Data set conversion between Spark Dataframe & Pandas Dataframe

In [ ]:
```
import findspark
import pyspark
findspark.init()
```

In [2]:
```
from pyspark.sql import SparkSession
from pyspark import SparkContext, SparkConf

spark = SparkSession.builder.appName('abc').getOrCreate()
sc = spark.sparkContext
```

In [3]:
```
from sklearn.datasets import load_boston
import pandas as pd
import numpy as np

data = load_boston()

df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data['target']
df.head(4)
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94

From Pandas to Spark Dataframe

In [4]:
```
df = spark.createDataFrame(df)
```

In [5]:
```
df.show(5)
```

```
+-------+----+-----+----+-----+-----+----+------+---+-----+-------+------+-----
+------+
|   CRIM|  ZN|INDUS|CHAS|  NOX|   RM| AGE|   DIS|RAD|  TAX|PTRATIO|     B|LSTAT
|target|
+-------+----+-----+----+-----+-----+----+------+---+-----+-------+------+-----
+------+
|0.00632|18.0| 2.31| 0.0|0.538|6.575|65.2|  4.09|1.0|296.0|   15.3| 396.9| 4.98
|  24.0|
|0.02731| 0.0| 7.07| 0.0|0.469|6.421|78.9|4.9671|2.0|242.0|   17.8| 396.9| 9.14
|  21.6|
|0.02729| 0.0| 7.07| 0.0|0.469|7.185|61.1|4.9671|2.0|242.0|   17.8|392.83| 4.03
|  34.7|
|0.03237| 0.0| 2.18| 0.0|0.458|6.998|45.8|6.0622|3.0|222.0|   18.7|394.63| 2.94
|  33.4|
|0.06905| 0.0| 2.18| 0.0|0.458|7.147|54.2|6.0622|3.0|222.0|   18.7| 396.9| 5.33
|  36.2|
+-------+----+-----+----+-----+-----+----+------+---+-----+-------+------+-----
+------+
only showing top 5 rows
```

In [21]:
```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(
    inputCols=["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM","AGE","DIS","RAD","TAX'
    outputCol="features")

newdf=df.select('CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM','AGE','DIS','RAD','TA

outputdf = assembler.transform(newdf)
outputdf2=outputdf.withColumnRenamed('target','label').select('features','label'
outputdf2.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[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|
|[0.06905,0.0,2.18...| 36.2|
+--------------------+-----+
only showing top 5 rows
```

In [22]:
```
train, test = outputdf2.randomSplit([0.9, 0.1], seed=12345)
```

In [23]:
```
train.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.00632,18.0,2.3...| 24.0|
|[0.01311,90.0,1.2...| 35.4|
|[0.0136,75.0,4.0,...| 18.9|
|[0.01432,100.0,1....| 31.6|
|[0.02055,85.0,0.7...| 24.7|
+--------------------+-----+
only showing top 5 rows
```

In [24]:
```
test.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.01951,17.5,1.3...| 33.0|
|[0.02763,75.0,2.9...| 30.8|
|[0.09744,0.0,5.96...| 20.0|
|[0.12744,0.0,6.91...| 26.6|
|[0.13262,0.0,8.56...| 19.5|
+--------------------+-----+
only showing top 5 rows
```

In [31]:
```
from pyspark.ml.regression import LinearRegression

lr = LinearRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)

# Fit the model
lrModel = lr.fit(train)

#Print the coefficients and intercept for linear regression
print("Coefficients: %s" % str(lrModel.coefficients))
print("Intercept: %s" % str(lrModel.intercept))
```

```
Coefficients: [-0.033317902942330495,0.011055812732540832,-0.000710694032301676
2,2.829889672594076,-7.913708773315601,3.5132249990706783,0.0,-0.65798993441328
11,0.0,0.0,-0.8461695942769107,-0.5774501662132965]
Intercept: 30.05586810287888
```

In [40]:
```
pred=lrModel.transform(test).select("features", "label", "prediction")
```

In [60]:
```
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator()
evaluator.setPredictionCol("prediction")
print("evalue=",evaluator.evaluate(pred))

trainingSummary = lrModel.summary
print("numIterations: %d" % trainingSummary.totalIterations)
print("objectiveHistory: %s" % str(trainingSummary.objectiveHistory))
print("r2: %f" % trainingSummary.r2)
```

```
evalue= 4.988020181561825
numIterations: 11
objectiveHistory: [0.5, 0.43493758197977145, 0.24582010755406988, 0.22415052630
939936, 0.19655797185117405, 0.1927276209409042, 0.19152113350873332, 0.1906875
6638540232, 0.18989579900151002, 0.18932994193018401, 0.18924907583456094]
r2: 0.694956
```

In [ ]: