Exercises with Spark ML

This lab is split into two part. The first part is very guided and the goal to make you write a program capable of estimating whether a tumor is malign or benign according to a few features collected from a biopsy! The second part is much more exploratory with several ML tasks on several datasets. One of the goal of this lab is to make you efficient at reading and using the documentation.

The main page to look for the documentation of spark ML is `https://spark.apache.org/docs/latest/ml-guide.html`.

The winconsin breast cancer dataset

The Winconsin breast cancer dataset contains 699 cases of breast cancers. The dataset is presented here : `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)` and you can directly download the dataset at the following URL : `https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data`.

Our goal here is to determine whether a tumor is *malign* or *benign* using the provided features. Each line of the dataset represents a case and contains 11 numerical values separated by commas. Here is a description of the 11 columns :

#	**Attribute**	**Domain**
1	Sample code number	id number
2	Clump Thickness	1 - 10
3	Uniformity of Cell Size	1 - 10
4	Uniformity of Cell Shape	1 - 10
5	Marginal Adhesion	1 - 10
6	Single Epithelial Cell Size	1 - 10
7	Bare Nuclei	1 - 10
8	Bland Chromatin	1 - 10
9	Normal Nucleoli	1 - 10
10	Mitoses	1 - 10
11	Class :	(2 for benign, 4 for malignant)

1 Load the data into Spark

The first step for our ML application consist in setup up Spark and reading the dataset. To do this you can use the following code snippet :

```
from pyspark.sql.types import StructType, StructField
from pyspark.sql.types import DoubleType, IntegerType
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SQLContext
from pyspark.ml.linalg import VectorUDT,Vectors

def toFloat(x):
    if x == '?':
        return 5.0
    else:
        return float(x)

def doLine(l):
    item=l.split(",")
    label = 1
    if item[10]=='2':
        label=0
    return (Vectors.dense([toFloat(e) for e in item[1:10]]),label)

raw_data = sc.textFile(path+"/breast-cancer-wisconsin.data")
schema = StructType([StructField("features", VectorUDT(), True),
                    StructField("label",IntegerType(),True)])
data = SQLContext(sc).createDataFrame(raw_data.map(doLine),schema)
```

This code is also provided here `https://p.jachiet.com/fgtC4t89` for easier copy/paste.

DO: *Explain what the provided code does.*
DO: *What does `data` looks like ?*
DO: *What is the schema of the `data` ?*
DO: *In our dataset, how many tumors are benign ? malign ?*

2 Splitting into training and testing

To build our model we first need to split our data into a *training* set and a *testing* set. Here we will split according to the usual 1-9 rule, which means that 90% of the dataset will be used for training while 10% will be used to test our model. For this you can use the function `randomSplit` (see documentation).
DO: *Split `data` into two DataFrames `test` and `train`.*

3 Building the model

Building the model is done using the object DecisionTree. `DecisionTreeClassifier` in the `from pyspark.ml.c`
package.
DO: *Include the relevant packages with the following code snippet :*

```
from pyspark.ml.classification import DecisionTreeClassifier
```

DO: *Read the documentation of `DecisionTreeClassifier`.*
DO: *Train a model called `bc_model` with your training data !*

4 Testing your model

Computing predictions for the test data can be done by applying the model on the test data :

```
predictions = bc_model.transform(test)
evaluator = BinaryClassificationEvaluator(rawPredictionCol="prediction",
                                          labelCol="label",
                                          metricName='areaUnderROC')
result = evaluator.evaluate(predictions)
```

DO: *What does the DataFrame* `prediction` *contain ?*
DO: *What is the area under ROC of our classifier ?*
DO: *What is the accuracy of our classifier ?*

5 Improving the model

The model is based on a set of parameters (number of bins, depth, etc.). Spark ML has tools to help you decide which parameters are well suited for your application (see `https://spark.apache.org/docs/2.2.0/ml-tuning.html`)
DO: *Use train-validation to determine a good set of parameters.*
DO: *Use cross-validation to determine a good set of parameters.*
DO: *What is the area under ROC of your model now ?*

6 When you are done (OPTIONAL)

The following tasks will help you see a (relatively) large spectrum of tasks that can be done with Spark ML.

6.1 Improve the classification

Consider different models for the Breast Cancer task (e.g. Logistic Regression, Random Forest model and Gradient-boosted tree classifier).

6.2 Another classification task (IRIS dataset)

Use the knowledge developed in this lab to classify plants. Download the dataset `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data` presented here `https://archive.ics.uci.edu/ml/datasets/Iris` and make a classifier.

6.3 Visualization task (IRIS dataset)

Use the PCA algorithm and matplotlib to help you visualize the Iris dataset (see section above). Can you guess the groups ?

6.4 Clustering task (IRIS dataset)

Remove the “class” information from the IRIS dataset and try to cluster then compare the result of your clustering with the label information.

6.5 Regression task (WINE QUALITY dataset)

Estimate Wine Quality based on chemical components. See `https://archive.ics.uci.edu/ml/datasets/Wine+Quality` for the dataset. The two datasets can be found here : `https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/` (what happens if you train a regression on the white wine dataset and use it for the red wine dataset ?).