ex51

August 17, 2022

[ ]:

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from pyspark.ml import Pipeline, PipelineModel
from pyspark.mllib.linalg import Vector
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression
from pyspark.sql.types import * #questo per definire bene il tipo nelle UDF

conf = SparkConf().setAppName("ex51")
sc = SparkContext(conf=conf)
ssql = SparkSession.builder.getOrCreate()
```

[2]:

```
inputPathLabels = "data/Ex51/data/trainingData.csv"
inputPathUnlabeld = "data/Ex51/data/unlabeledData.csv"
outputPath = "out51/"
```

[3]:

```
label_data = ssql.read.load(
    inputPathLabels,
    format="csv",
    header=True,
    inferSchema=True
)

no_label_data = ssql.read.load(
    inputPathUnlabeld,
    format="csv",
    header=True,
    inferSchema=True
)
```

[4]:

```
label_data.show(), label_data.printSchema()
```

```
+-----+--------------------+
|label|                text|
+-----+--------------------+
|    1|The Spark system ...|
|    1|Spark is a new di...|
|    0|Turin is a beauti...|
|    0|Turin is in the n…|
+-----+--------------------+

root
 |-- label: integer (nullable = true)
 |-- text: string (nullable = true)
```

```
[4]: (None, None)
```

```
[5]: no_label_data.show(), no_label_data.printSchema()
```

```
+-----+--------------------+
|label|                text|
+-----+--------------------+
| null|Spark performs be…|
| null|Comparison betwee…|
| null|Turin is in Piedmont|
+-----+--------------------+

root
 |-- label: string (nullable = true)
 |-- text: string (nullable = true)
```

```
[5]: (None, None)
```

```
[ ]: def countWords(text):
         return len(text.split(" "))

     def isSpark(text):
         return text.lower().find("spark")>=0

     ssql.udf.register("wordsCount", countWords, IntegerType())
     ssql.udf.register("isSpark", isSpark, BooleanType())
```

```
[8]: featuresDF = label_data.selectExpr("label", "text", "wordsCount(text) AS␣
       ↪wordsInText", "isSpark(text) as containsSpark")
```

```
[9]: featuresDF.show(), featuresDF.printSchema()
```

```
+-----+--------------------+-----------+-------------+
|label|                text|wordsInText|containsSpark|
+-----+--------------------+-----------+-------------+
|    1|The Spark system …|          7|         true|
|    1|Spark is a new di…|          6|         true|
|    0|Turin is a beauti…|          5|        false|
|    0|Turin is in the n…|          8|        false|
+-----+--------------------+-----------+-------------+

root
 |-- label: integer (nullable = true)
 |-- text: string (nullable = true)
 |-- wordsInText: integer (nullable = true)
 |-- containsSpark: boolean (nullable = true)
```

```
[9]: (None, None)
```

```
[13]: #definisco una pipeline per effettuare la classificazione
      assembler = VectorAssembler(inputCols=["wordsInText", "containsSpark"],␣
       ↪outputCol="features")
      lr = LogisticRegression().setMaxIter(10).setRegParam(0.01)
      pipeline = Pipeline().setStages([assembler, lr])
      classificationModel = pipeline.fit(featuresDF)
```

```
[12]: formattedNoLabelDataDF = no_label_data.selectExpr("label", "text",␣
       ↪"wordsCount(text) AS wordsInText", "isSpark(text) as containsSpark")
      formattedNoLabelDataDF.show(), formattedNoLabelDataDF.printSchema()
```

```
+-----+--------------------+-----------+-------------+
|label|                text|wordsInText|containsSpark|
+-----+--------------------+-----------+-------------+
| null|Spark performs be…|          5|         true|
| null|Comparison betwee…|          5|         true|
| null|Turin is in Piedmont|          4|        false|
+-----+--------------------+-----------+-------------+

root
 |-- label: string (nullable = true)
 |-- text: string (nullable = true)
 |-- wordsInText: integer (nullable = true)
 |-- containsSpark: boolean (nullable = true)
```

```
[12]: (None, None)
```

```
[14]: predictionDF = classificationModel.transform(formattedNoLabelDataDF)
      predictionDF.show(), predictionDF.printSchema()
```

```
+-----+--------------------+-----------+-------------+---------+----------------
----+--------------------+----------+
|label|                text|wordsInText|containsSpark| features|
rawPrediction|         probability|prediction|
+-----+--------------------+-----------+------------+---------+----------------
----+--------------------+----------+
| null|Spark performs be…|          5|
true|[5.0,1.0]|[-3.1328695876505…|[0.04177159569658…|       1.0|
| null|Comparison betwee…|          5|
true|[5.0,1.0]|[-3.1328695876505…|[0.04177159569658…|       1.0|
| null|Turin is in Piedmont|          4|
false|[4.0,0.0]|[3.13286958765052…|[0.95822840430341…|       0.0|
+-----+--------------------+-----------+------------+---------+----------------
----+--------------------+----------+

root
 |-- label: string (nullable = true)
 |-- text: string (nullable = true)
 |-- wordsInText: integer (nullable = true)
 |-- containsSpark: boolean (nullable = true)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

[14]: (None, None)

[15]:

```
finalDF = predictionDF.select("text", "prediction")
finalDF.show(), finalDF.printSchema()
```

```
+--------------------+----------+
|                text|prediction|
+--------------------+----------+
|Spark performs be…|       1.0|
|Comparison betwee…|       1.0|
|Turin is in Piedmont|       0.0|
+--------------------+----------+

root
 |-- text: string (nullable = true)
 |-- prediction: double (nullable = false)
```

[15]: (None, None)

[16]:

```
finalDF.write.csv(outputPath, header=True)
```

[ ]:

```
```