PySpark Training

The purpose of this document is that you get familiar with the environment and tools required for the lab session. This is an optional task, thus no delivery is expected and no grade will be assigned.

Exercise 1. Analyzing banking data with Spark

The first exercise consists on exploring a dataset with direct marketing campaigns (phone calls) of a Portuguese banking institution[1]. The file, which is a is a comma-separated (CSV) file, is already provided in the project's resources folder with the name bank.csv. You can check its description, together with the attributes it contains and their types, in the file **bank-names.txt**.

We are interested in analyzing whether the type of job and having or not a personal loan affects the account balance. Thus, we precisely ask you to implement in class Exercise 1.java the computation of average balance of personal loan holders and non-personal loan holders per job type, ordered by job type name. You should obtain the following output:

"admin.": avg balance with loan = 861.0769230769231, without loan = 1312.71834625323
"blue-collar": avg balance with loan = 799.2115384615385, without loan = 1141.6278481012657
"entrepreneur": avg balance with loan = 1143.2682926829268, without loan = 1807.1417322834645
"housemaid": avg balance with loan = 2144.076923076923, without loan = 2075.8888888888887
"retired": avg balance with loan = 1171.6875, without loan = 2504.6464646464647
"self-employed": avg balance with loan = 792.1, without loan = 1510.1176470588234
"services": avg balance with loan = 628.027027027027, without loan = 1206.6355685131196
"student": avg balance with loan = 1161.0, without loan = 1548.433734939759
"technician": avg balance with loan = 799.8403361344538, without loan = 1428.3882896764253
"unemployed": avg balance with loan = 537.6153846153846, without loan = 1151.8
"unknown": avg balance with loan = 341.0, without loan = 1533.081081081081

Exercise 2. Predicting risk of heart diseases using kNN

In this exercise, we are going to classify heart diseases based on patient information. The algorithm we will use is one nearest neighbor (1NN from now on). Note that, in this exercise we are not going to use any external ML library but to implement a very naive version of the algorithm.

As many predictive algorithms, 1NN assumes a relatively large dataset that contains rows used as baseline for the predictions, and other different rows to predict the class (in this context the classes are "1", indicating the presence of a heart disease, or "0" indicating the absence of a heart disease). More specifically, the former is typically called the training set and the latter is the test set. The idea behind 1NN is to compute, for every row in the test set, its distance (Euclidean; see formula 1, where x and y are two different rows and d is the number of columns) with respect to all rows from the training set. The final predicted class corresponds to the closest row class from the training set.

1 You can learn more about this dataset in https://archive.ics.uci.edu/ml/datasets/Bank+Marketing.

$$dist(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_d - y_d)^2} \quad (1)$$

- The dataset you are given is a CSV file (heart.csv) with multiple columns composed as follows:id, representing the patient identifier;
- 13 variables that must be used to compute distances (d = 13);
- diagnosis, with values 1 (presence of heart disease) or 0 (absence of heart disease);
- dataset, with values TRAIN or TEST.

Next, we provide you with the high-level guidelines on how to implement the algorithm, for the details refer to the previously mentioned session.

1. Split the train and test sets
2. Generate all possible combinations of instances in the training and test sets
3. For each combination of training and test instances:
   - Calculate the euclidean distance of their predictor variables
4. For each test instance:
   - Use as prediction that from the training set that has the minimum euclidean distance.
5. Generate the confusion matrix

Your task consists on implementing the complete dataflow for 1NN and generation of the confusion matrix in a single Spark program. The expected output is the confusion matrix as below:

0_0 16
0_1 15
1_0 10
1_1 21

Exercise 3. Querying AIMS

In this exercise, we will obtain data from the AIMS database. Precisely, using Spark, we aim to count the number of airports that have some flight departuring with departure. You should query the “flights” table from AIMS. The expected output is the following string:

1116 airports with at least one departure

Exercise 4. Creation of a decision tree

This last exercise concerns the creation and evaluation of a decision tree using Spark MLlib. Note that this course focus on the data management aspect of Big Data processing pipelines, thus here we will use the library as provided in Spark’s tutorial. To this end, the goal of this exercise is to run the tutorial on decision trees (given that it will also be used in the project):

- https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#decision-tree-classifier

Assume the input data is a matrix provided in libsvm format from:

- https://github.com/apache/spark/blob/master/data/mllib/sample_libsvm_data.txt