Manual d'utilisation

Juliette CHOUDJAYE
François MATVIENKO
Armel MOTH
Lydia VIJAYARAJAH

14 mars 2018

Résumé

Pour pouvoir faire du Back2Back testing avec notre programme, on commence par se placer dans la classe Lanceur du package tools.

1 Back2Back Testing

1.1 Les Prérequis

- 1 - Le fichier CSV : Définir le chemin du fichier à traiter.
- 2 - Echantillon d'apprentissage et de test : créer une instance de SplitCSV qui va chercher le fichier au chemin renseigné et qui crée les deux jeux de données test et train.
- 3 - Choix des librairies et méthodes : On instancie les librairie que l'on veut comparer en précisant la méthode.
- 4 - Paramètres : Il est de plus possible d'ajouter des arguments en créant un HashMap ayant pour clé le nom du paramètre.

```
    // Exemple 1 Utilisation standard des 3 librairies pour le
    //jeu de données iris et la méthode RandomForest

1-> String pathDataIris = "src/main/resources/iris.csv";
2-> SplitCSV dataSplit = new SplitCSV(pathDataIris, "exemple1");

    // SplitCSV permet de séparer le jeu de donnée en une partie
    //d'apprentissage et une partie de test

    // Création des trois librairies

3-> Library sparkMLEx1 = new SparkMLLib(dataSplit,Methode.RANDOMFOREST);

4-> HashMap argumentsRenjin = new HashMap<String,String>();
    argumentsRenjin.put("indY", "5");
3-> Library rEnginEx1 = new RenjinLib(dataSplit,Methode.RANDOMFOREST,argumentsRenjin);

3-> Library wekaEx1 = new WekaLib(dataSplit,Methode.RANDOMFOREST);
```

FIGURE 1 – Exemple 1

1.2 Comparaison

— 5 - Accuracy : Il est possible d'obtenir l'accuracy d'une librairie avec la méthode spécifiée grâce à la fonction getAccuracy() de la classe abstraite Library

— 6 - Comparaison : Pour comparer les performances de deux librairies, créer une instance de la classe Comparateur ayant comme paramètres deux instances de la classe Library.

```
1-> String pathDataEx2 = "src/main/resources/statsFSEVarFry.csv";
2-> SplitCSV dataSplitEx2 = new SplitCSV(pathDataEx2, "exemple2");

4-> HashMap argumentsRenjin2 = new HashMap<String,String>();
    argumentsRenjin2.put("indY", "19");

3-> Library sparkMLEx2 = new SparkMLLib(dataSplitEx2,Methode.RANDOMFOREST);
    Library RenjinEx2 = new RenjinLib(dataSplitEx2,Methode.RANDOMFOREST,argumentsRenjin2);

6-> Comparateur comparateur = new Comparateur(sparkMLEx2,RenjinEx2);
    System.out.println(comparateur.getResult());
```

FIGURE 2 – Exemple 2

1.3 Quelques exemples

Vous trouverez dans la classe Lanceur des exemples qui utilisent le second jeu de données.

Exemple 3 : Comparer deux méthodes de la même librairie.

```
String pathDataEx3 = "src/main/resources/statsFSEVarFry.csv";
SplitCSV dataSplitEx3 = new SplitCSV(pathDataEx3, "exemple3");
HashMap argumentsLib1 = new HashMap<String,String>();
argumentsLib1.put("maxDepth", "1");
argumentsLib1.put("numTrees", "10");

HashMap argumentsLib2 = new HashMap<String,String>();
argumentsLib2.put("maxDepth", "10");


Library sparkMLEx3 = new SparkMLLib(dataSplitEx3,Methode.RANDOMFOREST,argumentsLib1);
Library sparkML2Ex3 = new SparkMLLib(dataSplitEx3,Methode.DECISIONTREE,argumentsLib2);

Comparateur comparateurEx3 = new Comparateur(sparkMLEx3,sparkML2Ex3);
System.out.println(comparateurEx3.getResult());
```

FIGURE 3 – Exemple 3

Exemple 4 : Comparer la même méthode d'une librairie mais avec des paramètres différents.

```
String pathDataEx4 = "src/main/resources/statsFSEVarFry.csv";
SplitCSV dataSplitEx4 = new SplitCSV(pathDataEx4, "exemple4");
HashMap argumentsLib1Ex4 = new HashMap<String,String>();
argumentsLib1Ex4.put("numTrees", "20");

HashMap argumentsLib2Ex4 = new HashMap<String,String>();
argumentsLib2Ex4.put("numTrees", "5");


Library sparkMLEx4 = new SparkMLLib(dataSplitEx4,Methode.RANDOMFOREST,argumentsLib1Ex4);
Library sparkML2Ex4 = new SparkMLLib(dataSplitEx4,Methode.RANDOMFOREST,argumentsLib2Ex4);

Comparateur comparateurEx4 = new Comparateur(sparkMLEx4,sparkML2Ex4);
System.out.println(comparateurEx4.getResult());
```

FIGURE 4 – Exemple 4