p2-manuel-martinez-v2

March 13, 2024

1 ¡¡¡ ANTES DE COMENZAR CAMBIA EL NOMBRE DE ESTE ARCHIVO !!!

Cambia el nombre del archivo de modo que empiece con las tres siguientes letras `P2-` y vaya seguido del nombre del alumno.Para los espacios en blanco usar guiones bajos `_`

Ejemplo para un alumno:

`P2-Don_Quijote.pynb`

1.1 Instalación de librerías necesarias

[1]:
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-3.4.2/spark-3.4.2-bin-hadoop3.
↪tgz
!tar xf spark-3.4.2-bin-hadoop3.tgz
!pip install -q findspark
```

1.1.1 Cargamos el entorno

[2]:
```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.1-bin-hadoop3.2"
import findspark
findspark.init("spark-3.4.2-bin-hadoop3")# SPARK_HOME
```

1.1.2 Importamos pyspark y creamos una sesión

[3]:
```
from pyspark.sql import SparkSession
ss = SparkSession.builder.master("local[*]").getOrCreate()
```

1.1.3 Importemos los datos del fichero csv como un DataFrame

Inferimos el tipo de datos y comprobamos que sea correcto

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Data set http://archive.ics.uci.edu/ml/datasets/Adult

Datos: https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

[4]:
```
# Descargamos el fichero de datos (os lo subo a blackboard por si acaso)
import requests
adult_data = requests.get('https://archive.ics.uci.edu/ml/
 ↪machine-learning-databases/adult/adult.data')
with open("adult.data", "w") as f:
    f.
 ↪write("age,workclass,fnlwgt,education,education_num,marital_status,occupation,relationship,
    f.write(adult_data.content.decode("utf-8").replace(' ',''))
```

[5]:
```
# Cargamos los datos
df = ss.read.csv("adult.data", inferSchema=True, header=True, nullValue='?',␣
 ↪nanValue='?')
# Meto algunos valores no definidos extra
df = df.replace(78,None, ('age'))
print("Primera linea")
print(df.first())
print("Tipos de datos")
print(df.dtypes)
```

```
Primera linea
Row(age=39, workclass='State-gov', fnlwgt=77516, education='Bachelors',
education_num=13, marital_status='Never-married', occupation='Adm-clerical',
relationship='Not-in-family', race='White', sex='Male', capital_gain=2174,
capital_loss=0, hours_per_week=40, native_country='United-States',
salary='<=50K')
Tipos de datos
[('age', 'int'), ('workclass', 'string'), ('fnlwgt', 'int'), ('education',
'string'), ('education_num', 'int'), ('marital_status', 'string'),
('occupation', 'string'), ('relationship', 'string'), ('race', 'string'),
('sex', 'string'), ('capital_gain', 'int'), ('capital_loss', 'int'),
('hours_per_week', 'int'), ('native_country', 'string'), ('salary', 'string')]
```

## DataFrame

Aquí tenemos una muestra de nuestro dataframe.

La función show te muestra los datos de un dataframe.

[6]:
```
df.show()
```

```
+---+----------------+------+------------+-------------+--------------------+---
--------------+-------------+------------------+------+------------+------------
+--------------+--------------+------+
|age|       workclass|fnlwgt|   education|education_num|      marital_status|
occupation| relationship|              race|
sex|capital_gain|capital_loss|hours_per_week|native_country|salary|
+---+----------------+------+------------+-------------+--------------------+---
--------------+-------------+------------------+------+------------+------------
```

```
+--------------+-------------+------+
| 39|       State-gov| 77516|   Bachelors|           13|       Never-married|
Adm-clerical|Not-in-family|            White|  Male|        2174|           0|
40| United-States| <=50K|
| 50|Self-emp-not-inc| 83311|   Bachelors|           13|  Married-civ-spouse|
Exec-managerial|      Husband|            White|  Male|           0|
0|            13| United-States| <=50K|
| 38|         Private|215646|     HS-grad|            9|
Divorced|Handlers-cleaners|Not-in-family|            White|  Male|           0|
0|            40| United-States| <=50K|
| 53|         Private|234721|        11th|            7|  Married-civ-
spouse|Handlers-cleaners|      Husband|            Black|  Male|           0|
0|            40| United-States| <=50K|
| 28|         Private|338409|   Bachelors|           13|  Married-civ-spouse|
Prof-specialty|         Wife|            Black|Female|           0|
0|            40|          Cuba| <=50K|
| 37|         Private|284582|     Masters|           14|  Married-civ-spouse|
Exec-managerial|         Wife|            White|Female|           0|
0|            40| United-States| <=50K|
| 49|         Private|160187|         9th|            5|Married-spouse-ab…|
Other-service|Not-in-family|            Black|Female|           0|           0|
16|       Jamaica| <=50K|
| 52|Self-emp-not-inc|209642|     HS-grad|            9|  Married-civ-spouse|
Exec-managerial|      Husband|            White|  Male|           0|
0|            45| United-States|  >50K|
| 31|         Private| 45781|     Masters|           14|       Never-married|
Prof-specialty|Not-in-family|            White|Female|       14084|
0|            50| United-States|  >50K|
| 42|         Private|159449|   Bachelors|           13|  Married-civ-spouse|
Exec-managerial|      Husband|            White|  Male|        5178|
0|            40| United-States|  >50K|
| 37|         Private|280464|Some-college|           10|  Married-civ-spouse|
Exec-managerial|      Husband|            Black|  Male|           0|
0|            80| United-States|  >50K|
| 30|       State-gov|141297|   Bachelors|           13|  Married-civ-spouse|
Prof-specialty|      Husband|Asian-Pac-Islander|  Male|           0|
0|            40|         India|  >50K|
| 23|         Private|122272|   Bachelors|           13|       Never-married|
Adm-clerical|    Own-child|            White|Female|           0|           0|
30| United-States| <=50K|
| 32|         Private|205019|  Assoc-acdm|           12|       Never-married|
Sales|Not-in-family|            Black|  Male|           0|           0|
50| United-States| <=50K|
| 40|         Private|121772|   Assoc-voc|           11|  Married-civ-spouse|
Craft-repair|      Husband|Asian-Pac-Islander|  Male|           0|           0|
40|          null|  >50K|
| 34|         Private|245487|     7th-8th|            4|  Married-civ-spouse|
Transport-moving|      Husband|Amer-Indian-Eskimo|  Male|           0|
0|              45|        Mexico| <=50K|
| 25|Self-emp-not-inc|176756|     HS-grad|            9|       Never-married|
Farming-fishing|    Own-child|             White|  Male|           0|
0|              35| United-States| <=50K|
| 32|         Private|186824|     HS-grad|            9|       Never-
married|Machine-op-inspct|    Unmarried|             White|  Male|           0|
0|              40| United-States| <=50K|
| 38|         Private| 28887|        11th|            7|  Married-civ-spouse|
Sales|      Husband|             White|  Male|           0|           0|
50| United-States| <=50K|
| 43|Self-emp-not-inc|292175|     Masters|           14|            Divorced|
Exec-managerial|    Unmarried|             White|Female|           0|
0|              45| United-States|  >50K|
+---+----------------+------+------------+-------------------+---
--------------+-------------+-----------------+------+-----------+------------
+--------------+-------------+------+
only showing top 20 rows
```

Con summary podemos obtener un resumen estadistico de los datos.

```
[7]: df.summary().show()
```

```
+-------+------------------+-----------+------------------+------------+--------
---------+--------------+----------------+------------+------------------+------
+------------------+-----------------+------------------+------------+------+
|summary|               age|  workclass|            fnlwgt|   education|
education_num|marital_status|      occupation|relationship|              race|
sex|      capital_gain|    capital_loss|
hours_per_week|native_country|salary|
+-------+------------------+-----------+------------------+------------+--------
---------+--------------+----------------+------------+------------------+------
+------------------+-----------------+------------------+------------+------+
|  count|             32538|      30725|             32561|       32561|
32561|         32561|           30718|       32561|             32561| 32561|
32561|             32561|            32561|         31978| 32561|
|   mean| 38.55378326879341|       null|189778.36651208502|        null|
10.0806793403151|          null|            null|        null|
null|  null|1077.6488437087312| 87.303829734959|40.437455852092995|
null|  null|
| stddev|13.604917560434425|       null|105549.97769702227|
null|2.572720332067397|           null|            null|        null|
null|  null| 7385.292084840354|402.960218649002|12.347428681731838|
null|  null|
|    min|                17|Federal-gov|             12285|        10th|
1|       Divorced|    Adm-clerical|     Husband|Amer-Indian-Eskimo|Female|
0|              0|                 1|      Cambodia| <=50K|
|    25%|                28|       null|            117802|        null|
9|          null|                null|        null|              null|  null|
0|               0|                  40|           null|  null|
|    50%|                   37|       null|             178353|        null|
10|          null|                null|        null|              null|  null|
0|               0|                  40|           null|  null|
|    75%|                   48|       null|             236994|        null|
12|          null|                null|        null|              null|  null|
0|               0|                  45|           null|  null|
|    max|                   90|Without-pay|            1484705|Some-college|
16|       Widowed|Transport-moving|        Wife|             White|  Male|
99999|            4356|                  99|    Yugoslavia|  >50K|
+-------+------------------+-----------+------------------+------------+--------
---------+--------------+----------------+------------+------------------+------
+------------------+----------------+------------------+--------------+------+
```

1.1.4 Para cada ejercicio, es obligatorio almacenar el resultado en las variables definidas en cada TODO

1. Obtener solo la media y desviación tipica.

Para este ejercicio el dataframe resultante debe ser almacenado en `uno`

El dataframe resultante tiene la siguiente estructura:

```
[Row(summary='mean', age='38.55378326879341', workclass=None,
fnlwgt='189778.36651208502', education=None, education_num='10.0806793403151',
marital_status=None, occupation=None, relationship=None, race=None,
sex=None, capital_gain='1077.6488437087312', capital_loss='87.303829734959',
hours_per_week='40.437455852092995', native_country=None, salary=None), ...]
```

[8]:
```python
#TODO
from pyspark.sql.functions import col

# Calculamos la media y la desviación estándar
stats = df.describe().filter((col("summary") == "mean") | (col("summary") ==␣
 ↪"stddev"))

# Almacenamos los resultados en un nuevo DataFrame
uno = stats.select([col(c).alias(c+"_summary") for c in stats.columns])

#Muestra el resultado
uno.show()
```

```
+---------------+------------------+-----------------+------------------+-------
----------+---------------------+----------------------+------------------+-----
---------------+------------+------------+--------------------+------------------
--+----------------------+----------------------+---------------+
|summary_summary|       age_summary|workclass_summary|    fnlwgt_summary|educati
on_summary|education_num_summary|marital_status_summary|occupation_summary|relat
```

```
ionship_summary|race_summary|sex_summary|capital_gain_summary|capital_loss_summa
ry|hours_per_week_summary|native_country_summary|salary_summary|
+---------------+------------------+-----------------+------------------+-------
----------+---------------------+----------------------+------------------+-----
---------------+------------+-----------+--------------------+------------------
--+---------------------+----------------------+--------------+
|           mean| 38.55378326879341|             null|189778.36651208502|
null|     10.0806793403151|                 null|             null|
null|        null|       null|  1077.6488437087312|     87.303829734959|
40.437455852092995|                 null|          null|
|         stddev|13.604917560434425|             null|105549.97769702227|
null|    2.572720332067397|                 null|             null|
null|        null|       null|   7385.292084840354|    402.960218649002|
12.347428681731838|                 null|          null|
+---------------+------------------+-----------------+------------------+-------
----------+---------------------+----------------------+------------------+-----
---------------+------------+-----------+--------------------+------------------
--+---------------------+----------------------+--------------+
```

2. Obtener solo la media y desviación típica de la variable edad.

Para este ejercicio el dataframe resultante debe ser almacenado en `dos`

El dataframe resultante tiene la siguiente estructura:

`[Row(summary='mean', age='38.55378326879341'), ...]`

[9]:
```
#TODO
from pyspark.sql.functions import mean, stddev

# Calculamos la media y la desviación estándar de la variable 'age'
dos = df.select(mean("age").alias("mean_age"), stddev("age").
 ↪alias("stddev_age"))

#Muestra el resultado
dos.show()
```

```
+-----------------+------------------+
|         mean_age|        stddev_age|
+-----------------+------------------+
|38.55378326879341|13.604917560434425|
+-----------------+------------------+
```

3. Obtener datos estadísticos de la variable `capital_gain`.

Utilizar `describe`.

Para este ejercicio el dataframe resultante debe ser almacenado en `tres`

El dataframe resultante tiene la siguiente estructura:

```
[Row(summary='count', capital_gain='32561'),  Row(summary='mean',
capital_gain='1077.6488437087312'), ...]
```

[10]:
```
#TODO
# Aplicamos el método describe a la columna 'capital_gain'
tres = df.describe('capital_gain')

#Muestra el resultado
tres.show()
```

```
+-------+------------------+
|summary|      capital_gain|
+-------+------------------+
|  count|             32561|
|   mean|1077.6488437087312|
| stddev| 7385.292084840354|
|    min|                 0|
|    max|             99999|
+-------+------------------+
```

4. Obtener las tuplas con edad inferior a 18 años.

Utilizar `filter`.

Para este ejercicio el dataframe resultante debe ser almacenado en `cuatro`

La primera linea del dataframe resultante tiene la siguiente estructura:

```
Row(age=17, workclass=None, fnlwgt=304873.0, education='10th', education_num=6.0,
marital_status='Never-married', occupation=None, relationship='Own-child',
race='White', sex='Female', capital_gain=34095.0, capital_loss=0.0,
hours_per_week=32.0, native_country='United-States', salary='<=50K')
```

[11]:
```
#TODO
# Filtramos las filas con edad inferior a 18 años
cuatro = df.filter(df['age'] < 18)

#Muestra el resultado
cuatro.show()
```

```
+---+----------------+------+---------+-------------+-------------+------------
-----+--------------+-----+------+------------+------------+-------------+-----
---------+------+
|age|       workclass|fnlwgt|education|education_num|marital_status|
occupation|  relationship| race|
sex|capital_gain|capital_loss|hours_per_week|native_country|salary|
+---+----------------+------+---------+-------------+-------------+------------
-----+--------------+-----+------+------------+------------+-------------+-----
---------+------+
| 17|            null|304873|     10th|            6| Never-married|
null|     Own-child|White|Female|       34095|           0|            32|
United-States| <=50K|
| 17|         Private| 65368|     11th|            7| Never-married|
Sales|     Own-child|White|Female|           0|           0|            12|
United-States| <=50K|
| 17|         Private|245918|     11th|            7| Never-married|    Other-
service|     Own-child|White|  Male|           0|           0|            12|
United-States| <=50K|
| 17|         Private|191260|      9th|            5| Never-married|    Other-
service|     Own-child|White|  Male|        1055|           0|            24|
United-States| <=50K|
| 17|         Private|270942|  5th-6th|            3| Never-married|    Other-
service|Other-relative|White|  Male|           0|           0|            48|
Mexico| <=50K|
| 17|         Private| 89821|     11th|            7| Never-married|    Other-
service|     Own-child|White|  Male|           0|           0|            10|
United-States| <=50K|
| 17|         Private|175024|     11th|            7| Never-married|Handlers-
cleaners|     Own-child|White|  Male|        2176|           0|            18|
United-States| <=50K|
| 17|            null|202521|     11th|            7| Never-married|
null|     Own-child|White|  Male|           0|           0|            40|
United-States| <=50K|
| 17|            null|258872|     11th|            7| Never-married|
null|     Own-child|White|Female|           0|           0|             5|
United-States| <=50K|
| 17|         Private|211870|      9th|            5| Never-married|    Other-
service| Not-in-family|White|  Male|           0|           0|             6|
United-States| <=50K|
| 17|         Private|242718|     11th|            7| Never-married|
Sales|     Own-child|White|  Male|           0|           0|            12|
United-States| <=50K|
| 17|         Private|169658|     10th|            6| Never-married|    Other-
service|     Own-child|White|Female|           0|           0|            21|
United-States| <=50K|
| 17|            null| 80077|     11th|            7| Never-married|
null|     Own-child|White|Female|           0|           0|            20|
United-States| <=50K|
| 17|Self-emp-not-inc|368700|     11th|            7| Never-married|  Farming-
fishing|     Own-child|White|  Male|           0|           0|            10|
United-States| <=50K|
| 17|         Private|102726|     12th|            8| Never-married|    Other-
service|     Own-child|White|  Male|           0|           0|            16|
United-States| <=50K|
| 17|         Private|316929|     12th|            8| Never-married|Handlers-
cleaners|     Own-child|White|  Male|           0|           0|            20|
United-States| <=50K|
| 17|         Private|193830|     11th|            7| Never-married|
Sales|     Own-child|White|Female|           0|           0|            20|
United-States| <=50K|
| 17|          Private| 32607|     10th|           6| Never-married|  Farming-
fishing|     Own-child|White|  Male|           0|           0|            20|
United-States| <=50K|
| 17|          Private|198124|     11th|           7| Never-married|
Sales|     Own-child|White|  Male|           0|           0|            20|
United-States| <=50K|
| 17|          Private|368700|     11th|           7| Never-married|
Sales|     Own-child|White|  Male|           0|           0|            28|
United-States| <=50K|
+---+----------------+------+---------+-------------+--------------+------------
-----+--------------+-----+------+------------+------------+-------------+-----
---------+------+
only showing top 20 rows
```

5. Obtener workclass para los personas con edad inferior a 18 años.

Utilizar `select`.

Para este ejercicio el dataframe resultante debe ser almacenado en `cinco`

La primera linea del dataframe resultante tiene la siguiente estructura:

`Row(workclass=None)`

[12]:
```
#TODO
# Filtramos las filas con edad inferior a 18 años y seleccionamos la columna␣
↪'workclass'
cinco = df.filter(df['age'] < 18).select('workclass')

#Muestra el resultado
cinco.show()
```

```
+----------------+
|       workclass|
+----------------+
|            null|
|         Private|
|         Private|
|         Private|
|         Private|
|         Private|
|         Private|
|            null|
|            null|
|         Private|
|         Private|
|         Private|
|            null|
|Self-emp-not-inc|
|         Private|
|         Private|
|         Private|
|         Private|
|         Private|
|         Private|
+----------------+
only showing top 20 rows
```

Fijate que existen campos sin definir `null`. Después tendremos que asignarles algún valor para poder trabajar con ellos.

6. ¿Cuántas personas menores de 20 años tienen un salario superior a 50K dólares (>50K)?

Utilizar `count`.

Para este ejercicio el valor resultante debe ser almacenado en `seis`

[13]:
```
#TODO
# Contamos el número de personas menores de 20 años con salario >50K
seis = df.filter((df['age'] < 20) & (df['salary'] == '>50K')).count()

#Muestra el resultado
seis
```

[13]:
```
2
```

7. ¿Cuántas personas menores de 20 años hay para cada diferente estado marital (marital_status)?

Utilizar `groupby` y `agg({"*": "count"})`

Para este ejercicio el dataframe resultante debe ser almacenado en `siete`

El dataframe resultante tiene la siguiente estructura:

`[Row(marital_status='Widowed', count(1)=1), Row(marital_status='Married-spouse-absent', count(1)=4), Row(marital_status='Married-AF-spouse', count(1)=2), ...]`

[14]:
```
#TODO
from pyspark.sql.functions import count

# Contamos el número de personas menores de 20 años para cada estado marital
siete = df.filter(df['age'] < 20).groupby('marital_status').agg(count("*").
 ↪alias("count"))

#Muestra el resultado
siete.show()
```

```
+--------------------+-----+
|      marital_status|count|
+--------------------+-----+
|           Separated|    6|
|       Never-married| 1613|
|Married-spouse-ab…|    4|
|            Divorced|    7|
|             Widowed|    1|
|   Married-AF-spouse|    2|
|  Married-civ-spouse|   24|
+--------------------+-----+
```

8. ¿Existen variables con valores nulos? Obtener el conjunto de tuplas cuya edad es nula.

Probad únicamente si la variable `age` tiene valores nulos

Utilizar `isNull`

Para este ejercicio el dataframe resultante debe ser almacenado en `ocho`

La primera linea del dataframe tiene que tener una estructura similar a la siguiente.

```
Row(age=None, workclass='Private', fnlwgt=182977.0, education='HS-grad',
education_num=9.0, marital_status='Widowed', occupation='Other-service',
relationship='Not-in-family', race='Black', sex='Female', capital_gain=2964.0,
capital_loss=0.0, hours_per_week=40.0, native_country='United-States',
salary='<=50K')
```

[15]:
```
#TODO
ocho = df.filter(df['age'].isNull())

#Muestra el resultado
ocho.show()
```

```
+----+----------------+------+----------+-------------+------------------+------
-----------+-------------+------------------+------+------------+------------+--
------------+------------------+------+
| age|       workclass|fnlwgt| education|education_num|    marital_status|
occupation| relationship|              race|
sex|capital_gain|capital_loss|hours_per_week|    native_country|salary|
+----+----------------+------+----------+-------------+------------------+------
-----------+-------------+------------------+------+------------+------------+--
------------+------------------+------+
|null|         Private|182977|   HS-grad|            9|           Widowed|
Other-service|Not-in-family|             Black|Female|        2964|           0|
40|     United-States| <=50K|
|null|       Local-gov|136198| Bachelors|           13|Married-civ-spouse|
Exec-managerial|      Husband|             White|  Male|           0|
0|             15|     United-States| <=50K|
|null|            null|363134|   HS-grad|            9|           Widowed|
null|Not-in-family|            White|Female|           0|           0|
1|     United-States| <=50K|
|null|            null|165694|   Masters|           14|           Widowed|
null|Not-in-family|            White|Female|           0|           0|
15|     United-States| <=50K|
|null|Self-emp-not-inc|316261| Bachelors|           13|     Never-married|
Exec-managerial|Not-in-family|            White|  Male|       99999|
0|            20|     United-States|  >50K|
|null|            null| 27979|   HS-grad|            9|Married-civ-spouse|
null|      Husband|            White|  Male|        2228|           0|
32|     United-States| <=50K|
|null|         Private|180239|   Masters|           14|           Widowed|
Craft-repair|    Unmarried|Asian-Pac-Islander|  Male|           0|           0|
40|            South| <=50K|
|null|         Private|111189|   7th-8th|            4|     Never-
married|Machine-op-inspct|Not-in-family|            White|Female|           0|
0|            35|Dominican-Republic| <=50K|
|null|            null| 33186|   7th-8th|            4|Married-civ-spouse|
null|      Husband|            White|  Male|           0|           0|
60|     United-States| <=50K|
|null|            null| 83511|   7th-8th|            4|Married-civ-spouse|
null|      Husband|            White|  Male|           0|           0|
40|         Portugal| <=50K|
|null|            null|135839|   HS-grad|            9|           Widowed|
null|Not-in-family|            White|Female|        1086|           0|
20|     United-States| <=50K|
|null|    Self-emp-inc|188044| Bachelors|           13|Married-civ-spouse|
Exec-managerial|      Husband|            White|  Male|           0|
2392|            40|     United-States|  >50K|
|null|    Self-emp-inc|212660|      11th|            7|Married-civ-spouse|
Exec-managerial|      Husband|            White|  Male|           0|
0|            10|     United-States| <=50K|
|null|Self-emp-not-inc| 59583|   7th-8th|            4|Married-civ-spouse|
Farming-fishing|      Husband|            White|  Male|           0|
0|            25|     United-States| <=50K|
|null|    Self-emp-inc|385242| Bachelors|           13|Married-civ-spouse|
Exec-managerial|      Husband|            White|  Male|        9386|
0|            45|     United-States|  >50K|
|null|    Self-emp-inc|237294|   HS-grad|            9|           Widowed|
Sales|Not-in-family|            White|  Male|           0|           0|
45|     United-States|  >50K|
|null|            null| 91534| Bachelors|           13|Married-civ-spouse|
null|      Husband|            White|  Male|           0|           0|
3|     United-States| <=50K|
|null|            null|292019|   7th-8th|            4|Married-civ-spouse|
null|      Husband|            White|  Male|           0|           0|
20|     United-States| <=50K|
|null|            null| 74795|Assoc-acdm|           12|           Widowed|
null|Not-in-family|             White|Female|           0|           0|
4|     United-States| <=50K|
|null|         Private|105586|   5th-6th|            3|Married-civ-spouse|
Transport-moving|      Husband|Asian-Pac-Islander|  Male|           0|
0|            36|     United-States| <=50K|
+----+----------------+------+----------+-------------+------------------+------
-----------+-------------+-----------------+------+------------+------------+--
------------+-----------------+------+
only showing top 20 rows
```

9. Rellenemos los valores nulos de las variables continuas con el valor medio de cada variable.

Primero obtengamos un diccionario con las medias de estas variables

`'age','capital_gain','capital_loss','education_num','fnlwgt', 'hours_per_week'`

Para este ejercicio el diccionario resultante debe ser almacenado en `nueve`

Este tendrá una estructura similar a la siguiente:

`{'age': 12,  'capital_gain': 1231,  'capital_loss': 23,  ...}`

[16]:
```python
from pyspark.sql.functions import mean

# Calculamos el valor medio de las variables continuas
variables_continuas = ['age', 'capital_gain', 'capital_loss', 'education_num',␣
 ↪'fnlwgt', 'hours_per_week']
nueve = df.agg(*[mean(c).alias(c) for c in variables_continuas]).collect()[0].
 ↪asDict()

#Muestra el resultado
nueve
```

[16]:
```
{'age': 38.55378326879341,
 'capital_gain': 1077.6488437087312,
 'capital_loss': 87.303829734959,
 'education_num': 10.0806793403151,
 'fnlwgt': 189778.36651208502,
 'hours_per_week': 40.437455852092995}
```

Si los valores de las medias son actualmente strings, tendremos que convertirlos a enteros, por ejemplo. Lo hago por vosotros. También quito el atributo summary si está en el diccionario ya que no nos sive después.

[17]:
```python
if 'summary' in nueve:
    del(nueve['summary'])
nueve = {k: int(float(v)) for k, v in nueve.items()}
nueve
```

```
[17]: {'age': 38,
       'capital_gain': 1077,
       'capital_loss': 87,
       'education_num': 10,
       'fnlwgt': 189778,
       'hours_per_week': 40}
```

10. Rellenar ahora el dataframe con las medias almacenadas en el diccionario.

Utilizar `fillna` o `na.fill`

Para este ejercicio el dataframe resultante debe ser almacenado en `df_con_medias`

```
[18]: #TODO
df_con_medias= df.fillna(nueve)

#Si mostrasemos ahora las tuplas del dataframe df_con_medias con valores nulos␣
 ↪para age no debería mostrar ninguna.
df_con_medias.filter(df_con_medias['age'].isNull()).show()
```

```
+---+---------+------+---------+-------------+--------------+----------+--------
----+----+---+------------+------------+--------------+--------------+------+
|age|workclass|fnlwgt|education|education_num|marital_status|occupation|relation
ship|race|sex|capital_gain|capital_loss|hours_per_week|native_country|salary|
+---+---------+------+---------+-------------+--------------+----------+--------
----+----+---+------------+------------+--------------+--------------+------+
+---+---------+------+---------+-------------+--------------+----------+--------
----+----+---+------------+------------+--------------+--------------+------+
```

11. Ahora eliminar aquellas tuplas que tengan aún tengan valores nulos. El dataframe resultante debe ser almacenado en `df_limpio`. Partir del dataframe `df_con_medias`.

Utilizar `dropna` o `na.drop`

¿Cuántas tuplas se han eliminado?

Para este ejercicio el número de tuplas eliminadas debe ser almacenado en `once`

```
[19]: #TODO
df_limpio = df_con_medias.dropna()
once = df_con_medias.count() - df_limpio.count()

once
```

```
[19]: 2399
```

A continuación transformaremos el data frame de modo que tenga un atributo label con el atributo salary y otra columna features con el resto de atributos almacenados en un vector de tipo dense.

Utilizamos StringIndexer para convertir las variables nominales en variables númericas que representan cada uno de los posibles valores. Las variables nominales son las siguientes:

‘salary’,‘workclass’, ‘education’, ‘marital_status’, ‘occupation’, ‘relationship’, ‘race’, ‘sex’, ‘native_country’

La clase Pipeline puede ser útil para poder ejecutar varias etapas de preprocesado o modelos en secuencia.

[20]:
```
from pyspark.ml import Pipeline
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import StringIndexer

cols_to_index = ['salary','workclass', 'education', 'marital_status',␣
 ↪'occupation', 'relationship', 'race', 'sex', 'native_country']
indexed_cols = [col+"_index" for col in cols_to_index]
cols_to_scale = ['age','capital_gain','capital_loss','education_num','fnlwgt',␣
 ↪'hours_per_week']

indexers = [StringIndexer(inputCol=col, outputCol=col+"_index").fit(df_limpio)␣
 ↪for col in cols_to_index]



pipeline = Pipeline(stages=indexers)
df_indexado = pipeline.fit(df_limpio).transform(df_limpio).
 ↪select(indexed_cols+cols_to_scale).rdd.map(lambda x: (x[0], Vectors.
 ↪dense(x[1:]))).toDF(('label','sinNormFeatures'))
```

Ahora no existe variables nominales con valores definidos como strings. Esto nos permite trabajar con una variedad de modelos de aprendizaje automático mucho mayor.

[21]:
```
# Muestra como están guardados los datos
df_indexado.first()
```

[21]:
```
Row(label=0.0, sinNormFeatures=DenseVector([3.0, 2.0, 1.0, 3.0, 1.0, 0.0, 0.0,
0.0, 39.0, 2174.0, 0.0, 13.0, 77516.0, 40.0]))
```

12. Utilizar `StandardScaler` para escalar todas las variables y conseguir que tengan una variabilidad similar y estabilidad numérica.

Normalmente, a partir de este apartado se trabaja con el conjunto de datos separados en al menos dos subconjuntos. Un subconjunto de entrenamiento y otro de testeo. Por simplicidad, trabajaremos con un único conjunto. Continua utilizando el dataframe `df_indexado` obtenido en el ejercicio anterior.

Para este ejercicio el dataframe resultante debe ser almacenado en `df_procesado`

Tened en cuenta que el nombre de las variables a normalizar (input) es `sinNormFeatures` y el resultante (output) debe llamarse `features`

La primera linea del dataframe tiene que tener una estructura similar a la siguiente.

```
Row(label=0.0, sinNormFeatures=DenseVector([3.0, 2.0, 1.0, 3.0, 1.0, 0.0, 0.0,
0.0, 39.0, 2174.0, 0.0, 13.0, 77516.0]), features=DenseVector([1.8109, -0.1326,
0.1196, -0.2642, -0.1517, -0.3458, -0.6928, -0.2219, 0.0443, 0.1461, -0.2186,
1.1289, -1.0627]))
```

[22]:
```python
from pyspark.ml.feature import StandardScaler

#TODO
# Creamos el objeto StandardScaler
scaler = StandardScaler(inputCol="sinNormFeatures", outputCol="features",␣
 ↪withStd=True, withMean=True)

# Aplicamos el escalado al DataFrame
scaler_model = scaler.fit(df_indexado)
df_procesado = scaler_model.transform(df_indexado)

#Muestra la primera linea
df_procesado.first()
```

[22]:
```
Row(label=0.0, sinNormFeatures=DenseVector([3.0, 2.0, 1.0, 3.0, 1.0, 0.0, 0.0,
0.0, 39.0, 2174.0, 0.0, 13.0, 77516.0, 40.0]), features=DenseVector([1.8109,
-0.1326, 0.1195, -0.2643, -0.1517, -0.3459, -0.6928, -0.2219, 0.0443, 0.1461,
-0.2186, 1.1289, -1.0627, -0.0777]))
```

13. Utilizar `DecisionTreeClassifier` para crear un modelo de clasificación que sea capaz de resolver el problema propuesto.

Utilizar `BinaryClassificationEvaluator` para evaluar la precisión del modelo entrenado.

Para este ejercicio el arbol de decisión entrenado debe ser almacenado en `dt` y la precisión resultante debe ser almacenada en `precision`

[23]:
```python
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator

#TODO
# Creamos el clasificador DecisionTreeClassifier
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features")

# Entrenamos el modelo
model = dt.fit(df_procesado)

# Hacemos predicciones en el conjunto de datos de entrenamiento
predictions = model.transform(df_procesado)

# Creamos un evaluador BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(labelCol="label",␣
 ↪rawPredictionCol="prediction", metricName="areaUnderROC")

# Evaluamos el modelo y obtenemos la precisión
precision = evaluator.evaluate(predictions)
```

```
#Muestra la precision obtenida por el modelo
print("Precisión = %g " % (precision))
```

```
Precisión = 0.725687
```

1.2 Extra (No forma parte de la evaluación, se puede sacar un 10 sin esto pero en caso de haber fallado algo, ayuda a que la calificación suba)

- Utilizar pandas para importar y procesar los datos.
- Utilizar pairplot de seaborn para mostrar las distribuciones de las variables.
- Utilizar LabelEncoder de scikit learn para gestionar las variables categoricas.
- Utilizar StandardScaler de scikit learn para normalizar las variables.
- Probar distintos modelos de clasificación de scikit learn para ver cual consigue mayor precisión de clasificación.
- El alumno puede elegir un dataset de su interés y aplicarla al menos un algoritmo de Mchine learning, discutiendo y comentando los resultados obtenidos. Hay que proporcionar un link desde donde se ha obtenido el dataset y subir una copia del mismo a blackboard

```
[ ]:
```