TP3 - SparkLab

September 28, 2021

Ahmed Guizani (Groupe 2) Tom Sinnah (Groupe 1)

0.1 1 - Installation Test

[2]:
```
import findspark
findspark.init()
```

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

0.2 2 - Analysis and Regression on Boston housing dataset

0.2.1 2.1)

[4]:
```
dataset = spark.read.csv('./Data/HousingData.csv',inferSchema=True,header=True)
```

[5]:
```
dataset.printSchema()
dataset.show()
```

```
root
 |-- CRIM: string (nullable = true)
 |-- ZN: string (nullable = true)
 |-- INDUS: string (nullable = true)
 |-- CHAS: string (nullable = true)
 |-- NOX: double (nullable = true)
 |-- RM: double (nullable = true)
 |-- AGE: string (nullable = true)
 |-- DIS: double (nullable = true)
 |-- RAD: integer (nullable = true)
 |-- TAX: integer (nullable = true)
 |-- PTRATIO: double (nullable = true)
 |-- B: double (nullable = true)
 |-- LSTAT: string (nullable = true)
 |-- MEDV: double (nullable = true)

+-------+----+-----+----+-----+-----+----+------+---+---+-------+------+-----+--
--+
|   CRIM|  ZN|INDUS|CHAS|  NOX|   RM| AGE|   DIS|RAD|TAX|PTRATIO|
B|LSTAT|MEDV|
```

```
+-------+----+-----+----+-----+-----+----+------+---+---+-------+------+-----+--
--+
|0.00632|  18| 2.31|   0|0.538|6.575|65.2|  4.09|  1|296|   15.3| 396.9|
4.98|24.0|
|0.02731|   0| 7.07|   0|0.469|6.421|78.9|4.9671|  2|242|   17.8| 396.9|
9.14|21.6|
|0.02729|   0| 7.07|   0|0.469|7.185|61.1|4.9671|  2|242|   17.8|392.83|
4.03|34.7|
|0.03237|   0| 2.18|   0|0.458|6.998|45.8|6.0622|  3|222|   18.7|394.63|
2.94|33.4|
|0.06905|   0| 2.18|   0|0.458|7.147|54.2|6.0622|  3|222|   18.7| 396.9|
NA|36.2|
|0.02985|   0| 2.18|   0|0.458| 6.43|58.7|6.0622|  3|222|   18.7|394.12|
5.21|28.7|
|0.08829|12.5| 7.87|  NA|0.524|6.012|66.6|5.5605|  5|311|   15.2|
395.6|12.43|22.9|
|0.14455|12.5| 7.87|   0|0.524|6.172|96.1|5.9505|  5|311|   15.2|
396.9|19.15|27.1|
|0.21124|12.5| 7.87|   0|0.524|5.631| 100|6.0821|  5|311|
15.2|386.63|29.93|16.5|
|0.17004|12.5| 7.87|  NA|0.524|6.004|85.9|6.5921|  5|311|   15.2|386.71|
17.1|18.9|
|0.22489|12.5| 7.87|   0|0.524|6.377|94.3|6.3467|  5|311|
15.2|392.52|20.45|15.0|
|0.11747|12.5| 7.87|   0|0.524|6.009|82.9|6.2267|  5|311|   15.2|
396.9|13.27|18.9|
|0.09378|12.5| 7.87|   0|0.524|5.889|  39|5.4509|  5|311|   15.2|
390.5|15.71|21.7|
|0.62976|   0| 8.14|   0|0.538|5.949|61.8|4.7075|  4|307|   21.0| 396.9|
8.26|20.4|
|0.63796|   0| 8.14|  NA|0.538|6.096|84.5|4.4619|  4|307|
21.0|380.02|10.26|18.2|
|0.62739|   0| 8.14|   0|0.538|5.834|56.5|4.4986|  4|307|   21.0|395.62|
8.47|19.9|
|1.05393|   0| 8.14|   0|0.538|5.935|29.3|4.4986|  4|307|   21.0|386.85|
6.58|23.1|
| 0.7842|   0| 8.14|   0|0.538| 5.99|81.7|4.2579|  4|307|
21.0|386.75|14.67|17.5|
|0.80271|   0| 8.14|   0|0.538|5.456|36.6|3.7965|  4|307|
21.0|288.99|11.69|20.2|
| 0.7258|   0| 8.14|   0|0.538|5.727|69.5|3.7965|  4|307|
21.0|390.95|11.28|18.2|
+-------+----+-----+----+-----+-----+----+------+---+---+-------+------+-----+--
--+
only showing top 20 rows
```

0.2.2 2.2)

[7]:
```
from pyspark.ml.feature import VectorAssembler # Permet d'assembler toutes les␣
↪features en un seul vecteur
from pyspark.ml.regression import LinearRegression
```

[8]:
```
# Conversion des colonnes du dataset au bon type de données
from pyspark.sql.types import DoubleType, IntegerType, StringType, LongType

dataset = dataset.withColumn('CRIM',dataset["CRIM"].cast(DoubleType()))
dataset = dataset.withColumn('ZN',dataset["ZN"].cast(DoubleType()))
dataset = dataset.withColumn('INDUS',dataset["INDUS"].cast(DoubleType()))
dataset = dataset.withColumn('CHAS',dataset["CHAS"].cast(IntegerType()))
dataset = dataset.withColumn('AGE',dataset["AGE"].cast(DoubleType()))
dataset = dataset.withColumn('LSTAT',dataset["LSTAT"].cast(DoubleType()))

# On crée l'objet assembler
assembler =␣
↪VectorAssembler(inputCols=['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','P
                           outputCol='Attributes',
                           handleInvalid="skip") #Pour passer les valeurs 'NUL'

output = assembler.transform(dataset)
finilized_data = output.select("Attributes","medv") # medv est la colonne à␣
↪prédire
finilized_data.show()
```

```
+--------------------+----+
|          Attributes|medv|
+--------------------+----+
|[0.00632,18.0,2.3…|24.0|
|[0.02731,0.0,7.07…|21.6|
|[0.02729,0.0,7.07…|34.7|
|[0.03237,0.0,2.18…|33.4|
|[0.02985,0.0,2.18…|28.7|
|[0.14455,12.5,7.8…|27.1|
|[0.21124,12.5,7.8…|16.5|
|[0.22489,12.5,7.8…|15.0|
|[0.11747,12.5,7.8…|18.9|
|[0.09378,12.5,7.8…|21.7|
|[0.62976,0.0,8.14…|20.4|
|[0.62739,0.0,8.14…|19.9|
|[1.05393,0.0,8.14…|23.1|
|[0.7842,0.0,8.14,…|17.5|
|[0.80271,0.0,8.14…|20.2|
|[0.7258,0.0,8.14,…|18.2|
|[1.25179,0.0,8.14…|13.6|
|[0.85204,0.0,8.14…|19.6|
```

```
|[1.23247,0.0,8.14…|15.2|
|[0.98843,0.0,8.14…|14.5|
+--------------------+----+
only showing top 20 rows
```

0.2.3 2.3)

[9]:
```
# Training & testing data
train_data, test_data = finilized_data.randomSplit([0.8,0.2])
```

0.2.4 2.4)

[10]:
```
regressor = LinearRegression(featuresCol='Attributes',labelCol='medv')

# Learn to fit the model from training set
regressor = regressor.fit(train_data)

#To predict the prices on testing set
pred = regressor.evaluate(test_data)

#Predict the model
pred.predictions.show()
```

```
+--------------------+----+------------------+
|          Attributes|medv|        prediction|
+--------------------+----+------------------+
|[0.00906,90.0,2.9…|32.2| 32.55546192299896|
|[0.01432,100.0,1…|31.6|33.056525872951276|
|[0.01501,80.0,2.0…|24.5|28.140543321953267|
|[0.02498,0.0,1.89…|16.5|21.825144021686757|
|[0.0315,95.0,1.47…|34.9|30.755995147038526|
|[0.03237,0.0,2.18…|33.4|29.034406525979332|
|[0.04113,25.0,4.8…|28.0|28.785525243360897|
|[0.04297,52.5,5.3…|24.8|27.579173412944513|
|[0.05083,0.0,5.19…|22.2|22.821448099282872|
|[0.05497,0.0,5.19…|19.0|21.825620705805406|
|[0.05515,33.0,2.1…|36.1|34.459238270588855|
|[0.0566,0.0,3.41,…|23.6|30.081769304881703|
|[0.06047,0.0,2.46…|29.6|24.903651291847577|
|[0.06129,20.0,3.3…|46.0| 40.09850491253814|
|[0.06642,0.0,4.05…|29.9|30.973019245264837|
|[0.0686,0.0,2.89,…|33.2| 32.61284840503428|
|[0.07165,0.0,25.6…|20.3| 21.79809050170061|
|[0.07875,45.0,3.4…|32.0| 32.98920557801274|
|[0.09103,0.0,2.46…|37.9| 33.36082480568782|
|[0.09849,0.0,25.6…|18.8|19.918069069665453|
+--------------------+----+------------------+
```

```
only showing top 20 rows
```

0.2.5 2.5)

[11]:
```
#coefficient of the regression model
coeff = regressor.coefficients

# X and Y intercept
intr = regressor.intercept

print('The coefficient of the model is : %a' %coeff)
print('The intercept of the model is   : %f' %intr)
```

```
The coefficient of the model is : DenseVector([-0.0974, 0.0468, -0.0201, 2.1987,
-13.0049, 4.9019, -0.0235, -1.3801, 0.3206, -0.0146, -0.7962, 0.0126, -0.3853])
The intercept of the model is   : 23.479128
```

0.2.6 2.6)

[12]:
```
from pyspark.ml.evaluation import RegressionEvaluator

eval = RegressionEvaluator(labelCol='medv',
                           predictionCol='prediction',
                           metricName='rmse')
```

[13]:
```
# Root Mean Square Error
rmse = eval.evaluate(pred.predictions)

# Mean Square Error
mse = eval.evaluate(pred.predictions,{eval.metricName:"mse"})

# Mean Absolute Error
mae = eval.evaluate(pred.predictions,{eval.metricName:"mae"})

# r2 - coefficient de determination
r2 = eval.evaluate(pred.predictions,{eval.metricName:"r2"})

print("RMSE : %.3f" %rmse)
print("MSE  : %.3F" %mse)
print("MAE  : %.3f" %mae)
print("R2   : %.3f" %r2)
```

```
RMSE : 5.629
MSE  : 31.682
MAE  : 3.670
R2   : 0.631
```

0.2.7 2.7)

[14]:
```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

# Trains a k-means model
kmeans = KMeans(featuresCol='Attributes').setK(2).setSeed(1)
model = kmeans.fit(finilized_data)

# Make a prediction
predictions = model.transform(finilized_data)

predictions.show()
```

```
+--------------------+----+----------+
|          Attributes|medv|prediction|
+--------------------+----+----------+
|[0.00632,18.0,2.3…|24.0|         0|
|[0.02731,0.0,7.07…|21.6|         0|
|[0.02729,0.0,7.07…|34.7|         0|
|[0.03237,0.0,2.18…|33.4|         0|
|[0.02985,0.0,2.18…|28.7|         0|
|[0.14455,12.5,7.8…|27.1|         0|
|[0.21124,12.5,7.8…|16.5|         0|
|[0.22489,12.5,7.8…|15.0|         0|
|[0.11747,12.5,7.8…|18.9|         0|
|[0.09378,12.5,7.8…|21.7|         0|
|[0.62976,0.0,8.14…|20.4|         0|
|[0.62739,0.0,8.14…|19.9|         0|
|[1.05393,0.0,8.14…|23.1|         0|
|[0.7842,0.0,8.14,…|17.5|         0|
|[0.80271,0.0,8.14…|20.2|         0|
|[0.7258,0.0,8.14,…|18.2|         0|
|[1.25179,0.0,8.14…|13.6|         0|
|[0.85204,0.0,8.14…|19.6|         0|
|[1.23247,0.0,8.14…|15.2|         0|
|[0.98843,0.0,8.14…|14.5|         0|
+--------------------+----+----------+
only showing top 20 rows
```

0.3 3 - Churn analysis in Spark (Churn = CHange and tURN)

0.3.1 3.1)

[15]:
```
import pyspark.sql.functions as F
datasetCalls = spark.read.csv('./Data/CallsData.csv',header=True)
datasetContract = spark.read.csv('./Data/ContractData.csv',header=True)
```

```
print("Contract Schema: ")
datasetContract.printSchema()
print("Calls Schema: ")
datasetCalls.printSchema()

# On effectue une jointure en utilisant la colonne 'Phone'
datasetCalls_ = datasetCalls.withColumn('Phone',F.
 ↪monotonically_increasing_id()).drop('Area Code')
datasetContract_ = datasetContract.withColumn('Phone',F.
 ↪monotonically_increasing_id()).drop('State')

data = datasetCalls_.join(datasetContract_,'Phone','inner')
print('Contract + Calls Schema:')
data.printSchema()
```

```
Contract Schema:
root
 |-- Account Length: string (nullable = true)
 |-- Churn: string (nullable = true)
 |-- Int'l Plan: string (nullable = true)
 |-- VMail Plan: string (nullable = true)
 |-- State: string (nullable = true)
 |-- Area Code: string (nullable = true)
 |-- Phone: string (nullable = true)

Calls Schema:
root
 |-- VMail Message: string (nullable = true)
 |-- Day Mins: string (nullable = true)
 |-- Eve Mins: string (nullable = true)
 |-- Night Mins: string (nullable = true)
 |-- Intl Mins: string (nullable = true)
 |-- CustServ Calls: string (nullable = true)
 |-- Day Calls: string (nullable = true)
 |-- Day Charge: string (nullable = true)
 |-- Eve Calls: string (nullable = true)
 |-- Eve Charge: string (nullable = true)
 |-- Night Calls: string (nullable = true)
 |-- Night Charge: string (nullable = true)
 |-- Intl Calls: string (nullable = true)
 |-- Intl Charge: string (nullable = true)
 |-- Area Code: string (nullable = true)
 |-- Phone: string (nullable = true)

Contract + Calls Schema:
root
```

```
|-- Phone: long (nullable = false)
|-- VMail Message: string (nullable = true)
|-- Day Mins: string (nullable = true)
|-- Eve Mins: string (nullable = true)
|-- Night Mins: string (nullable = true)
|-- Intl Mins: string (nullable = true)
|-- CustServ Calls: string (nullable = true)
|-- Day Calls: string (nullable = true)
|-- Day Charge: string (nullable = true)
|-- Eve Calls: string (nullable = true)
|-- Eve Charge: string (nullable = true)
|-- Night Calls: string (nullable = true)
|-- Night Charge: string (nullable = true)
|-- Intl Calls: string (nullable = true)
|-- Intl Charge: string (nullable = true)
|-- Account Length: string (nullable = true)
|-- Churn: string (nullable = true)
|-- Int'l Plan: string (nullable = true)
|-- VMail Plan: string (nullable = true)
|-- Area Code: string (nullable = true)
```

0.3.2 3.2)

[16]:
```
data = data.withColumn('Phone', data['Phone'].cast(LongType()))
data = data.withColumn('VMail Message', data['VMail Message'].
 ↪cast(IntegerType()))
data = data.withColumn('Day Mins', data['Day Mins'].cast(DoubleType()))
data = data.withColumn('Eve Mins', data['Eve Mins'].cast(DoubleType()))
data = data.withColumn('Night Mins', data['Night Mins'].cast(DoubleType()))
data = data.withColumn('Intl Mins', data['Intl Mins'].cast(DoubleType()))
data = data.withColumn('CustServ Calls', data['CustServ Calls'].
 ↪cast(IntegerType()))
data = data.withColumn('Day Calls', data['Day Calls'].cast(DoubleType()))
data = data.withColumn('Day Charge', data['Day Charge'].cast(DoubleType()))
data = data.withColumn('Eve Calls', data['Eve Calls'].cast(DoubleType()))
data = data.withColumn('Eve Charge', data['Eve Charge'].cast(DoubleType()))
data = data.withColumn('Night Calls', data['Night Calls'].cast(DoubleType()))
data = data.withColumn('Night Charge', data['Night Charge'].cast(DoubleType()))
data = data.withColumn('Intl Calls', data['Intl Calls'].cast(DoubleType()))
data = data.withColumn('Intl Charge', data['Intl Charge'].cast(DoubleType()))
data = data.withColumn('Area Code', data['Area Code'].cast(IntegerType()))
data = data.withColumn('Account Length', data['Account Length'].
 ↪cast(IntegerType()))
data = data.withColumn('Churn', data['Churn'].cast(IntegerType()))
data = data.withColumn("Int'l Plan", data["Int'l Plan"].cast(IntegerType()))
data = data.withColumn('VMail Plan', data['VMail Plan'].cast(IntegerType()))
```

```
data = data.withColumn('Intl Calls', data['Intl Calls'].cast(DoubleType()))
data = data.withColumn('Intl Charge', data['Intl Charge'].cast(DoubleType()))



assembler_ = VectorAssembler(inputCols=['VMail Message',
                                        'Day Mins',
                                         'Eve Mins',
                                         'Night Mins',
                                         'Intl Mins',
                                         'CustServ Calls',
                                         'Day Calls',
                                         'Day Charge',
                                         'Eve Calls',
                                         'Eve Charge',
                                         'Night Calls',
                                         'Night Charge',
                                         'Intl Calls',
                                         'Intl Charge',
                                         'Area Code',
                                         'Account Length',
                                         "Int'l Plan",
                                          'VMail Plan',
                                         'Area Code'],
                             outputCol = 'Attributes_',
                             handleInvalid='skip')

output = assembler_.transform(data)
data_ = output.select('Attributes_','Churn')
data_.show()
```

```
+--------------------+-----+
|         Attributes_|Churn|
+--------------------+-----+
|[25.0,265.1,197.4…|    0|
|[26.0,161.6,195.5…|    0|
|[0.0,243.4,121.2,…|    0|
|[0.0,299.4,61.9,1…|    0|
|[0.0,166.7,148.3,…|    0|
|[0.0,223.4,220.6,…|    0|
|[24.0,218.2,348.5…|    0|
|[0.0,157.0,103.1,…|    0|
|[0.0,184.5,351.6,…|    0|
|[37.0,258.6,222.0…|    0|
|[0.0,129.1,228.5,…|    1|
|[0.0,187.7,163.4,…|    0|
|[0.0,128.8,104.9,…|    0|
|[0.0,156.6,247.6,…|    0|
```

```
|[0.0,120.7,307.2,…|    0|
|[0.0,332.9,317.8,…|    1|
|[27.0,196.4,280.9…|    0|
|[0.0,190.7,218.2,…|    0|
|[33.0,189.7,212.8…|    0|
|[0.0,224.4,159.5,…|    0|
+--------------------+-----+
only showing top 20 rows
```

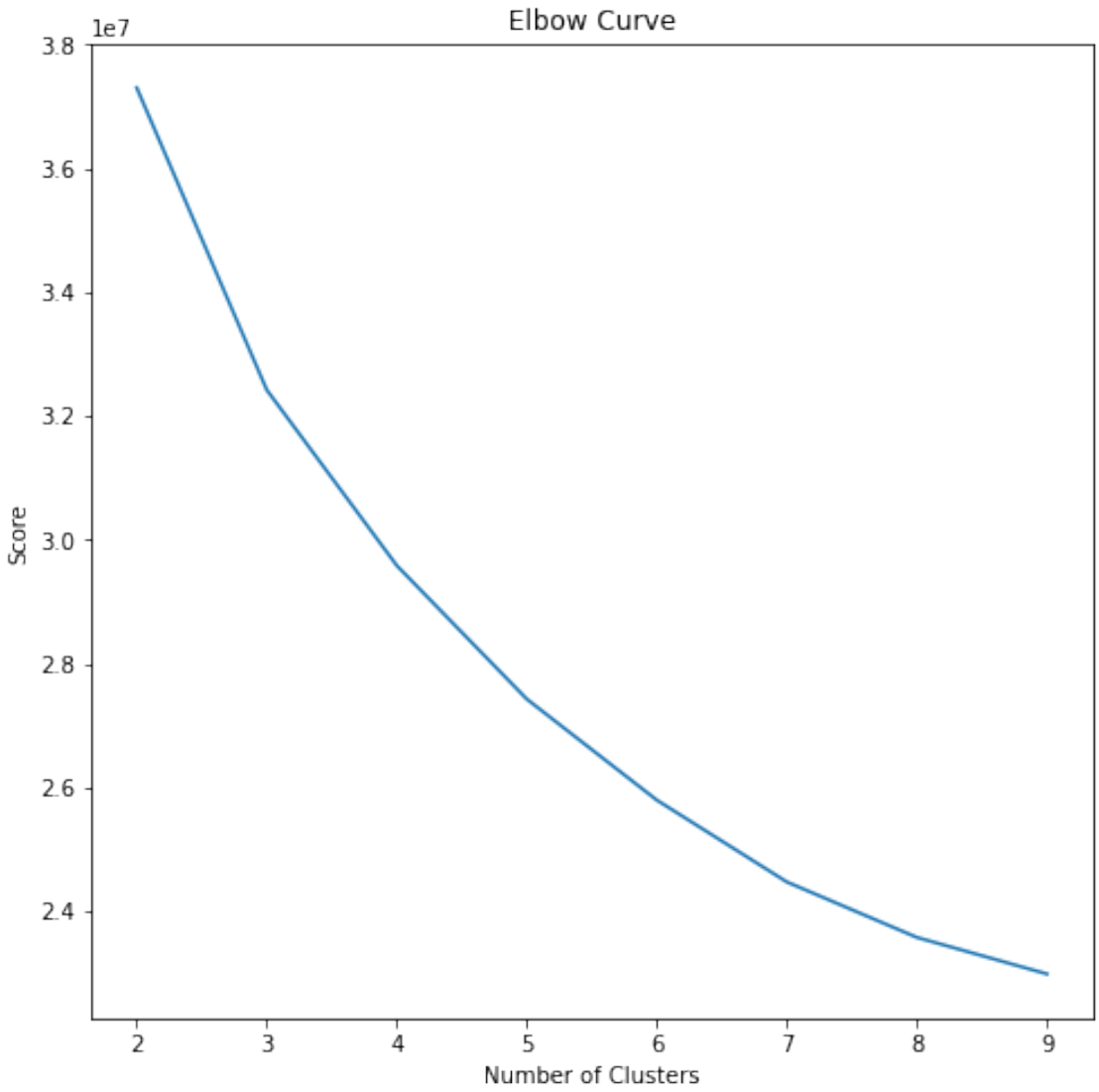
[52]:
```
import pandas as pd
import numpy as np

# Calculate cost and plot
cost = np.zeros(10)

for i in range(2,10):
    kmeans = KMeans(featuresCol='Attributes_',k=i)
    model = kmeans.fit(data_)
    cost[i] = model.summary.trainingCost

# Plot the cost
df_cost = pd.DataFrame(cost[2:])
df_cost.columns = ["cost"]
new_col = [2,3,4]
df_cost.insert(0, 'cluster', new_col)

import pylab as pl
pl.figure(figsize=(8,8))
pl.plot(df_cost.cluster, df_cost.cost)
pl.xlabel('Number of Clusters')
pl.ylabel('Score')
pl.title('Elbow Curve')
pl.show()
```



- Les données sont multivariées - La normalisation n'est pas toujours nécessaire, mais il serait plus prudent de normaliser les données. En effet, laisser les variances inégales équivaut à donner plus de poids aux variables avec une variance plus faible. - on choisit k=2

0.3.3 3.3)

[45]:
```
from pyspark.ml.feature import PCA as PCAml

pca = PCAml(k=2, inputCol="Attributes_", outputCol="pca")
model = pca.fit(data_)
transformed = model.transform(data_)
transformed.show()
```

```
+--------------------+-----+--------------------+
|         Attributes_|Churn|                 pca|
```

```
+--------------------+-----+--------------------+
|[25.0,265.1,197.4…|    0|[567.649601009782…|
|[26.0,161.6,195.5…|    0|[573.104783261662…|
|[0.0,243.4,121.2,…|    0|[569.278777029948…|
|[0.0,299.4,61.9,1…|    0|[556.257748835515…|
|[0.0,166.7,148.3,…|    0|[574.086111539053…|
|[0.0,223.4,220.6,…|    0|[705.262624651160…|
|[24.0,218.2,348.5…|    0|[706.717087051588…|
|[0.0,157.0,103.1,…|    0|[572.689025345200…|
|[0.0,184.5,351.6,…|    0|[564.434016783563…|
|[37.0,258.6,222.0…|    0|[566.672376718725…|
|[0.0,129.1,228.5,…|    1|[576.686091783384…|
|[0.0,187.7,163.4,…|    0|[572.681621540551…|
|[0.0,128.8,104.9,…|    0|[565.444290228554…|
|[0.0,156.6,247.6,…|    0|[709.704442024259…|
|[0.0,120.7,307.2,…|    0|[578.365541365921…|
|[0.0,332.9,317.8,…|    1|[566.867519698282…|
|[27.0,196.4,280.9…|    0|[565.470492380628…|
|[0.0,190.7,218.2,…|    0|[708.533870816247…|
|[33.0,189.7,212.8…|    0|[708.466259947281…|
|[0.0,224.4,159.5,…|    0|[571.146327834011…|
+--------------------+-----+--------------------+
only showing top 20 rows
```