spark-example

February 7, 2025

[1]:
```
#example-1
from pyspark.sql import SparkSession

# Initialize SparkSession with the master URL
spark = SparkSession.builder.appName("example-1").master("spark://spark-master:
 ↪7077").getOrCreate()

# Example: Create a DataFrame
data = [("Alice", 34), ("Bob", 45), ("Cathy", 29)]
columns = ["Name", "Age"]
df = spark.createDataFrame(data, columns)

# Show the DataFrame
df.show()

# Stop the SparkSession
spark.stop()
```

```
25/02/05 02:42:13 WARN SparkConf: The configuration key 'spark.executor.port'
has been deprecated as of Spark 2.0.0 and may be removed in the future. Not used
anymore
25/02/05 02:42:13 WARN SparkConf: The configuration key 'spark.executor.port'
has been deprecated as of Spark 2.0.0 and may be removed in the future. Not used
anymore
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
25/02/05 02:42:14 WARN SparkConf: The configuration key 'spark.executor.port'
has been deprecated as of Spark 2.0.0 and may be removed in the future. Not used
anymore
25/02/05 02:42:14 WARN SparkConf: The configuration key 'spark.executor.port'
has been deprecated as of Spark 2.0.0 and may be removed in the future. Not used
anymore
25/02/05 02:42:14 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

```
+-----+---+
| Name|Age|
+-----+---+
```

```
|Alice| 34|
|  Bob| 45|
|Cathy| 29|
+-----+---+
```

[2]:
```
#example-2 (You need to upload the information.csv)
from pyspark.sql import SparkSession
import logging
import warnings

# Suppress PySpark and Py4J warnings
logging.getLogger("py4j").setLevel(logging.ERROR)
logging.getLogger("pyspark").setLevel(logging.ERROR)
logging.getLogger("sparkConf").setLevel(logging.ERROR)

# Suppress Python warnings
warnings.filterwarnings("ignore")

# Initialize SparkSession with the master URL
spark = SparkSession.builder.appName("example-2").master("spark://spark-master:
 ↪7077").getOrCreate()
# Set Spark log level to ERROR
spark.sparkContext.setLogLevel("ERROR")
# Read a CSV file into a DataFrame
file_path = "/spark/user/information.csv"
df = spark.read.csv(file_path, header=True)

# Show the DataFrame
df.show()

# Print the schema of the DataFrame
df.printSchema()

# Stop the SparkSession
spark.stop()
```

```
25/02/05 02:42:49 WARN SparkConf: The configuration key 'spark.executor.port'
has been deprecated as of Spark 2.0.0 and may be removed in the future. Not used
anymore

+-------+---+-----------+
|   Name|Age|       City|
+-------+---+-----------+
|  Alice| 30|   New York|
|    Bob| 25|Los Angeles|
|Charlie| 35|    Chicago|
+-------+---+-----------+
```

```
root
 |-- Name: string (nullable = true)
 |-- Age: string (nullable = true)
 |-- City: string (nullable = true)
```

[3]:
```
#example-3 Run spark locally
from pyspark.sql import SparkSession

# Step 1: Initialize SparkSession
spark = SparkSession.builder.appName("example-3").master("local[*]").
 ↪getOrCreate()
# Step 2: Create a synthetic dataset
data = [
    (1.0, 2.0),
    (2.0, 4.0),
    (3.0, 6.0),
    (4.0, 8.0),
    (5.0, 10.0)
]
columns = ["feature", "label"]
df = spark.createDataFrame(data, columns)
df.show()
```

```
[Stage 0:>                                                  (0 + 1) / 1]
```

```
+-------+-----+
|feature|label|
+-------+-----+
|    1.0|  2.0|
|    2.0|  4.0|
|    3.0|  6.0|
|    4.0|  8.0|
|    5.0| 10.0|
+-------+-----+
```

[23]:
```
#Example-4
#Build one ML model by using array data and using it the Spark cluster
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import RegressionEvaluator

# Step 1: Initialize SparkSession
```

```
spark = SparkSession.builder.appName("LinearRegressionExample").master("spark://
 ↪spark-master:7077").getOrCreate()

# Step 2: Create a synthetic dataset
data = [
    (1.0, 2.0),
    (2.0, 4.0),
    (3.0, 6.0),
    (4.0, 8.0),
    (5.0, 10.0)
]
columns = ["feature", "label"]
df = spark.createDataFrame(data, columns)

# Step 3: Prepare the data for Linear Regression
# Combine features into a single vector column
assembler = VectorAssembler(inputCols=["feature"], outputCol="features")
df = assembler.transform(df)

# Step 4: Split the data into training and testing sets
train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)

# Step 5: Create and train the Linear Regression model
lr = LinearRegression(featuresCol="features", labelCol="label")
lr_model = lr.fit(train_data)

# Step 6: Make predictions on the test data
predictions = lr_model.transform(test_data)

# Step 7: Evaluate the model
evaluator = RegressionEvaluator(labelCol="label", predictionCol="prediction",␣
 ↪metricName="rmse")
rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Step 8: Show the predictions
predictions.select("features", "label", "prediction").show()

# Step 9: Print model coefficients and intercept
print(f"Coefficients: {lr_model.coefficients}")
print(f"Intercept: {lr_model.intercept}")

# Step 10: Stop the SparkSession
spark.stop()
```

```
Root Mean Squared Error (RMSE): 0.0
```

```
+--------+-----+----------+
|features|label|prediction|
+--------+-----+----------+
|   [3.0]|  6.0|       6.0|
+--------+-----+----------+

Coefficients: [2.0]
Intercept: 0.0
```

[ ]: