|Region_code|State_code|City_code|  Contact|Month|Day_of_week|Dur
ation|Campaign|Pdays|Previous|   Poutcome|emp_var_rate|cons_price_i
dx|cons_conf_idx|euribor3m|nr_employed|Postal_code|response|Region_
name|    State_name|      City_name|
+----------+---+-----------+-------+-------------------+-------+--
--+-----------+----------+---------+---------+-----+-----------+---
-----+--------+-----+--------+-----------+------------+-----------
--+-------------+---------+-----------+-----------+--------+-------
----+--------------+--------------+
|          1| 56|   services|married|        high.school|     no| y
es|          3|         S1|        C1|telephone|  may|          mon|
307|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      42420|      no|      Sou
th|      Kentucky|      Henderson|
|          2| 45|   services|married|           basic.9y|     no|
no|          3|         S1|        C1|telephone|  may|          mon|
198|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      42420|      no|      Sou
th|      Kentucky|      Henderson|
|          3| 59|     admin.|married|professional.course|     no|
no|          4|         S2|        C2|telephone|  may|          mon|
139|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90036|      no|       We
st|    California|    Los Angeles|
|          5| 24| technician| single|professional.course|    yes|
no|          3|         S3|        C3|telephone|  may|          mon|
380|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      33311|      no|      Sou
th|       Florida|Fort Lauderdale|
|          6| 25|   services| single|        high.school|    yes|
no|          4|         S2|        C2|telephone|  may|          mon|
50|       1|  999|       0|nonexistent|         1.1|        93.994|
-36.4|    4.857|     5191.0|      90032|      no|       West|    Ca
lifornia|    Los Angeles|
|          8| 25|   services| single|        high.school|    yes|
no|          4|         S2|        C2|telephone|  may|          mon|
222|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|      no|       We
st|    California|    Los Angeles|
|          9| 29|blue-collar| single|        high.school|     no| y
es|          4|         S2|        C2|telephone|  may|          mon|
137|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|      no|       We
st|    California|    Los Angeles|
|         11| 35|blue-collar|married|           basic.6y|    yes|
no|          4|         S2|        C2|telephone|  may|          mon|
146|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|      no|       We
st|    California|    Los Angeles|
|         12| 54|    retired|married|           basic.9y|    yes| y
es|          4|         S2|        C2|telephone|  may|          mon|
174|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|      no|       We
st|    California|    Los Angeles|
|         13| 35|blue-collar|married|           basic.6y|    yes|
```

```
no|          3|        S4|       C4|telephone|  may|        mon|
312|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      28027|      no|      Sou
th|North Carolina|       Concord|
|          14| 46|blue-collar|married|           basic.6y|    yes| y
es|          4|        S5|       C5|telephone|  may|        mon|
440|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      98103|      no|       We
st|    Washington|       Seattle|
|          15| 50|blue-collar|married|           basic.9y|    yes| y
es|          1|        S6|       C6|telephone|  may|        mon|
353|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      76106|      no|    Centr
al|         Texas|    Fort Worth|
|          16| 39| management| single|           basic.9y|     no|
no|          1|        S6|       C6|telephone|  may|        mon|
195|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      76106|      no|    Centr
al|         Texas|    Fort Worth|
|          17| 55|blue-collar|married|           basic.4y|    yes|
no|          1|        S7|       C7|telephone|  may|        mon|
262|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      53711|      no|    Centr
al|     Wisconsin|       Madison|
|          18| 55|    retired| single|        high.school|    yes|
no|          4|        S8|       C8|telephone|  may|        mon|
342|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      84084|      no|       We
st|          Utah|   West Jordan|
|          19| 41| technician| single|        high.school|    yes|
no|          4|        S2|       C9|telephone|  may|        mon|
181|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      94109|      no|       We
st|    California|  San Francisco|
|          20| 37|     admin.|married|        high.school|    yes|
no|          4|        S2|       C9|telephone|  may|        mon|
172|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      94109|      no|       We
st|    California|  San Francisco|
|          21| 35| technician|married|  university.degree|     no| y
es|          4|        S2|       C9|telephone|  may|        mon|
99|       1|  999|       0|nonexistent|         1.1|        93.994|
-36.4|    4.857|     5191.0|      94109|      no|       West|    Ca
lifornia|  San Francisco|
|          23| 54| technician| single|  university.degree|     no|
no|          1|        S9|      C10|telephone|  may|        mon|
255|       2|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      68025|      no|    Centr
al|      Nebraska|       Fremont|
|          27| 54| management|married|           basic.4y|    yes|
no|          4|        S2|       C2|telephone|  may|        mon|
230|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90049|      no|       We
st|    California|    Los Angeles|
+------------+---+-----------+-------+-------------------+-------+--
--+-----------+----------+----------+----------+-----+-----------+---
-----+--------+-----+--------+-----------+-------------+-------------
--+-------------+----------+------------+-----------+--------+-------
----+--------------+--------------+
only showing top 20 rows
```

Saving the final dataset into hive tables on a hadoop cluster.

In [46]:
```
spark.sql('show databases').show()
```

```
+------------+
|databaseName|
+------------+
|    capstone|
|     default|
+------------+
```

In [47]:
```
spark.sql('use capstone').show()
```

```
++
||
++
++
```

In [48]:
```
spark.sql('show tables').show()
```

```
+--------+----------------+-----------+
|database|       tableName|isTemporary|
+--------+----------------+-----------+
|capstone|       projectdb|      false|
|capstone|projectdb_bucket|      false|
+--------+----------------+-----------+
```

In [49]:
```
j7.write.saveAsTable('projectdb')
```

Storing the dataset into hive tables on a hadoop cluster in an optimized format.

In [50]:
```
j7.write.partitionBy('State_name').saveAsTable('capstone.projectdb_
bucket')
```

Developing a Pyspark Application to implement and evaluate the Machine Learning model identified with appropriate metrics.

In [51]:
```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
from pyspark.ml import Pipeline
from pyspark.sql.functions import *
```

Encoding all the categorical variables in order to include them as features in the ML model

In [52]:
```
j7=j7.withColumn("housing", \
       when(col("housing")=="yes" ,1) \
          .otherwise(0))
```

In [53]:
```
j7=j7.withColumn("loan", \
       when(col("loan")=="yes" ,1) \
          .otherwise(0))
```

In [54]:
```
j7=j7.withColumn("response", \
       when(col("response")=="yes" ,1) \
          .otherwise(0))
```

In [55]:

```
from pyspark.ml.feature import StringIndexer
jobs_indexer = StringIndexer(inputCol="jobs", outputCol="jobsInde
x")
j7_encoded= jobs_indexer.fit(j7).transform(j7)
j7_encoded.show()
```

```
+----------+---+-----------+-------+-------------------+-------+--
--+-----------+----------+--------+---------+-----+-----------+---
-----+--------+-----+--------+------------+------------+------------
--+------------+---------+-----------+-----------+--------+-------
----+--------------+-----------------+---------+
|Customer_id|Age|       jobs|marital|          education|housing|lo
an|Region_code|State_code|City_code|  Contact|Month|Day_of_week|Dur
ation|Campaign|Pdays|Previous|   Poutcome|emp_var_rate|cons_price_i
dx|cons_conf_idx|euribor3m|nr_employed|Postal_code|response|Region_
name|    State_name|      City_name|jobsIndex|
+----------+---+-----------+-------+-------------------+-------+--
--+-----------+----------+--------+---------+-----+-----------+---
-----+--------+-----+--------+------------+------------+------------
--+------------+---------+-----------+-----------+--------+-------
----+--------------+-----------------+---------+
|          1| 56|   services|married|        high.school|      0|
1|          3|         S1|       C1|telephone|  may|        mon|
307|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      42420|       0|      Sou
th|      Kentucky|      Henderson|      3.0|
|          2| 45|   services|married|           basic.9y|      0|
0|          3|         S1|       C1|telephone|  may|        mon|
198|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      42420|       0|      Sou
th|      Kentucky|      Henderson|      3.0|
|          3| 59|     admin.|married|professional.course|      0|
0|          4|         S2|       C2|telephone|  may|        mon|
139|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90036|       0|       We
st|    California|    Los Angeles|      0.0|
|          5| 24| technician| single|professional.course|      1|
0|          3|         S3|       C3|telephone|  may|        mon|
380|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      33311|       0|      Sou
th|       Florida|Fort Lauderdale|      2.0|
|          6| 25|   services| single|        high.school|      1|
0|          4|         S2|       C2|telephone|  may|        mon|
50|       1|  999|       0|nonexistent|         1.1|        93.994|
-36.4|    4.857|     5191.0|      90032|       0|       West|    Ca
lifornia|    Los Angeles|      3.0|
|          8| 25|   services| single|        high.school|      1|
0|          4|         S2|       C2|telephone|  may|        mon|
222|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|       0|       We
st|    California|    Los Angeles|      3.0|
|          9| 29|blue-collar| single|        high.school|      0|
1|          4|         S2|       C2|telephone|  may|        mon|
137|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|       0|       We
st|    California|    Los Angeles|      1.0|
|         11| 35|blue-collar|married|           basic.6y|      1|
0|          4|         S2|       C2|telephone|  may|        mon|
146|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|       0|       We
st|    California|    Los Angeles|      1.0|
|         12| 54|    retired|married|           basic.9y|      1|
1|          4|         S2|       C2|telephone|  may|        mon|
174|       1|  999|       0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90032|       0|       We
st|    California|    Los Angeles|      5.0|
|         13| 35|blue-collar|married|           basic.6y|      1|
```

```
0|          3|         S4|         C4|telephone|  may|         mon|
312|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      28027|       0|      Sou
th|North Carolina|        Concord|      1.0|
|          14| 46|blue-collar|married|           basic.6y|      1|
1|          4|         S5|         C5|telephone|  may|         mon|
440|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      98103|       0|       We
st|    Washington|        Seattle|      1.0|
|          15| 50|blue-collar|married|           basic.9y|      1|
1|          1|         S6|         C6|telephone|  may|         mon|
353|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      76106|       0|    Centr
al|          Texas|     Fort Worth|      1.0|
|          16| 39| management| single|           basic.9y|      0|
0|          1|         S6|         C6|telephone|  may|         mon|
195|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      76106|       0|    Centr
al|          Texas|     Fort Worth|      4.0|
|          17| 55|blue-collar|married|           basic.4y|      1|
0|          1|         S7|         C7|telephone|  may|         mon|
262|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      53711|       0|    Centr
al|     Wisconsin|        Madison|      1.0|
|          18| 55|    retired| single|        high.school|      1|
0|          4|         S8|         C8|telephone|  may|         mon|
342|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      84084|       0|       We
st|          Utah|    West Jordan|      5.0|
|          19| 41| technician| single|        high.school|      1|
0|          4|         S2|         C9|telephone|  may|         mon|
181|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      94109|       0|       We
st|    California|  San Francisco|      2.0|
|          20| 37|     admin.|married|        high.school|      1|
0|          4|         S2|         C9|telephone|  may|         mon|
172|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      94109|       0|       We
st|    California|  San Francisco|      0.0|
|          21| 35| technician|married|  university.degree|      0|
1|          4|         S2|         C9|telephone|  may|         mon|
99|       1|  999|        0|nonexistent|         1.1|        93.994|
-36.4|    4.857|     5191.0|      94109|       0|       West|    Ca
lifornia|  San Francisco|      2.0|
|          23| 54| technician| single|  university.degree|      0|
0|          1|         S9|        C10|telephone|  may|         mon|
255|       2|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      68025|       0|    Centr
al|      Nebraska|        Fremont|      2.0|
|          27| 54| management|married|           basic.4y|      1|
0|          4|         S2|         C2|telephone|  may|         mon|
230|       1|  999|        0|nonexistent|         1.1|        93.994
|        -36.4|    4.857|     5191.0|      90049|       0|       We
st|    California|    Los Angeles|      4.0|
+------------+---+-----------+-------+--------------------+-------+--
--+-----------+-----------+----------+----------+------+------------+---
-----+---------+------+---------+------------+-------------+------------
--+---------------+----------+------------+-----------+---------+-------
----+---------------+---------------+----------+
only showing top 20 rows
```

In [56]:
```
education_indexer = StringIndexer(inputCol="education", outputCol
="educationIndex")
j7_encoded= education_indexer .fit(j7_encoded).transform(j7_encode
d)
# j7_encoded.show()
```

In [57]:
```
marital_indexer = StringIndexer(inputCol="marital", outputCol="mari
talIndex")
j7_encoded= marital_indexer .fit(j7_encoded).transform(j7_encoded)
# j7_encoded.show()
```

In [58]:
```
Region_code_indexer = StringIndexer(inputCol="Region_code", outputC
ol="Region_codeIndex")
j7_encoded= Region_code_indexer .fit(j7_encoded).transform(j7_encod
ed)
# j7_encoded.show()
```

In [59]:
```
State_code_indexer = StringIndexer(inputCol="State_code", outputCol
="State_codeIndex")
j7_encoded= State_code_indexer .fit(j7_encoded).transform(j7_encode
d)
```

In [60]:
```
City_code_indexer = StringIndexer(inputCol="City_code", outputCol
="City_codeIndex")
j7_encoded= City_code_indexer .fit(j7_encoded).transform(j7_encode
d)
```

In [61]:
```
Contact_indexer = StringIndexer(inputCol="Contact", outputCol="Cont
actIndex")
j7_encoded= Contact_indexer .fit(j7_encoded).transform(j7_encoded)
```

In [62]:
```
Month_indexer = StringIndexer(inputCol="Month", outputCol="MonthInd
ex")
j7_encoded= Month_indexer.fit(j7_encoded).transform(j7_encoded)
# j7_encoded.show()
```

In [63]:
```
poutcome_indexer = StringIndexer(inputCol="Poutcome", outputCol="Po
utcomeIndex")
j7_encoded= poutcome_indexer.fit(j7_encoded).transform(j7_encoded)
```

In [64]:
```
Day_of_week_indexer = StringIndexer(inputCol="Day_of_week", outputC
ol="Day_of_weekIndex")
j7_encoded= Day_of_week_indexer.fit(j7_encoded).transform(j7_encode
d)
```

In [65]:
```
j7_encoded=j7_encoded.drop(*['jobs','marital','education','Region_c
ode','State_code','City_code','Contact',
                             'Month','Day_of_week','Poutcome','Regi
on_name','State_name','City_name'])
```

Using the features defined to train ML models.

In [66]:
```
###encoding_new_method
features=['Customer_id','Age','jobsIndex','maritalIndex','education
Index','housing','loan','Region_codeIndex',
          'State_codeIndex','City_codeIndex','ContactIndex','MonthI
ndex','Day_of_weekIndex','Duration','Campaign','Pdays',
          'Previous','PoutcomeIndex','emp_var_rate','cons_price_id
x','cons_conf_idx','euribor3m',
          'nr_employed','Postal_code']
```

In [67]:
```
assembler= VectorAssembler(inputCols=['Age','jobsIndex','maritalInd
ex','educationIndex','housing','loan','Region_codeIndex',
          'State_codeIndex','City_codeIndex','ContactIndex','MonthI
ndex','Day_of_weekIndex','Duration','Campaign','Pdays',
          'Previous','PoutcomeIndex','emp_var_rate','cons_price_id
x','cons_conf_idx','euribor3m',
          'nr_employed','Postal_code'],outputCol='features_new')
```

In [68]:
```
assembler
```

Out[68]:
```
VectorAssembler_430b9dc0247ff34e9e73
```

In [69]:
```
output= assembler.transform(j7_encoded)
```

In [70]:
```
model_df=output.select('features_new','response')
```

Splitting the final datatset into training dataset and testing dataset.

In [71]:
```
training_df,test_df=model_df.randomSplit([0.7,0.3])
```

In [72]:
```
print(training_df.count())
```

```
24142
```

In [73]:
```
print(test_df.count())
```

```
10492
```

Applying Logistic Regression to the splitted dataset.

In [74]:
```
from pyspark.ml.classification import LogisticRegression
```

In [75]:
```
model=LogisticRegression(featuresCol='features_new',labelCol='respo
nse')
```

In [76]:
```
log_reg=model.fit(training_df)
```

In [77]:
```
# training results
train_results=log_reg.evaluate(training_df).predictions
```

In [78]:
```
results=log_reg.evaluate(test_df).predictions
```

In [79]:
```
results.select(['response','prediction']).show(10,False)
```

```
+--------+----------+
|response|prediction|
+--------+----------+
|0       |0.0       |
|1       |1.0       |
|0       |0.0       |
|0       |0.0       |
|0       |0.0       |
|0       |0.0       |
|0       |0.0       |
|0       |0.0       |
|1       |0.0       |
|0       |1.0       |
+--------+----------+
only showing top 10 rows
```

Calculating true positives(tp),true negatives(tn),false positives (fp),false negatives (fn) for confusion matrix.

In [80]:
```
# confusion matrix
tp = results[(results.response == 1) & (results.prediction == 1)].c
ount()
tp
```

Out[80]: 469

In [81]:
```
tn = results[(results.response == 0) & (results.prediction ==0)].co
unt()
tn
```

Out[81]: 9064

In [82]:
```
fp = results[(results.response== 0) & (results.prediction == 1)].co
unt()
fp
```

Out[82]: 244

In [83]:
```
fn = results[(results.response == 1) & (results.prediction ==0)].co
unt()
fn
```

Out[83]: 715

Calculating final accuracy of logistic regression

In [84]:
```
# accuracy
accuracy=float((tp+tn)/(results.count()))
print("Accuracy : ",accuracy)
```

```
Accuracy :  0.9085970263057568
```

In [85]:
```
# recall
recall = float(tn)/(tp + tn)
print("Recall: ",recall)
```

```
Recall:  0.9508024756110354
```

Applying Decision tree classifier to the splitted dataset.

In [86]:
```
from pyspark.ml.classification import DecisionTreeClassifier
```

In [87]:
```
dtc = DecisionTreeClassifier(featuresCol="features_new", labelCol
='response',maxDepth=5, maxBins=1000)
dtc = dtc.fit(training_df)
```

In [88]:
```
pred = dtc.transform(test_df)
pred.show(3)
```

```
+--------------------+--------+---------------+--------------------
+----------+
|        features_new|response|  rawPrediction|         probability
|prediction|
+--------------------+--------+---------------+--------------------
+----------+
|(23,[0,1,2,3,8,12...|       0|[15919.0,103.0]|[0.99357133940831...
|       0.0|
|(23,[0,1,2,3,8,12...|       1|  [166.0,345.0]|[0.32485322896281...
|       1.0|
|(23,[0,1,2,3,8,12...|       0|[15919.0,103.0]|[0.99357133940831...
|       0.0|
+--------------------+--------+---------------+--------------------
+----------+
only showing top 3 rows
```

In [89]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

In [90]:
```
features_new=['features_new','rawPrediction','probability','predict
ion']
```

In [91]:
```
pred = pred.select(col("response").alias("label"),*features_new)
```

In [92]:
```
pred.show(3)
```

```
+-----+--------------------+---------------+--------------------+--
--------+
|label|         features_new|  rawPrediction|         probability|pr
ediction|
+-----+--------------------+---------------+--------------------+--
--------+
|    0|(23,[0,1,2,3,8,12...|[15919.0,103.0]|[0.99357133940831...|
0.0|
|    1|(23,[0,1,2,3,8,12...|  [166.0,345.0]|[0.32485322896281...|
1.0|
|    0|(23,[0,1,2,3,8,12...|[15919.0,103.0]|[0.99357133940831...|
0.0|
+-----+--------------------+---------------+--------------------+--
--------+
only showing top 3 rows
```

In [93]:
```
# confusion matrix
tp = pred[(pred.label == 1) & (pred.prediction == 1)].count()
tp
```

Out[93]: 670

In [94]:
```
tn = pred[(pred.label == 0) & (pred.prediction ==0)].count()
tn
```

Out[94]: 8947

In [95]:
```
# accuracy
accuracy=float((tp+tn)/(pred.count()))
print("Accuracy: ",accuracy)
```

```
Accuracy:  0.916603126191384
```

In [96]:
```
recall = float(tn)/(tp + tn)
print("Recall: ",recall)
```

```
Recall:  0.9303317042736821
```

Conclusion : For the given dataset , decision tree classifier has a better accuracy as compared to logistic regression.

We have also ensured that the best practices have being followed and the design and code using the features of Pyspark.