PKA-ENTITY-RESOLUTION

December 11, 2023

[1]:
```
import pyspark
import os
import sys
from pyspark import SparkContext
os.environ['PYSPARK_PYTHON'] = sys.executable
os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable

from pyspark.sql import SparkSession
```

[2]:
```
spark = SparkSession.builder.config("spark.driver.memory", "16g").
  ↪appName('chapter_2').getOrCreate()
```

0.0.1 Setting Up Our Data

From the shell:

```
$ mkdir linkage
$ cd linkage/
$ curl -L -o donation.zip https://bit.ly/1Aoywaq
$ unzip donation.zip
$ unzip 'block_*.zip'
```

[3]:
```
prev = spark.read.csv("data/linkage/donation/block_1/block_1.csv")

prev
```

[3]:
```
DataFrame[_c0: string, _c1: string, _c2: string, _c3: string, _c4: string, _c5:
string, _c6: string, _c7: string, _c8: string, _c9: string, _c10: string, _c11:
string]
```

[4]:
```
prev.show(2)
```

```
+-----+-----+-----------------+-----------+-----------+-----------+-------+--
----+------+------+-------+--------+
|  _c0|  _c1|              _c2|        _c3|        _c4|        _c5|    _c6|
_c7|   _c8|   _c9|   _c10|    _c11|
+-----+-----+-----------------+-----------+-----------+-----------+-------+--
----+------+------+-------+--------+
| id_1| id_2|     cmp_fname_c1|cmp_fname_c2|cmp_lname_c1|cmp_lname_c2|cmp_sex|cm
```

```
p_bd|cmp_bm|cmp_by|cmp_plz|is_match|
|37291|53113|0.833333333333333|           ?|           1|           ?|      1|
1|     1|     1|      0|    TRUE|
+-----+-----+-----------------+------------+------------+------------+-------+--
----+------+------+-------+--------+
only showing top 2 rows
```

[5]:
```
parsed = spark.read.option("header", "true").option("nullValue", "?").\
          option("inferSchema", "true").csv("data/linkage/donation/block_1/
 ↪block_1.csv")
```

0.0.2 Analyzing Data with the DataFrame API

[6]:
```
parsed.printSchema()

parsed.show(5)
```

```
root
 |-- id_1: integer (nullable = true)
 |-- id_2: integer (nullable = true)
 |-- cmp_fname_c1: double (nullable = true)
 |-- cmp_fname_c2: double (nullable = true)
 |-- cmp_lname_c1: double (nullable = true)
 |-- cmp_lname_c2: double (nullable = true)
 |-- cmp_sex: integer (nullable = true)
 |-- cmp_bd: integer (nullable = true)
 |-- cmp_bm: integer (nullable = true)
 |-- cmp_by: integer (nullable = true)
 |-- cmp_plz: integer (nullable = true)
 |-- is_match: boolean (nullable = true)

+-----+-----+-----------------+------------+------------+------------+-------+--
----+------+------+-------+--------+
| id_1| id_2|     cmp_fname_c1|cmp_fname_c2|cmp_lname_c1|cmp_lname_c2|cmp_sex|cm
p_bd|cmp_bm|cmp_by|cmp_plz|is_match|
+-----+-----+-----------------+------------+------------+------------+-------+--
----+------+------+-------+--------+
|37291|53113|0.833333333333333|        null|         1.0|        null|      1|
1|     1|     1|      0|    true|
|39086|47614|              1.0|        null|         1.0|        null|      1|
1|     1|     1|      1|    true|
|70031|70237|              1.0|        null|         1.0|        null|      1|
1|     1|     1|      1|    true|
|84795|97439|              1.0|        null|         1.0|        null|      1|
1|     1|     1|      1|    true|
|36950|42116|              1.0|        null|         1.0|         1.0|      1|
1|     1|     1|      1|    true|
```

```
+-----+-----+----------------+-----------+-----------+-----------+-------+--
----+------+------+-------+--------+
only showing top 5 rows
```

[7]:
```
parsed.count()
```

[7]:
```
574913
```

[8]:
```
parsed.cache()
```

[8]:
```
DataFrame[id_1: int, id_2: int, cmp_fname_c1: double, cmp_fname_c2: double,
cmp_lname_c1: double, cmp_lname_c2: double, cmp_sex: int, cmp_bd: int, cmp_bm:
int, cmp_by: int, cmp_plz: int, is_match: boolean]
```

[9]:
```
from pyspark.sql.functions import col

parsed.groupBy("is_match").count().orderBy(col("count").desc()).show()
```

```
+--------+------+
|is_match| count|
+--------+------+
|   false|572820|
|    true|  2093|
+--------+------+
```

[10]:
```
parsed.createOrReplaceTempView("linkage")
```

[11]:
```
spark.sql("""
  SELECT is_match, COUNT(*) cnt
  FROM linkage
  GROUP BY is_match
  ORDER BY cnt DESC
""").show()
```

```
+--------+------+
|is_match|   cnt|
+--------+------+
|   false|572820|
|    true|  2093|
+--------+------+
```

0.0.3 Fast Summary Statistics for DataFrames

[12]:
```
summary = parsed.describe()
```

[13]:
```
summary.select("summary", "cmp_fname_c1", "cmp_fname_c2").show()
```

```
+-------+------------------+------------------+
|summary|      cmp_fname_c1|      cmp_fname_c2|
+-------+------------------+------------------+
|  count|            574811|             10325|
|   mean|0.7127592938253411|0.8977586763518969|
| stddev|0.3889286452463531|0.2742577520430532|
|    min|               0.0|               0.0|
|    max|               1.0|               1.0|
+-------+------------------+------------------+
```

[14]:
```
matches = parsed.where("is_match = true")
match_summary = matches.describe()

misses = parsed.filter(col("is_match") == False)
miss_summary = misses.describe()
```

0.0.4 PIvoting and Reshaping DataFrames

[15]:
```
summary_p = summary.toPandas()
```

[16]:
```
summary_p.head()
...
summary_p.shape
...
```

[16]: (5, 12)

[17]:
```
summary_p = summary_p.set_index('summary').transpose().reset_index()
...
summary_p = summary_p.rename(columns={'index':'field'})
...
summary_p = summary_p.rename_axis(None, axis=1)
...
summary_p.shape
```

[17]: (11, 6)

[18]:
```
summaryT = spark.createDataFrame(summary_p)
...
summaryT
```

[18]: DataFrame[field: string, count: string, mean: string, stddev: string, min: string, max: string]

[19]:
```
summaryT.printSchema()
```

```
root
 |-- field: string (nullable = true)
```

```
 |-- count: string (nullable = true)
 |-- mean: string (nullable = true)
 |-- stddev: string (nullable = true)
 |-- min: string (nullable = true)
 |-- max: string (nullable = true)
```

[20]:
```
from pyspark.sql.types import DoubleType
for c in summaryT.columns:
    if c == 'field':
        continue
    summaryT = summaryT.withColumn(c, summaryT[c].cast(DoubleType()))
...
summaryT.printSchema()
```

```
root
 |-- field: string (nullable = true)
 |-- count: double (nullable = true)
 |-- mean: double (nullable = true)
 |-- stddev: double (nullable = true)
 |-- min: double (nullable = true)
 |-- max: double (nullable = true)
```

[21]:
```
from pyspark.sql import DataFrame
from pyspark.sql.types import DoubleType

def pivot_summary(desc):
    # convert to pandas dataframe
    desc_p = desc.toPandas()
    # transpose
    desc_p = desc_p.set_index('summary').transpose().reset_index()
    desc_p = desc_p.rename(columns={'index':'field'})
    desc_p = desc_p.rename_axis(None, axis=1)
    # convert to Spark dataframe
    descT = spark.createDataFrame(desc_p)
    # convert metric columns to double from string
    for c in descT.columns:
        if c == 'field':
            continue
        else:
            descT = descT.withColumn(c, descT[c].cast(DoubleType()))
        return descT
```

[22]:
```
match_summaryT = pivot_summary(match_summary)
miss_summaryT = pivot_summary(miss_summary)
```

0.0.5 Joining DataFrames and Selecting Features

[23]:
```
match_summaryT.createOrReplaceTempView("match_desc")
miss_summaryT.createOrReplaceTempView("miss_desc")
spark.sql("""
  SELECT a.field, a.count + b.count total, a.mean - b.mean delta
  FROM match_desc a INNER JOIN miss_desc b ON a.field = b.field
  WHERE a.field NOT IN ("id_1", "id_2")
  ORDER BY delta DESC, total DESC
""")
```

[23]:
```
DataFrame[field: string, total: double, delta: double]
```

0.0.6 Scoring and Model Evaluation

[24]:
```
good_features = ["cmp_lname_c1", "cmp_plz", "cmp_by", "cmp_bd", "cmp_bm"]
...
sum_expression = " + ".join(good_features)
...
sum_expression
```

[24]:
```
'cmp_lname_c1 + cmp_plz + cmp_by + cmp_bd + cmp_bm'
```

[25]:
```
from pyspark.sql.functions import expr
scored = parsed.fillna(0, subset=good_features).\
                withColumn('score', expr(sum_expression)).\
                select('score', 'is_match')
...
scored.show()
```

```
+-----+--------+
|score|is_match|
+-----+--------+
|  4.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  4.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  4.0|    true|
```

```
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
|  5.0|    true|
+-----+--------+
only showing top 20 rows
```

[26]:
```
def crossTabs(scored: DataFrame, t: DoubleType) -> DataFrame:
    return  scored.selectExpr(f"score >= {t} as above", "is_match").\
          groupBy("above").pivot("is_match", ("true", "false")).\
          count()
```

[27]:
```
crossTabs(scored, 4.0).show()
```

```
+-----+----+------+
|above|true| false|
+-----+----+------+
| true|2087|    66|
|false|   6|572754|
+-----+----+------+
```

[28]:
```
crossTabs(scored, 2.0).show()
```

```
+-----+----+------+
|above|true| false|
+-----+----+------+
| true|2093| 59729|
|false|null|513091|
+-----+----+------+
```