databricksRandom Forest PCA

```
# use spark 3 cluster
from pyspark.sql import functions as fun


path = "/FileStore/tables/"
fbank = path + "bank.csv"
bank = spark.read.csv( fbank, header=True, inferSchema=True, sep=";" )
bank.printSchema()
bank.columns
```

```
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: integer (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- day: integer (nullable = true)
 |-- month: string (nullable = true)
 |-- duration: integer (nullable = true)
 |-- campaign: integer (nullable = true)
 |-- pdays: integer (nullable = true)
 |-- previous: integer (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- y: string (nullable = true)

Out[1]: ['age',
 'job',
```

```
# default : if they had a loan before
# balance : earlier account in the bank
bank = bank.drop("day", "month", "campaign", "poutcome", "contact", "pdays")


bank.describe().show()
```

```
+-------+------------------+-------+--------+---------+-------+------------------+-
------+----+------------------+------------------+----+
|summary|               age|    job| marital|education|default|           balance|h
ousing|loan|          duration|          previous|   y|
+-------+------------------+-------+--------+---------+-------+------------------+-
```

```
------+----+-----------------+-----------------+----+
|  count|               541|    541|     541|      541|    541|               541|
541| 541|               541|              541| 541|
|   mean| 41.26987060998152|   null|    null|     null|   null|1444.7818853974122|
null|null|273.11645101663584|0.5360443622920518|null|
| stddev|10.555374170161665|   null|    null|     null|   null|2423.2722735171933|
null|null| 273.7158017319274| 1.521784487914756|null|
|    min|                19| admin.|divorced|  primary|     no|             -1206|
no|  no|                5|                0|  no|
|    max|                78|unknown|  single|  unknown|    yes|             16873|
yes| yes|             1877|                20| yes|
+-------+-----------------+-------+--------+---------+-------+------------------+-
------+----+-----------------+-----------------+----+
```

```
bank.show(5, False)
```

```
+---+-----------+-------+---------+-------+-------+-------+----+--------+--------+-
--+
|age|job        |marital|education|default|balance|housing|loan|duration|previous|y
|
+---+-----------+-------+---------+-------+-------+-------+----+--------+--------+-
--+
|30 |unemployed |married|primary  |no     |1787   |no     |no  |79      |0       |n
o |
|33 |services   |married|secondary|no     |4789   |yes    |yes |220     |4       |y
es|
|35 |management |single |tertiary |no     |1350   |yes    |no  |185     |1       |y
es|
|30 |management |married|tertiary |no     |1476   |yes    |yes |199     |0       |y
es|
|59 |blue-collar|married|secondary|no     |0      |yes    |no  |226     |0       |n
o |
+---+-----------+-------+---------+-------+-------+-------+----+--------+--------+-
--+
only showing top 5 rows
```

```
categs = [ "job", "marital", "education", "default", "housing", "loan", "y" ]
for c in categs :
  print( bank.select( c ).distinct().show() )
```

```
+-------------+
|          job|
+-------------+
|   management|
|      retired|
```

```
|      unknown|
|self-employed|
|      student|
|  blue-collar|
| entrepreneur|
|       admin.|
|   technician|
|     services|
|    housemaid|
|   unemployed|
+-------------+

None
+--------+
| marital|
```

```
from pyspark.ml.feature import StringIndexer
#from pyspark.ml.feature import OneHotEncoderEstimator #Spark 2
from pyspark.ml.feature import OneHotEncoder #Spark 3

categs = [ "job", "marital", "education", "default", "housing", "loan", "y" ]
bank2 = bank.select( bank.columns )
categsInd, categsHot = [], []
for c in categs :
  stinx = StringIndexer( inputCol=c, outputCol=c+"Ind" )
  bank2 = stinx.fit( bank2 ).transform( bank2 )
  if c != "y" :
    categsInd.append( c+"Ind" )
    categsHot.append( c+"Hot" )

categs.remove("y")
bank2 = bank2.drop( *categs )

hoten = OneHotEncoder( inputCols=categsInd, outputCols=categsHot )
#hoten = OneHotEncoderEstimator( inputCols=categsInd, outputCols=categsHot )
bank3 = hoten.fit( bank2 ).transform( bank2 ).drop( *categsInd )
```

```
#bank3.select( *categsHot ).show()
bank3.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- balance: integer (nullable = true)
 |-- duration: integer (nullable = true)
 |-- previous: integer (nullable = true)
```

```
 |-- y: string (nullable = true)
 |-- yInd: double (nullable = false)
 |-- loanHot: vector (nullable = true)
 |-- maritalHot: vector (nullable = true)
 |-- jobHot: vector (nullable = true)
 |-- defaultHot: vector (nullable = true)
 |-- educationHot: vector (nullable = true)
 |-- housingHot: vector (nullable = true)
```

```
bank3.show(3,False)
```

```
+---+-------+--------+--------+---+----+-------------+-------------+--------------+
-------------+-------------+-------------+
|age|balance|duration|previous|y  |yInd|loanHot      |maritalHot   |jobHot        |
defaultHot   |educationHot |housingHot   |
+---+-------+--------+--------+---+----+-------------+-------------+--------------+
-------------+-------------+-------------+
|30 |1787   |79      |0       |no |0.0 |(1,[0],[1.0])|(2,[0],[1.0])|(11,[8],[1.0])|
(1,[0],[1.0])|(3,[2],[1.0])|(1,[],[])    |
|33 |4789   |220     |4       |yes|1.0 |(1,[],[])    |(2,[0],[1.0])|(11,[4],[1.0])|
(1,[0],[1.0])|(3,[0],[1.0])|(1,[0],[1.0])|
|35 |1350   |185     |1       |yes|1.0 |(1,[0],[1.0])|(2,[1],[1.0])|(11,[0],[1.0])|
(1,[0],[1.0])|(3,[1],[1.0])|(1,[0],[1.0])|
+---+-------+--------+--------+---+----+-------------+-------------+--------------+
-------------+-------------+-------------+
only showing top 3 rows
```

```
# Feature Assembly
from pyspark.ml.feature import VectorAssembler
bank4 = VectorAssembler( inputCols=bank3.drop("yInd", "y").columns,
outputCol="features" ).transform( bank3 )
#bank4.show()
```

```
# Random Forest
from pyspark.ml.classification import RandomForestClassifier
rf1 = RandomForestClassifier( numTrees=30, featuresCol="features", labelCol="yInd",
predictionCol="pred_y" )
bank4_train, bank4_test = bank4.randomSplit( [.7, .3] )
rf1Fit = rf1.fit( bank4_train )
print ( rf1Fit.featureImportances )
bank4_pred = rf1Fit.transform( bank4_test )
```

```
(23,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22],[0.118550072999145
34,0.05034255184014035,0.07978345390563461,0.3724651657144923,0.006984069228603078,
0.07417911476162929,0.2088955049391495,0.007065945928774916,0.0017324625626102563,0
```

```
.007905661430050365,0.007489743573221569,0.003868563387658162,0.0072929263575095965
,0.008296042797236888,0.002043334930897917,0.0019254692581359965,0.0011909986189834
365,0.0027539324735613483,0.0034928755129285463,0.0041737864325970785,0.01056430083
3408346,0.009060094052272125,0.009943928461359035])

from pyspark.ml.evaluation import MulticlassClassificationEvaluator
eval1 = MulticlassClassificationEvaluator( labelCol="yInd", predictionCol="pred_y",
metricName="accuracy" )
print( eval1.evaluate( bank4_pred ) )
print( bank4_pred.groupby("y").count().orderBy("count").show() )
bank4_pred.groupby( "y", "yInd", "pred_y" ).count().show()
```

```
0.8580246913580247
+---+-----+
|  y|count|
+---+-----+
|yes|   63|
| no|   99|
+---+-----+

None
+---+----+------+-----+
|  y|yInd|pred_y|count|
+---+----+------+-----+
| no| 0.0|   0.0|   86|
|yes| 1.0|   0.0|   10|
| no| 0.0|   1.0|   13|
|yes| 1.0|   1.0|   53|
+---+----+------+-----+
```

```
# Principle Component Analysis
from pyspark.ml.feature import PCA
pca = PCA(k=11, inputCol="features", outputCol="Pfeatures")
pcaFit = pca.fit( bank4 )
bankPca = pcaFit.transform( bank4 )
print ( "Feature variances : ", pcaFit.explainedVariance )
pcaFit.pc
```

```
Feature variances :  [0.9873888693863303,0.012592181452097393,1.819211707756263e-05
,3.8999215602469916e-07,8.105147345390498e-08,5.865008013622503e-08,4.3554327950231
957e-08,3.725077058759742e-08,2.6421403224507027e-08,2.2731628571304534e-08,1.96227
8381835792e-08]
Out[74]: DenseMatrix(23, 11, [-0.0006, -1.0, 0.0023, 0.0, -0.0, 0.0, -0.0, -0.0, ..
., 0.0785, 0.0416, 0.0272, -0.0207, -0.2522, -0.069, 0.2827, -0.0338], 0)
```

```
bankPca.select("features", "Pfeatures").show(5)
```

```
+--------------------+--------------------+
|            features|           Pfeatures|
+--------------------+--------------------+
|(23,[0,1,2,4,5,15...|[-1786.8285772384...|
|(23,[0,1,2,3,5,11...|[-4788.4905202353...|
|(23,[0,1,2,3,4,6,...|[-1349.5842636998...|
|(23,[0,1,2,5,7,18...|[-1475.5478175915...|
|(23,[0,2,4,5,8,18...|[0.49256814566750...|
+--------------------+--------------------+
only showing top 5 rows
```

```
# RF fit
from pyspark.ml.classification import RandomForestClassifier
rf2 = RandomForestClassifier( numTrees=30, featuresCol="Pfeatures",
labelCol="yInd", predictionCol="pred_y" )
bankPca_train, bankPca_test = bankPca.randomSplit( [.7, .3] )
rf2Fit = rf2.fit( bankPca_train )
print ( rf2Fit.featureImportances )
bankPca_pred = rf2Fit.transform( bankPca_test )
```

```
(11,[0,1,2,3,4,5,6,7,8,9,10],[0.03471662014293975,0.06570140599054577,0.08714963638
862659,0.3413144694951512,0.03290341754375084,0.2779401296814776,0.0475147890271894
54,0.026927824277556713,0.02298009166837252,0.033803264429610944,0.0290483513547786
1])
```

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
eval2 = MulticlassClassificationEvaluator( labelCol="yInd", predictionCol="pred_y",
metricName="accuracy" )
print( eval2.evaluate( bankPca_pred ) )
print( bankPca_pred.groupby("y").count().orderBy("count").show() )
bankPca_pred.groupby( "y", "yInd", "pred_y" ).count().show()
```

```
0.8974358974358975
+---+-----+
|  y|count|
+---+-----+
|yes|   60|
| no|   96|
+---+-----+

None
+---+----+------+-----+
|  y|yInd|pred_y|count|
```

```
+---+----+------+-----+
| no| 0.0|   0.0|   86|
|yes| 1.0|   0.0|    6|
| no| 0.0|   1.0|   10|
|yes| 1.0|   1.0|   54|
+---+----+------+-----+
```

```
#Feature importance
rf2Fit.featureImportances
```

```
Out[78]: SparseVector(11, {0: 0.0347, 1: 0.0657, 2: 0.0871, 3: 0.3413, 4: 0.0329, 5
: 0.2779, 6: 0.0475, 7: 0.0269, 8: 0.023, 9: 0.0338, 10: 0.029})
```

```
+--------------------+-------------------+--------------------+-------------------+
--------------------+
|corr(divorced, yInd)|corr(married, yInd)|  corr(single, yInd)|    corr(age, yInd)|
corr(previous, yInd)|
+--------------------+-------------------+--------------------+-------------------+
--------------------+
|-0.37532412991335645|  0.463232849343605|-0.07812659940926989|-0.1823210432736521|
0.4167583356557848|
+--------------------+-------------------+--------------------+-------------------+
--------------------+
```