eBook Gratuit

APPRENEZ weka

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#weka

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: weka

It is an unofficial and free weka ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official weka.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec weka

Remarques

Cette section fournit une vue d'ensemble de ce que weka est et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de weka, et établir un lien avec les sujets connexes. La documentation de weka étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Installation ou configuration

Weka est une collection d'algorithmes d'apprentissage automatique pour les tâches d'exploration de données. Les algorithmes peuvent être appliqués directement à un jeu de données ou appelés depuis votre propre code Java. Weka contient des outils pour le prétraitement des données, la classification, la régression, le regroupement, les règles d'association et la visualisation. Il est également bien adapté au développement de nouveaux systèmes d'apprentissage automatique.

Téléchargement et installation de Weka

Il existe deux versions de Weka: Weka 3.8 est la dernière version stable et Weka 3.9 la version de développement. À la pointe de la technologie, il est également possible de télécharger des instantanés nocturnes.

Les versions stables ne reçoivent que des corrections de bogues, tandis que la version de développement reçoit de nouvelles fonctionnalités. Weka 3.8 et 3.9 disposent d'un système de gestion de paquets qui permet à la communauté Weka d'ajouter de nouvelles fonctionnalités à Weka. Le système de gestion des paquets nécessite une connexion Internet pour télécharger et installer les paquets.

Vous pouvez télécharger l'application pour Windows / MacOS / Linux ici .

Intégrez la bibliothèque WEKA dans votre code:

pox.xml:

```
<dependency>
    <groupId>nz.ac.waikato.cms.weka</groupId>
    <artifactId>weka-dev</artifactId>
    <version>3.9.1</version>
</dependency>
```

gradle:

```
compile group: 'nz.ac.waikato.cms.weka', name: 'weka-dev', version: '3.9.1'
```

Lire Commencer avec weka en ligne: https://riptutorial.com/fr/weka/topic/3699/commencer-avec-weka

Chapitre 2: Classification de texte

Examples

Classification de texte avec LibLinear

- Créer des instances d'entraînement à partir du fichier .arff

```
private static Instances getDataFromFile(String path) throws Exception{

    DataSource source = new DataSource(path);
    Instances data = source.getDataSet();

    if (data.classIndex() == -1){
        data.setClassIndex(data.numAttributes()-1);
        //last attribute as class index
    }

    return data;
}
```

```
Instances trainingData = getDataFromFile(pathToArffFile);
```

- Utilisez **StringToWordVector** pour transformer vos attributs de chaîne en représentation numérique:

  * Caractéristiques importantes de ce filtre:

    1. représentation tf-idf
    2. enracinement
    3. les lettres minuscules
    4. mots vides
    5. représentation n-gramme *

```
    StringToWordVector() filter = new StringToWordVector();
    filter.setWordsToKeep(1000000);
    if(useIdf){
        filter.setIDFTransform(true);
    }
    filter.setTFTransform(true);
    filter.setLowerCaseTokens(true);
    filter.setOutputWordCounts(true);
    filter.setMinTermFreq(minTermFreq);
    filter.setNormalizeDocLength(new
SelectedTag(StringToWordVector.FILTER_NORMALIZE_ALL,StringToWordVector.TAGS_FILTER));
    NGramTokenizer t = new NGramTokenizer();
    t.setNGramMaxSize(maxGrams);
    t.setNGramMinSize(minGrams);
    filter.setTokenizer(t);
    WordsFromFile stopwords = new WordsFromFile();
    stopwords.setStopwords(new File("data/stopwords/stopwords.txt"));
```

```
    filter.setStopwordsHandler(stopwords);
    if (useStemmer){
        Stemmer s = new /*Iterated*/LovinsStemmer();
        filter.setStemmer(s);
    }
    filter.setInputFormat(trainingData);
```

- Appliquez le filtre à trainingData: `trainingData = Filter.useFilter(trainingData, filter);`
- Créer le classificateur LibLinear

    1. SVMType 0 ci-dessous correspond à la régression logistique régularisée L2
    2. Définissez `setProbabilityEstimates(true)` pour imprimer les probabilités de sortie

```
        Classifier cls = null;
        LibLINEAR liblinear = new LibLINEAR();
        liblinear.setSVMType(new SelectedTag(0, LibLINEAR.TAGS_SVMTYPE));
        liblinear.setProbabilityEstimates(true);
        // liblinear.setBias(1); // default value
        cls = liblinear;
        cls.buildClassifier(trainingData);
```

- Enregistrer le modèle

```
    System.out.println("Saving the model...");
    ObjectOutputStream oos;
    oos = new ObjectOutputStream(new FileOutputStream(path+"mymodel.model"));
    oos.writeObject(cls);
    oos.flush();
    oos.close();
```

- Créer des instances de test à partir `.arff` fichier `.arff`

```
 Instances trainingData = getDataFromFile(pathToArffFile);
```

- Charger le classificateur

```
Classifier myCls = (Classifier) weka.core.SerializationHelper.read(path+"mymodel.model");
```

- **Utilisez le même filtre StringToWordVector que ci-dessus ou créez-en un nouveau pour testingData, mais n'oubliez pas d'utiliser le trainingData pour cette commande:** **`filter.setInputFormat(trainingData);`** *Cela rendra les instances de formation et de test compatibles. Sinon, vous pouvez utiliser* *`InputMappedClassifier`*
- Appliquez le filtre à testingData: `testingData = Filter.useFilter(testingData, filter);`
- Classer!

    1.Obtenez la valeur de la classe pour chaque instance du jeu de tests

```
 for (int j = 0; j < testingData.numInstances(); j++) {
    double res = myCls.classifyInstance(testingData.get(j));
```

```
}
```

`res` est une valeur double qui correspond à la classe nominale définie dans le fichier `.arff`. Pour obtenir la classe nominale, utilisez: `testintData.classAttribute().value((int)res)`

2.Obtenir la distribution de probabilité pour chaque instance

```
for (int j = 0; j < testingData.numInstances(); j++) {
   double[] dist = first.distributionForInstance(testInstances.get(j));
}
```

`dist` est un double tableau contenant les probabilités pour chaque classe définie dans le fichier `.arff`

Remarque. Le classificateur doit prendre en charge les distributions de probabilités et les activer avec: `myClassifier.setProbabilityEstimates(true);`

Lire Classification de texte en ligne: https://riptutorial.com/fr/weka/topic/7753/classification-de-texte

Chapitre 3: Comment utiliser CPython Scripting dans Weka?

Remarques

Comment installer CPython dans Weka?

Installez wekaPython

1. aller aux `tools` , ouvrir le `package manager`
2. recherche `wekaPython` , sélectionnez et cliquez pour installer

Installer les bibliothèques Python

1. installer anaconda ou conda
2. installer quatre paquets: numpy, pandas, matplotlib, scikit-learn
3. pour une documentation complète de l'installation, voir conda

Examples

Exemple de Hello World pour CPython de Weka

> Allez dans `Explorer` , `iris.arff` données `iris.arff` , puis allez dans `CPython Scripting` , copiez et collez les lignes de codes suivantes dans les `Python Scripts` :

```
hi = "Hello, CPython of Weka!"
hello = hi.upper()
iris = py_data
info = iris.describe()
```

> Pour voir la sortie, accédez à `Python Variables` , sélectionnez `hi` , par exemple, puis cliquez sur `Get text`

Lire Comment utiliser CPython Scripting dans Weka? en ligne:
https://riptutorial.com/fr/weka/topic/7921/comment-utiliser-cpython-scripting-dans-weka-

Chapitre 4: Comment utiliser R dans Weka

Remarques

Pourquoi utiliser R dans Weka?

1. R est un outil puissant pour le prétraitement des données
2. R possède un grand nombre de bibliothèques et continue de croître
3. R dans Weka, peut facilement obtenir des données, les traiter et passer à Weka de manière transparente

---

Comment configurer R dans Weka

Pour l'utilisateur Mac

1. remplacer l'ancienne info.Plist par la nouvelle donnée par Mark Hall
2. télécharger et installer R
3. installer `rJava` dans R avec

   install.packages ('rJava')
4. installer `Rplugin` avec `Weka Package Manager`
5. aller au dossier `weka 3-8-0` (si c'est la version que vous utilisez), et ouvrir son terminal, et
6. exécuter les 2 lignes de codes suivantes (merci à Michael Hall)

   export R_HOME = / Library / Frameworks / R.framework / Resources
   java -Xss10M -Xmx4096M -cp.: weka.jar weka.gui.GUIChooser
7. pour vous simplifier la vie, dans un répertoire où vous voulez travailler avec weka, enregistrez le code ci-dessus dans un fichier nommé `weka_r.sh`
8. rendez-le exécutable, à l'intérieur du terminal de ce répertoire, exécutez le code ci-dessous:

   chmod a + x weka_r.sh
9. collez `weka.jar` de weka 3-8-0 dans le répertoire et exécutez le code ci-dessous:

   ./weka_r.sh

Maintenant, vous êtes prêt à partir. La prochaine fois, il vous suffit d'aller au terminal du répertoire et d'exécuter `./weka_r.sh` pour lancer R avec Weka.

Comment recevoir des données de Weka?

ouvrez Weka du terminal :
allez dans le répertoire de `Weka 3-8-0` , ouvrez son terminal, exécutez le code suivant:

```
java -jar weka.jar
```

données via Weka Explorer :

1. panneau de `preprocess` , cliquez sur `open file` , choisissez un fichier de `weka data folder` ;
2. allez dans le panneau de la `R console` , tapez R scripts dans la `R console box`

données via Weka KnowledgeFlow :

1. Panneau des `Data mining processes` , cliquez sur `DataSources` pour choisir `ArffLoader` par exemple, cliquez dessus sur le canevas;
2. double-cliquez sur `ArffLoader` pour charger un fichier de données
3. Panneau de `Scripting` , cliquez sur `RscriptExecutor` sur le canevas
4. `option` + cliquez sur `ArffLoader` , sélectionnez le `dataset` , puis cliquez sur `RScript Executor` pour les lier
5. double-cliquez `RScript Executor` pour taper le script R, ou
6. cliquez sur `Settings` et sélectionnez `R Scripting` pour utiliser la console R avec les données de weka

---

Jouer des codes R

1. charger `iris.arff` avec Explorer ou KnowledgeFlow;
2. Essayez de `Plotting inside R Console` exemple de la `Plotting inside R Console` ci-dessus

Examples

Tracer à l'intérieur de la console R

> Les codes suivants peuvent être trouvés à partir du cours de Weka

Étant donné que `iris.arff` est chargé dans weka, dans la `R console` Weka Explorer ou dans `R Scripting` Weka KnowledgeFlow, vous pouvez jouer avec les codes suivants pour créer de beaux tracés:

```
library(ggplot2)

ggplot(rdata, aes(x = petallength)) + geom_density()

ggplot(rdata, aes(x = petallength)) + geom_density() + xlim(0,8)
```

```
ggplot(rdata, aes(x = petallength)) + geom_density(adjust = 0.5) + xlim(0,8)

ggplot(rdata, aes(x = petallength, color = class)) + geom_density(adjust = 0.5) + xlim(0,8)

ggplot(rdata, aes(x = petallength, color = class, fill = class)) + geom_density(adjust = 0.5)
+ xlim(0,8)

ggplot(rdata, aes(x = petallength, color = class, fill = class)) + geom_density(adjust = 0.5,
alpha = 0.5) + xlim(0,8)




library(reshape2)
ndata = melt(rdata)
ndata

ggplot(ndata, aes(x = value, color = class, fill = class)) + geom_density(adjust = 0.5, alpha
= 0.5) + xlim(0,8) + facet_grid(variable ~ .)

ggplot(ndata, aes(x = value, color = class, fill = class)) + geom_density(adjust = 0.5, alpha
= 0.5) + xlim(0,8) + facet_grid(. ~ variable)

ggplot(ndata, aes(y = value, x = class, colour = class)) + geom_boxplot() + facet_grid(. ~
variable)
```

Lire Comment utiliser R dans Weka en ligne: https://riptutorial.com/fr/weka/topic/7916/comment-utiliser-r-dans-weka

Chapitre 5: Comparaison simple des interfaces de Weka

Introduction

Weka a de nombreuses interfaces, Explorer, KnowledgeFlow, Experimenter, SimpleCLI, Workbench. Tous partagent la plupart du temps, peuvent effectuer les mêmes tâches, avec un accent et une flexibilité différents. Ici, nous allons explorer leurs différents axes et flexibilités.

Remarques

Explorateur

pro:

1. faire tout rapidement
2. donner une vue rapide et complète de la structure des données

cos: ne peut pas enregistrer le processus;

Expérimentateur

pro:

1. comparez plusieurs modèles à la fois, par exemple, exécutez 3 classificateurs différents contre 5 ensembles de données tous ensemble, et voyez le résultat comparé à un endroit;
2. l'expérience peut être sauvée

KnowledgeFlow

pro:

1. faire presque tout ce que l'explorateur peut faire
2. peut sauver le processus

cos:

1. KF ne peut pas faire le travail de l'expérimentateur, car il ne prend pas en charge les boucles, mais ADAMS peut vous aider.
2. KF ne peut pas accéder aux fonctionnalités de bas niveau de l'API Weka.

simpleCLI

pro: exécuter des tâches similaires à celles de l'explorateur en ligne de commande

cos: il ne peut pas accéder à toutes les fonctionnalités de l'API Weka, les scripts Jython ou Groovy sont recommandés pour cette tâche.

Table de travail

pro: il rassemble toutes les autres interfaces en un seul endroit

Examples

exemples simpleCLI et Jython

simpleCLI

aller à simpleCLI, entrez le code suivant

```
java weka.classifiers.rules.ZeroR -t path/to/a-file-of-dataset
```

Exemple Jython

les codes de la leçon de cours Advanced Weka MOOC 5.1

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.filters.Filter as Filter
import weka.filters.unsupervised.attribute.Remove as Remove
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")

# remove class attribute
rem = Remove()
rem.setOptions(["-R", "last"])
rem.setInputFormat(data)
dataNew = Filter.useFilter(data, rem)

# output filtered dataset
print(dataNew)
```

Lire Comparaison simple des interfaces de Weka en ligne:
https://riptutorial.com/fr/weka/topic/8042/comparaison-simple-des-interfaces-de-weka

Chapitre 6: Erreurs commises facilement lors de l'utilisation de KnowledgeFlow

Introduction

Weka KnowledgeFlow (KF) est une excellente interface à utiliser. Cependant, le manuel de Weka ne couvre pas tous les détails de l'utilisation de KF. Voici un endroit pour collecter ces petits trucs ou détails que j'ai appris de ces erreurs que j'ai faites ou que je ferai au fil du temps. Un grand merci aux gens de Wekalist (en particulier Mark Hall, Eibe Frank) pour avoir construit un environnement d'apprentissage merveilleux pour Weka!

Remarques

TrainingSetMaker et TestSetMaker

1. un `ClassAssigner` doit être lié entre `ArffLoader` et `TrainingSetMaker` ou `TestSetMaker` .

ArffSaver

2. Afin de sauvegarder le jeu de données dans un fichier arff avec succès, il est plus sûr de définir `relationNameForFilename` sur False dans la configuration d' `ArffSaver` .

Comment utiliser TimeSeriesForecasting dans KnowledgeFlow?

1. Ouvrir knowledgeFlow, charger un ensemble de données avec ArffLoader
2. aller à la configuration, vérifier la perspective de prévision des séries chronologiques, cliquer avec le bouton droit de la souris sur ArffLoader pour l'envoyer à toutes les perspectives
3. aller dans la perspective des prévisions de séries chronologiques pour mettre en place un modèle
4. exécuter le modèle et copier le modèle dans le presse-papiers
5. ctrl + v, et cliquez pour coller le modèle au canevas de processus d'exploration de données
6. enregistrer la prédiction avec les données d'origine avec ArffSaver

---

Examples

Comment ouvrir le fichier KnowledgeFlow directement depuis le terminal

1. ajouter la fonction suivante dans `.bash_profile` , enregistrer et quitter

```
function wekaflstart () {
export R_HOME = / Library / Frameworks / R.framework / Resources
java -Xss10M -Xmx4096M -cp: weka.jar weka.gui.knowledgeflow.KnowledgeFlow "$ 1"
}
```

2. dans un répertoire contenant un fichier `weka.jar` , ouvrez son terminal, exécutez `wekastart "path to a knowledgeflow file"`

Lire Erreurs commises facilement lors de l'utilisation de KnowledgeFlow en ligne: https://riptutorial.com/fr/weka/topic/8053/erreurs-commises-facilement-lors-de-l-utilisation-de-knowledgeflow

Chapitre 7: Instances de chargement

Examples

Fichiers ARFF

Les fichiers ARFF (Format de fichier de relations d'attributs) constituent le format le plus courant pour les données utilisées dans Weka. Chaque fichier ARFF doit avoir un en-tête décrivant à quoi devrait ressembler chaque instance de données. Les attributs pouvant être utilisés sont les suivants:

- Numérique

Nombre réel ou entier.

- Nominal

Les attributs nominaux doivent fournir un ensemble de valeurs possibles. Par exemple:

```
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

- Chaîne

Permet des valeurs de chaîne arbitraires. Généralement traité ultérieurement à l'aide du filtre `StringToWordVector` .

- Rendez-vous amoureux

Permet de spécifier les dates. Comme avec `SimpleDateFormat` de Java, cette date peut également être formatée; il sera par défaut au format ISO-8601.

Un exemple d'en-tête peut être vu comme suit:

```
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Après l'en-tête, chaque instance doit être répertoriée avec le nombre correct d'instances; si une valeur d'attribut pour une instance n'est pas connue a `?` peut être utilisé à la place. L'exemple suivant montre un ensemble d'instances dans un fichier ARFF:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
```

```
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
```

Chargement des fichiers ARFF

Selon la version de Weka utilisée, différentes méthodes de chargement des fichiers ARFF doivent être utilisées.

Weka <3.5.5

L'exemple de code suivant montre comment charger un fichier ARFF:

```
import weka.core.Instances;
import java.io.BufferedReader;
import java.io.FileReader;
...
BufferedReader reader = new BufferedReader(new FileReader("data.arff"));
Instances data = new Instances(reader);
reader.close();
data.setClassIndex(data.numAttributes() - 1);
```

L'index de classe montre quel attribut doit être utilisé pour la classification. Dans la plupart des fichiers ARFF, c'est le dernier attribut, c'est pourquoi il est défini sur `data.numAttributes() - 1` . Si vous utilisez une fonction Weka, telle que `buildClassifier` , vous devez définir l'index de classe.

Weka> = 3.5.5

Dans la dernière version de Weka, il est très facile de charger un fichier ARFF. Cette méthode peut également charger des fichiers CSV et tout autre fichier que Weka peut comprendre.

```
import weka.core.converters.ConverterUtils.DataSource;
...
DataSource source = new DataSource("data.arff");
Instances data = source.getDataSet();
if (data.classIndex() == -1) {
  data.setClassIndex(data.numAttributes() - 1);
}
```

Chargement depuis la base de données

De nombreuses bases de données peuvent être utilisées dans Weka. Tout d'abord, le fichier DatabaseUtils.props doit être modifié pour correspondre à votre base de données; Plus précisément, vous devez fournir le nom, l'emplacement, le port et le pilote corrects de votre base de données.

```
jdbcDriver=org.gjt.mm.mysql.Driver
jdbcURL=jdbc:mysql://localhost:3306/my_database
```

Ensuite, la base de données peut être chargée en utilisant un code simple.

```
import weka.core.Instances;
import weka.experiment.InstanceQuery;
...
InstanceQuery query = new InstanceQuery();
query.setUsername("user");
query.setPassword("pass");
query.setQuery("select * from mytable");
Instances data = query.retrieveInstances();
```

Quelques notes sur le chargement depuis une base de données:

- Assurez-vous que le pilote JDBC correct se trouve dans votre chemin de classe.
- Si vous utilisez Microsoft Access, le pilote JDBC-ODBC fourni avec le JDK peut être utilisé.
- La méthode `InstanceQuery` convertit VARCHAR en attributs nominaux et TEXT en attributs de chaîne. Un filtre, tel que `NominalToString` ou `StringToNormal` , peut convertir les attributs à leur type correct.

Lire Instances de chargement en ligne: https://riptutorial.com/fr/weka/topic/5928/instances-de-chargement

Chapitre 8: Premiers pas avec Jython dans Weka

Introduction

Pourquoi utiliserions-nous Jython dans Weka? 1. Si vous n'êtes pas satisfait de ce que l'explorateur, l'expérimentateur, le KnowledgeFlow, simpleCLI vous permettent de faire, et que vous cherchez quelque chose pour libérer la plus grande puissance de weka; 2. Avec Jython, nous pouvons accéder à toutes les fonctionnalités fournies par l'API Weka, directement dans Weka; 3. Sa syntaxe est de type Python, considéré comme un langage de script adapté aux débutants.

Remarques

Comment configurer Jython dans weka

1. installer la bibliothèque `Jython` et `JFreeChart` partir du gestionnaire de packages Weka;
2. aller au terminal du répertoire de base, entrez `nano .bash_profile`
3. dans le `.bash_profile` , ajoutez une ligne de code comme ci-dessous

   ```
   export Weka_Data=User/Documents/Directory/Of/Your/Data
   ```

4. sauvegarder et quitter
5. à l'intérieur du terminal `source .bash_profile`

Ensuite, redémarrez Weka, accédez aux `tools` et cliquez sur la `Jython console` , et vous pourrez essayer ces exemples ci-dessus.

Examples

Charger et filtrer les données

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.filters.Filter as Filter
import weka.filters.unsupervised.attribute.Remove as Remove
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")

# remove class attribute
rem = Remove()
```

```
rem.setOptions(["-R", "last"])
rem.setInputFormat(data)
dataNew = Filter.useFilter(data, rem)

# output filtered dataset
print(dataNew)
```

Construire un classificateur

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.classifiers.trees.J48 as J48
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = J48()
cls.setOptions(["-C", "0.3"])

# build classifier
cls.buildClassifier(data)

# output model
print(cls)
```

Validation croisée du classificateur

```
# imports
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.classifiers.Evaluation as Evaluation
import weka.classifiers.trees.J48 as J48
import java.util.Random as Random
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = J48()
cls.setOptions(["-C", "0.3"])

# cross-validate classifier
evl = Evaluation(data)
evl.crossValidateModel(cls, data, 10, Random(1))

# print statistics
print(evl.toSummaryString("=== J48 on anneal (stats) ===", False))
print(evl.toMatrixString("=== J48 on anneal (confusion matrix) ==="))
```

Faire une prédiction

```
# imports
```

```
import weka.classifiers.trees.J48 as J48
import weka.core.converters.ConverterUtils.DataSource as DS
import os

# load training data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal_train.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = J48()
cls.setOptions(["-C", "0.3"])

# build classifier on training data
cls.buildClassifier(data)

# load unlabeled data
dataUnl = DS.read(os.environ.get("MOOC_DATA") + os.sep + "anneal_unlbl.arff")
dataUnl.setClassIndex(dataUnl.numAttributes() - 1)

# test compatibility of train/unlabeled datasets
msg = dataUnl.equalHeadersMsg(data)
if msg is not None:
    print("train and prediction data are not compatible:\n" + msg)

# make predictions
for inst in dataUnl:
    dist = cls.distributionForInstance(inst)
    labelIndex = cls.classifyInstance(inst)
    label = dataUnl.classAttribute().value(int(labelIndex))
    print(str(dist) + " - " + str(labelIndex) + " - " + label)
```

Validation croisée des erreurs de classificateur

```
# Note: install jfreechartOffscreenRenderer package as well for JFreeChart library

# imports
import weka.classifiers.Evaluation as Evaluation
import weka.classifiers.functions.LinearRegression as LinearRegression
import weka.core.converters.ConverterUtils.DataSource as DS
import java.util.Random as Random
import org.jfree.data.xy.DefaultXYZDataset as DefaultXYZDataset
import org.jfree.chart.ChartFactory as ChartFactory
import org.jfree.chart.plot.PlotOrientation as PlotOrientation
import org.jfree.chart.ChartPanel as ChartPanel
import org.jfree.chart.renderer.xy.XYBubbleRenderer as XYBubbleRenderer
import org.jfree.chart.ChartUtilities as ChartUtilities
import javax.swing.JFrame as JFrame
import java.io.File as File
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "bodyfat.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = LinearRegression()
cls.setOptions(["-C", "-S", "1"])

# cross-validate classifier
```

```
evl = Evaluation(data)
evl.crossValidateModel(cls, data, 10, Random(1))

# collect predictions
act = []
prd = []
err = []
for i in range(evl.predictions().size()):
    prediction = evl.predictions().get(i)
    act.append(prediction.actual())
    prd.append(prediction.predicted())
    err.append(abs(prediction.actual() - prediction.predicted()))

# create plot
plotdata = DefaultXYZDataset()
plotdata.addSeries("LR on " + data.relationName(), [act, prd, err])
plot = ChartFactory.createScatterPlot(\
    "Classifier errors", "Actual", "Predicted", \
    plotdata, PlotOrientation.VERTICAL, True, True, True)
plot.getPlot().setRenderer(XYBubbleRenderer())

# display plot
frame = JFrame()
frame.setTitle("Weka")
frame.setSize(800, 800)
frame.setLocationRelativeTo(None)
frame.getContentPane().add(ChartPanel(plot))
frame.setVisible(True)
```

Graphique d'affichage

```
# imports
import weka.classifiers.bayes.BayesNet as BayesNet
import weka.core.converters.ConverterUtils.DataSource as DS
import weka.gui.graphvisualizer.GraphVisualizer as GraphVisualizer
import javax.swing.JFrame as JFrame
import os

# load data
data = DS.read(os.environ.get("MOOC_DATA") + os.sep + "iris.arff")
data.setClassIndex(data.numAttributes() - 1)

# configure classifier
cls = BayesNet()
cls.setOptions(["-Q", "weka.classifiers.bayes.net.search.local.K2", "--", "-P", "2"])

# build classifier
cls.buildClassifier(data)

# display tree
gv = GraphVisualizer()
gv.readBIF(cls.graph())
frame = JFrame("BayesNet - " + data.relationName())
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE)
frame.setSize(800, 600)
frame.getContentPane().add(gv)
frame.setVisible(True)

# adjust tree layout
```

```
gv.layoutGraph()
```

Lire Premiers pas avec Jython dans Weka en ligne:
https://riptutorial.com/fr/weka/topic/8046/premiers-pas-avec-jython-dans-weka

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec weka	Community, Gal Dreiman
2	Classification de texte	xro7
3	Comment utiliser CPython Scripting dans Weka?	Daniel
4	Comment utiliser R dans Weka	Daniel
5	Comparaison simple des interfaces de Weka	Daniel
6	Erreurs commises facilement lors de l'utilisation de KnowledgeFlow	Daniel
7	Instances de chargement	SJB
8	Premiers pas avec Jython dans Weka	Daniel