TP-ML-NANA-ROMARIC-v1

September 16, 2021

1 TRAVAUX PRATIQUES

1.0.1 COURS DE MACHINE LEARNING - UNIVERSITE VIRTUELLE DU BURKINA FASO - MASTER FD & IA

- ETUDIANT : NANA SIDWENDLUIAN ROMARIC
- ENSEIGNANT : MADAME BIRBA ELIANE

1.0.2 1. Chargement des données

[1]:

```
#importing pyspark
import pyspark

#importing sparksession
from pyspark.sql import SparkSession

from pyspark.sql.functions import *

from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator

from pyspark.ml.classification import DecisionTreeClassifier

from pyspark.ml.classification import RandomForestClassifier
```

[2]:

```
#creating a sparksession object and providing appName
spark=SparkSession.builder.master("local").appName("tp").getOrCreate()
```

[3]:

```
datadft = spark.read.format("csv").options(header=True,inferSchema=True).
 ↪load("data/ccdefault.csv")
```

1.0.3 2. Analyse exploratoire

[4]:

```
datadft.printSchema()
```

```
root
 |-- ID: integer (nullable = true)
 |-- LIMIT_BAL: integer (nullable = true)
 |-- SEX: integer (nullable = true)
 |-- EDUCATION: integer (nullable = true)
 |-- MARRIAGE: integer (nullable = true)
 |-- AGE: integer (nullable = true)
 |-- PAY_0: integer (nullable = true)
 |-- PAY_2: integer (nullable = true)
 |-- PAY_3: integer (nullable = true)
 |-- PAY_4: integer (nullable = true)
 |-- PAY_5: integer (nullable = true)
 |-- PAY_6: integer (nullable = true)
 |-- BILL_AMT1: integer (nullable = true)
 |-- BILL_AMT2: integer (nullable = true)
 |-- BILL_AMT3: integer (nullable = true)
 |-- BILL_AMT4: integer (nullable = true)
 |-- BILL_AMT5: integer (nullable = true)
 |-- BILL_AMT6: integer (nullable = true)
 |-- PAY_AMT1: integer (nullable = true)
 |-- PAY_AMT2: integer (nullable = true)
 |-- PAY_AMT3: integer (nullable = true)
 |-- PAY_AMT4: integer (nullable = true)
 |-- PAY_AMT5: integer (nullable = true)
 |-- PAY_AMT6: integer (nullable = true)
 |-- DEFAULT: integer (nullable = true)
```

[5]:

```
datadft.count()
```

[5]: 30000

[6]:

```
datadft.show(5)
```

```
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+--
-------+---------+---------+---------+---------+---------+--------+--------+----
----+--------+--------+--------+-------+
| ID|LIMIT_BAL|SEX|EDUCATION|MARRIAGE|AGE|PAY_0|PAY_2|PAY_3|PAY_4|PAY_5|PAY_6|BI
LL_AMT1|BILL_AMT2|BILL_AMT3|BILL_AMT4|BILL_AMT5|BILL_AMT6|PAY_AMT1|PAY_AMT2|PAY_
AMT3|PAY_AMT4|PAY_AMT5|PAY_AMT6|DEFAULT|
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+--
-------+---------+---------+---------+---------+---------+--------+--------+----
----+--------+--------+--------+-------+
|  1|    20000|  2|        2|       1| 24|    2|    2|   -1|   -1|   -2|   -2|
3913|     3102|      689|        0|        0|        0|       0|     689|
0|       0|       0|       0|      1|
|  2|   120000|  2|        2|       2| 26|   -1|    2|    0|    0|    0|    2|
2682|     1725|     2682|     3272|     3455|     3261|       0|    1000|
1000|    1000|       0|    2000|      1|
|  3|    90000|  2|        2|       2| 34|    0|    0|    0|    0|    0|    0|
29239|    14027|    13559|    14331|    14948|    15549|    1518|    1500|
1000|    1000|    1000|    5000|      0|
|  4|    50000|  2|        2|       1| 37|    0|    0|    0|    0|    0|    0|
46990|    48233|    49291|    28314|    28959|    29547|    2000|    2019|
1200|    1100|    1069|    1000|      0|
|  5|    50000|  1|        2|       1| 57|   -1|    0|   -1|    0|    0|    0|
8617|     5670|    35835|    20940|    19146|    19131|    2000|   36681|
10000|    9000|     689|     679|      0|
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+--
-------+---------+---------+---------+---------+---------+--------+--------+----
----+--------+--------+--------+-------+
only showing top 5 rows
```

[7]:

```
datadft.describe(datadft.columns).show()
```

```
+-------+-----------------+------------------+------------------+---------------
---+------------------+-----------------+------------------+--------------------
+------------------+--------------------+------------------+-----------------+--
---------------+-----------------+-----------------+------------------+---------
--------+----------------+-----------------+------------------+-----------------
+------------------+------------------+-----------------+-------------------+
|summary|               ID|         LIMIT_BAL|               SEX|
EDUCATION|          MARRIAGE|              AGE|             PAY_0|
PAY_2|             PAY_3|               PAY_4|             PAY_5|
PAY_6|        BILL_AMT1|        BILL_AMT2|        BILL_AMT3|         BILL_AMT4|
BILL_AMT5|       BILL_AMT6|         PAY_AMT1|          PAY_AMT2|
PAY_AMT3|          PAY_AMT4|          PAY_AMT5|         PAY_AMT6|
DEFAULT|
+-------+-----------------+------------------+------------------+---------------
---+------------------+-----------------+------------------+--------------------
+------------------+--------------------+------------------+-----------------+--
---------------+-----------------+-----------------+------------------+---------
--------+----------------+-----------------+------------------+-----------------
+------------------+------------------+-----------------+-------------------+
|  count|            30000|             30000|             30000|
30000|             30000|            30000|             30000|
30000|             30000|               30000|             30000|
30000|            30000|            30000|            30000|             30000|
30000|           30000|            30000|             30000|            30000|
30000|             30000|            30000|              30000|
|   mean|          15000.5|167484.32266666667|1.6037333333333332|1.853133333333
333|1.5518666666666667|          35.4855|
-0.0167|-0.13376666666666667|           -0.1662|-0.22066666666666668|
-0.2662|          -0.2911|       51223.3309|49179.07516666667|       47013.1548|
43262.94896666666|40311.40096666667|      38871.7604|        5663.5805|
5921.1635|        5225.6815| 4826.076866666666|
4799.387633333334|5215.502566666667|             0.2212|
| stddev|8660.398374208891|129747.66156720246|0.4891291960902602|0.7903486597207
```

```
269|0.5219696006132467|9.217904068090155|1.1238015279973335|
1.1971859730345495|1.1968675684465686|  1.1691386224023357|1.1331874060027525|1.
149987625607897|73635.86057552966|71173.76878252832|69349.38742703677|64332.8561
33916444|60797.15577026471|59554.1075367459|16563.28035402577|23040.870402057186
|17606.96146980311|15666.159744032062|15278.305679144742|17777.46577543531|0.415
06180569093254|
|    min|                1|             10000|                 1|
0|                 0|               21|                -2|                  -2|
-2|                 -2|                -2|              -2|          -165580|
-69777|          -157264|          -170000|          -81334|         -339603|
0|                 0|                0|                 0|                 0|
0|                  0|
|    max|            30000|           1000000|                 2|
6|                 3|               79|                 8|                   8|
8|                  8|                 8|               8|           964511|
983931|          1664089|            891586|          927171|          961664|
873552|           1684259|            896040|           621000|
426529|           528666|                  1|
+-------+-----------------+------------------+------------------+---------------
---+------------------+------------------+------------------+-------------------
+-------------------+--------------------+------------------+-----------------+--
---------------+------------------+------------------+------------------+---------
--------+-----------------+------------------+------------------+-----------------
+-------------------+------------------+-----------------+-------------------+
```

[8]:
```
datadft.describe("LIMIT_BAL", "BILL_AMT1","PAY_AMT1","BILL_AMT2","PAY_AMT2").
↪show()
```

```
+-------+------------------+-----------------+-----------------+----------------
-+------------------+
|summary|         LIMIT_BAL|        BILL_AMT1|         PAY_AMT1|
BILL_AMT2|          PAY_AMT2|
+-------+------------------+-----------------+-----------------+----------------
-+------------------+
|  count|             30000|            30000|            30000|
30000|             30000|
|   mean|167484.32266666667|       51223.3309|
5663.5805|49179.07516666667|         5921.1635|
| stddev|129747.66156720246|73635.86057552966|16563.28035402577|71173.7687825283
2|23040.870402057186|
|    min|             10000|          -165580|                0|
-69777|                 0|
|    max|           1000000|           964511|           873552|
983931|           1684259|
+-------+------------------+-----------------+-----------------+----------------
-+------------------+
```

[9]:

```
datadft.describe("LIMIT_BAL", "AGE").show()
```

```
+-------+------------------+-----------------+
|summary|         LIMIT_BAL|              AGE|
+-------+------------------+-----------------+
|  count|             30000|            30000|
|   mean|167484.32266666667|          35.4855|
| stddev|129747.66156720246|9.217904068090155|
|    min|             10000|               21|
|    max|           1000000|               79|
+-------+------------------+-----------------+
```

[10]:

```
datadft.groupBy("SEX").count().orderBy(asc("count")).show() # SEXE ? HOMME=1␣
 ↪FEMME=2
```

```
+---+-----+
|SEX|count|
+---+-----+
|  1|11888|
|  2|18112|
+---+-----+
```

[11]:

```
datadft.groupBy("DEFAULT").count().orderBy(asc("count")).show() # DÉFAUT DE␣
 ↪PAIEMENT ? OUI=1 NON=0
```

```
+-------+-----+
|DEFAULT|count|
+-------+-----+
|      1| 6636|
|      0|23364|
+-------+-----+
```

[12]:

```
datadft.groupBy(['DEFAULT','SEX']).count().orderBy(asc("DEFAULT")).show()
# repartition de la variable cible en fonction du sexe
```

```
+-------+---+-----+
|DEFAULT|SEX|count|
+-------+---+-----+
|      0|  1| 9015|
|      0|  2|14349|
|      1|  2| 3763|
|      1|  1| 2873|
+-------+---+-----+
```

[13]:

```
datadft.groupBy(['DEFAULT','EDUCATION']).count().orderBy(asc("DEFAULT")).show()
# repartition de la variable cible en fonction du niveau d'instruction
```

```
+-------+---------+-----+
|DEFAULT|EDUCATION|count|
+-------+---------+-----+
|      0|        0|   14|
|      0|        1| 8549|
|      0|        5|  262|
|      0|        6|   43|
|      0|        2|10700|
|      0|        3| 3680|
|      0|        4|  116|
|      1|        2| 3330|
|      1|        3| 1237|
|      1|        1| 2036|
|      1|        5|   18|
|      1|        4|    7|
|      1|        6|    8|
+-------+---------+-----+
```

Les clients qui ont un niveau d'instruction plus élévé sont les plus nombreux à avoir un défaut de paiement D'après l'énoncé, la variable EDUCATION à pour valeur { 1 = graduate school; 2 = university; 3 = high school; 4 = others }. Nous constatons cependant qu'il y a des lignes où la variable EDUCATION a des valeurs plus grandes que 4. Dans ce fichier nous conservons ces lignes, puis nous appliquerons un traitement plus tard dans un autre fichier.

[14]:

```
datadft.groupBy(['DEFAULT','MARRIAGE']).count().orderBy(asc("DEFAULT")).show()
# repartition de la variable cible en fonction du statut matrimonial
```

```
+-------+--------+-----+
|DEFAULT|MARRIAGE|count|
+-------+--------+-----+
|      0|       0|   49|
|      0|       1|10453|
|      0|       2|12623|
|      0|       3|  239|
|      1|       0|    5|
|      1|       2| 3341|
|      1|       1| 3206|
|      1|       3|   84|
+-------+--------+-----+
```

D'après l'énoncé, la variable MARRIAGE à pour valeur { 1 = married; 2 = single; 3 = others }. Nous constatons cependant qu'il y a des lignes où la variable MARRIAGE a une valeur 0 dans certaines lignes. Dans ce fichier nous conservons ces lignes, puis nous appliquerons un traitement plus tard dans un autre fichier.

[15]:

```
datadft.groupBy(['DEFAULT','AGE']).count().orderBy(asc("DEFAULT")).show()
```

```
+-------+---+-----+
|DEFAULT|AGE|count|
+-------+---+-----+
|      0| 42|  609|
|      0| 27| 1164|
|      0| 39|  755|
|      0| 58|   91|
|      0| 71|    3|
|      0| 28| 1123|
|      0| 56|  129|
|      0| 50|  310|
|      0| 22|  391|
|      0| 31|  988|
|      0| 67|   11|
|      0| 40|  683|
|      0| 57|   95|
|      0| 32|  933|
|      0| 60|   44|
|      0| 73|    1|
|      0| 65|   19|
|      0| 70|    8|
|      0| 48|  362|
|      0| 25|  884|
+-------+---+-----+
only showing top 20 rows
```

[16]:

```
datadft.groupBy(['DEFAULT','LIMIT_BAL']).count().orderBy(desc("DEFAULT")).
 ↪orderBy(desc("count")).show()
```

```
+-------+---------+-----+
|DEFAULT|LIMIT_BAL|count|
+-------+---------+-----+
|      0|    50000| 2480|
|      0|    20000| 1278|
|      0|   200000| 1258|
|      0|    80000| 1204|
|      0|    30000| 1042|
|      0|   150000|  923|
|      1|    50000|  885|
|      0|   180000|  819|
|      0|   100000|  776|
|      0|   360000|  727|
|      1|    20000|  698|
|      0|   500000|  641|
|      0|   230000|  624|
|      0|   210000|  613|
|      0|    60000|  592|
|      0|   140000|  579|
|      0|   130000|  572|
|      1|    30000|  568|
|      0|   160000|  557|
|      0|   120000|  547|
+-------+--------+-----+
only showing top 20 rows
```

[17]:

```
datadft.createOrReplaceTempView("dataView")
spark.sql("SELECT DEFAULT, avg(LIMIT_BAL) AS BALANCE FROM dataView GROUP BY␣
 ↪DEFAULT ORDER BY BALANCE DESC").show()
```

```
+-------+------------------+
|DEFAULT|           BALANCE|
+-------+------------------+
|      0|178099.72607430234|
|      1|130109.65641952984|
+-------+------------------+
```

[18]:

```
datadft
```

[18]:

```
DataFrame[ID: int, LIMIT_BAL: int, SEX: int, EDUCATION: int, MARRIAGE: int, AGE:
int, PAY_0: int, PAY_2: int, PAY_3: int, PAY_4: int, PAY_5: int, PAY_6: int,
BILL_AMT1: int, BILL_AMT2: int, BILL_AMT3: int, BILL_AMT4: int, BILL_AMT5: int,
BILL_AMT6: int, PAY_AMT1: int, PAY_AMT2: int, PAY_AMT3: int, PAY_AMT4: int,
PAY_AMT5: int, PAY_AMT6: int, DEFAULT: int]
```

1.0.4 3. Preparation des données

Renommons la colonne DEFAULT en label

[19]:

```
# datadft_bis = datadft.
 ↪withColumn("ID","LIMIT_BAL","SEX","EDUCATION","MARRIAGE","AGE","PAY_0","PAY_2","PAY_3","PAY
renamedDatadft = datadft.withColumnRenamed("DEFAULT","label")
```

[20]:

```
renamedDatadft.show(5)
```

```
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+--
-------+---------+---------+---------+---------+---------+--------+--------+----
----+--------+--------+--------+-----+
| ID|LIMIT_BAL|SEX|EDUCATION|MARRIAGE|AGE|PAY_0|PAY_2|PAY_3|PAY_4|PAY_5|PAY_6|BI
LL_AMT1|BILL_AMT2|BILL_AMT3|BILL_AMT4|BILL_AMT5|BILL_AMT6|PAY_AMT1|PAY_AMT2|PAY_
AMT3|PAY_AMT4|PAY_AMT5|PAY_AMT6|label|
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+--
-------+---------+---------+---------+---------+---------+--------+--------+----
```

```
----+--------+--------+--------+-----+
|  1|    20000|  2|        2|        1| 24|     2|     2|    -1|    -1|    -2|    -2|
3913|     3102|      689|        0|        0|        0|       0|     689|
0|       0|       0|       0|    1|
|  2|   120000|  2|        2|        2| 26|    -1|     2|     0|     0|     0|     2|
2682|     1725|     2682|     3272|     3455|     3261|       0|    1000|
1000|    1000|       0|    2000|    1|
|  3|    90000|  2|        2|        2| 34|     0|     0|     0|     0|     0|     0|
29239|    14027|    13559|    14331|    14948|    15549|    1518|    1500|
1000|    1000|    1000|    5000|    0|
|  4|    50000|  2|        2|        1| 37|     0|     0|     0|     0|     0|     0|
46990|    48233|    49291|    28314|    28959|    29547|    2000|    2019|
1200|    1100|    1069|    1000|    0|
|  5|    50000|  1|        2|        1| 57|    -1|     0|    -1|     0|     0|     0|
8617|     5670|    35835|    20940|    19146|    19131|    2000|   36681|
10000|    9000|     689|     679|    0|
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+--
-------+---------+---------+---------+---------+---------+--------+--------+----
----+--------+--------+--------+-----+
only showing top 5 rows
```

[21]:
```
#  colonne des etiquettes
colLabel = "label"

#  colonne numerique
colNum = [col for col in renamedDatadft.columns if col!= colLabel]
```

[22]:
```
colNum
```

[22]:
```
['ID',
 'LIMIT_BAL',
 'SEX',
 'EDUCATION',
 'MARRIAGE',
 'AGE',
 'PAY_0',
 'PAY_2',
 'PAY_3',
 'PAY_4',
 'PAY_5',
 'PAY_6',
 'BILL_AMT1',
 'BILL_AMT2',
 'BILL_AMT3',
 'BILL_AMT4',
 'BILL_AMT5',
 'BILL_AMT6',
 'PAY_AMT1',
 'PAY_AMT2',
 'PAY_AMT3',
 'PAY_AMT4',
 'PAY_AMT5',
 'PAY_AMT6']
```

[23]:

```
from pyspark.ml.feature import VectorAssembler, StandardScaler

va =  VectorAssembler().setInputCols(colNum).
 ↪setOutputCol("to_be_scaled_features")

featuredDatadft = va.transform(renamedDatadft)

scaler =  StandardScaler().setInputCol("to_be_scaled_features").
 ↪setOutputCol("features")

dataset = scaler.fit(featuredDatadft).transform(featuredDatadft).
 ↪select("features", "label")

dataset.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[1.15468129385139...|    1|
|[2.30936258770278...|    1|
|[3.46404388155417...|    0|
|[4.61872517540556...|    0|
|[5.77340646925695...|    0|
+--------------------+-----+
only showing top 5 rows
```

1.0.5 4. Application des modèles

[24]:

```
trainSet, testSet = dataset.randomSplit([0.8,0.2])
```

[25]:

```
trainSet.count()
```

[25]:

```
24010
```

[26]:

```
testSet.count()
```

[26]:

```
5990
```

[27]:

```
trainSet.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(24,[0,1,2,3,4,5,…|    0|
|(24,[0,1,2,3,4,5,…|    0|
|(24,[0,1,2,3,4,5,…|    1|
|(24,[0,1,2,3,4,5,…|    1|
|(24,[0,1,2,3,4,5,…|    1|
+--------------------+-----+
only showing top 5 rows
```

1.0.6 4.1 Logistic Regression

[28]:

```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator

lr = LogisticRegression(maxIter=100, regParam=0.0001, elasticNetParam=0.1)
lrModel = lr.fit(trainSet)
```

[29]:

```
# trainingSummary = lrModel.summary
print("Coefficients: " + str(lrModel.coefficients))
print("Intercept: " + str(lrModel.intercept))
```

```
Coefficients: [-0.021741799077511652,-0.10177122033887191,-0.04521580458898812,-
0.07114937293373588,-0.07936493858026904,0.06967836813214995,0.6251280383863709,
0.11493582924386118,0.06944719240978961,0.05505572695441151,0.037448435283436596
,0.0005579010558137603,-0.3227225426356211,-0.031944358435879О4,0.11429461888777
985,0.05079451862353235,0.01823735451185488,0.05562332444776182,-0.1562284789069
5163,-0.2291329735750258,-0.08667448207609935,-0.05519722837168079,-0.0664881969
8795533,-0.045074589359989314]
Intercept: -0.6853446776524614
```

[30]:

```
summary = lrModel.summary
print("training set AreaUnderROC: ",summary.areaUnderROC)
summary.roc.show()
summary.pr.show()
```

```
training set AreaUnderROC:  0.7221510619656176
+--------------------+--------------------+
|                 FPR|                 TPR|
+--------------------+--------------------+
|                 0.0|                 0.0|
|4.812834224598930…|0.002824858757062147|
|0.001069518716577…|0.005273069679849341|
|0.001497326203208556|0.008286252354048965|
|0.001871657754010…|0.011487758945386064|
|0.002085561497326…|0.015254237288135594|
|0.002139037433155…|0.019585687382297552|
|0.002459893048128…|0.022975517890772127|
|0.002513368983957…|0.027306967984934087|
|0.002727272727272…|0.031073446327683617|
|0.003262032085561…| 0.03370998116760829|
|0.003475935828877…|0.037476459510357815|
|0.003850267379679…| 0.04067796610169491|
|0.004278074866310…|0.043691148775894535|
|0.004652406417112299| 0.04689265536723164|
|0.005026737967914439|0.050094161958568736|
|0.00524064171122994 6| 0.05386064030131827|
|0.005508021390374332| 0.05743879472693032|
|0.005989304812834224|0.060263653483992465|
|0.006310160427807487| 0.06365348399246705|
+--------------------+--------------------+
only showing top 20 rows

+--------------------+-------------------+
|              recall|          precision|
+--------------------+-------------------+
|                 0.0|              0.625|
|0.002824858757062147|              0.625|
|0.005273069679849341|0.5833333333333334|
|0.008286252354048965|0.6111111111111112|
|0.011487758945386064|0.6354166666666666|
|0.015254237288135594|              0.675|
|0.019585687382297552|0.7222222222222222|
|0.022975517890772127|0.7261904761904762|
|0.027306967984934087|0.7552083333333334|
|0.031073446327683617|0.7638888888888888|
| 0.03370998116760829|0.7458333333333333|
|0.037476459510357815|0.7537878787878788|
| 0.04067796610169491|               0.75|
|0.043691148775894535|0.7435897435897436|
| 0.04689265536723164|0.7410714285714286|
|0.050094161958568736|0.7388888888888889|
| 0.05386064030131827|0.7447916666666666|
| 0.05743879472693032|0.7475490196078431|
|0.060263653483992465|0.7407407407407407|
| 0.06365348399246705|0.7412280701754386|
+--------------------+-------------------+
only showing top 20 rows
```

[31]:

```
lr_predictions = lrModel.transform(testSet)
lr_predictions.select("prediction", "label", "features").show(25)

lr_evaluator = BinaryClassificationEvaluator()
print('Test Area Under ROC', lr_evaluator.evaluate(lr_predictions))
```

```
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
+----------+-----+--------------------+
only showing top 25 rows

Test Area Under ROC 0.7324437493693866
```

[32]:

```
lr_evaluator2 = BinaryClassificationEvaluator()
accuracy = lr_evaluator2.evaluate(lr_predictions)
print("Accuracy = %s" % (accuracy))
print("Test Error = %s" % (1.0 - accuracy))
```

```
Accuracy = 0.7324437493693866
Test Error = 0.2675562506306134
```

1.0.7 4.2 Decision Tree

[34]:

```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator

dt2 = DecisionTreeClassifier().setLabelCol("label").setFeaturesCol("features").
 ↪setMaxDepth(8)

dtModel2 = dt2.fit(trainSet)

# make predictions on the test data
dt_predictions2 = dtModel2.transform(testSet)
dt_predictions2.select("prediction", "label", "features").show(25)

# evaluate the model
dt_evaluator2 = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(dt_evaluator2.evaluate(dt_predictions2,␣
 ↪{dt_evaluator2.metricName: "areaUnderROC"})))
```

```
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
|       0.0|    1|(24,[0,1,2,3,4,5,...|
|       0.0|    0|(24,[0,1,2,3,4,5,...|
+----------+-----+--------------------+
only showing top 25 rows

Test Area Under ROC: 0.37598343203226675
```

1.0.8 4.3 Random Forest

[35]:

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator

rfClassifier = RandomForestClassifier().setLabelCol("label").
 ↪setFeaturesCol("features")

trainedModel = rfClassifier.fit(trainSet)

# make predictions on the test data
rf_predictions = trainedModel.transform(testSet)
rf_predictions.select("prediction", "label", "features").show(25)

# evaluate random forest model
rf_evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(rf_evaluator.evaluate(rf_predictions,␣
 ↪{rf_evaluator.metricName: "areaUnderROC"})))
```

```
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
|       0.0|    1|(24,[0,1,2,3,4,5,…|
|       0.0|    0|(24,[0,1,2,3,4,5,…|
+----------+-----+--------------------+
only showing top 25 rows

Test Area Under ROC: 0.7687806898059399
```

[36]:

```
rf_evaluator2 = BinaryClassificationEvaluator(labelCol="label")
rf_accuracy = rf_evaluator2.evaluate(rf_predictions)
print("Accuracy = %s" % (rf_accuracy))
print("Test Error = %s" % (1.0 - rf_accuracy))
```

```
Accuracy = 0.7687806898059399
Test Error = 0.23121931019406006
```