Evaluation of Decision Tree in Spark

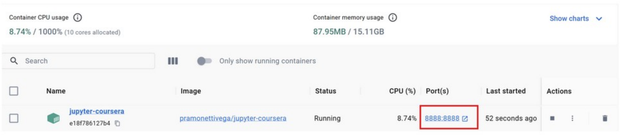
By the end of this activity, you will be able to perform the following in Spark:

1. Determine the accuracy of a classifier model
2. Display the confusion matrix for a classifier model

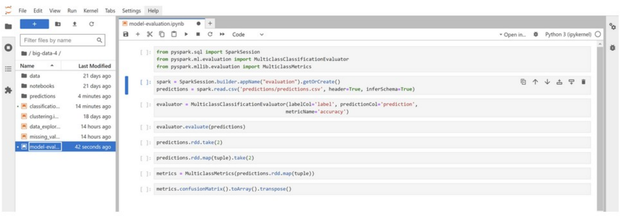
For this activity, you should have completed the creation of the JupyterLab container. If not follow, Steps 1-3 on the previous activity *Hand On: Data Exploration in Spark*, and then come back to Step 2 of this activity.

Step 1. Start the container. Open Docker Desktop and start your *jupyter-coursera* container.

When Jupyter starts running, click on the port to access JupyterLab in your browser:



Step 2. Open your notebook. Once you're in JupyterLab, go to the *big-data-4* folder and open the classification.*ipynb* notebook.



Step 3. Load predictions. Execute the first cell to load the classes used in this activity:

```
[1]: from pyspark.sql import SparkSession
     from pyspark.ml.evaluation import MulticlassClassificationEvaluator
     from pyspark.mllib.evaluation import MulticlassMetrics
```

Execute the next cell to load the predictions CSV file that we created at the end of the Week 3 Hands-On *Classification in Spark* into a DataFrame:

```
[2]: spark = SparkSession.builder.appName("evaluation").getOrCreate()
     predictions = spark.read.csv('predictions/predictions.csv', header=True, inferSchema=True)
```

Step 4. Compute accuracy. Let's create an instance of *MulticlassClassificationEvaluator* to determine the accuracy of the predictions:

```
[3]: evaluator = MulticlassClassificationEvaluator(labelCol='label', predictionCol='prediction',
                                                   metricName='accuracy')
```

The first two arguments specify the names of the label and prediction columns, and the third argument specifies that we want the overall precision.

We can compute the accuracy by calling *evaluate()*:

```
[4]: evaluator.evaluate(predictions)

[4]: 0.7844036697247706
```

Step 5. Display confusion matrix. The *MulticlassMetrics* class can be used to generate a confusion matrix of our classifier model. However, unlike *MulticlassClassificationEvaluator*, *MulticlassMetrics* works with RDDs of numbers and not DataFrames, so we need to convert our *predictions* DataFrame into an RDD.

If we use the *rdd* attribute of *predictions*, we see this is an *RDD* of *Rows:*

```
[5]: predictions.rdd.take(2)

[5]: [Row(prediction=1.0, label=1.0), Row(prediction=1.0, label=1.0)]
```

Instead, we can map the RDD to *tuple* to get an RDD of numbers:

```
[6]: predictions.rdd.map(tuple).take(2)

[6]: [(1.0, 1.0), (1.0, 1.0)]
```

Let's create an instance of *MulticlassMetrics* with this RDD:

```
[7]: metrics = MulticlassMetrics(predictions.rdd.map(tuple))
```

NOTE: Ignore the warning. The reason for the warning is that the *MulticlassMetrics* class uses the *SQLContext* class in its design, and that class is deprecated.

The *confusionMatrix()* function returns a Spark *Matrix*, which we can convert to a Python Numpy array, and transpose to view:

```
[8]: metrics.confusionMatrix().toArray().transpose()

[8]: array([[87., 28.],
            [19., 84.]])
```

Step 6. Exiting the container. To exit JupyterLab, simply close the tab in your browser. To stop the container, go to Docker Desktop and click on the *stop* button. We recommend not to delete the container, as this container will be used for multiple activities across this specialization.