Kostenloses eBook

LERNEN

weka

Free unaffiliated eBook created from
Stack Overflow contributors.

#weka

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit Weka

Bemerkungen

Dieser Abschnitt bietet einen Überblick über das, was weka ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen innerhalb von weka erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für weka neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

Installation oder Setup

Weka ist eine Sammlung von Machine Learning-Algorithmen für Data Mining-Aufgaben. Die Algorithmen können entweder direkt auf ein Dataset angewendet oder von Ihrem eigenen Java-Code aus aufgerufen werden. Weka enthält Werkzeuge zur Datenvorverarbeitung, Klassifizierung, Regression, Clustering, Assoziationsregeln und Visualisierung. Es eignet sich auch für die Entwicklung neuer maschineller Lernprogramme.

Weka herunterladen und installieren

Es gibt zwei Versionen von Weka: Weka 3.8 ist die neueste stabile Version und Weka 3.9 ist die Entwicklungsversion. Für den blutenden Rand können auch nächtliche Schnappschüsse heruntergeladen werden.

Stabile Versionen erhalten nur Fehlerbehebungen, während die Entwicklungsversion neue Funktionen erhält. Weka 3.8 und 3.9 verfügen über ein Paketverwaltungssystem, das es der Weka-Community erleichtert, neue Funktionen zu Weka hinzuzufügen. Das Paketverwaltungssystem erfordert eine Internetverbindung, um Pakete herunterzuladen und zu installieren.

Sie können die Anwendung für Windows / MacOS / Linux hier herunterladen.

Integrieren Sie die WEKA-Bibliothek in Ihren 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'
```

Erste Schritte mit Weka online lesen: https://riptutorial.com/de/weka/topic/3699/erste-schritte-mit-weka

Kapitel 2: Bei der Verwendung von KnowledgeFlow können leicht Fehler gemacht werden

Einführung

Weka KnowledgeFlow (KF) ist eine großartige Benutzeroberfläche. Das Weka-Handbuch enthält jedoch nicht alle kleinen Details zur Verwendung von KF. Hier könnte man die kleinen Tricks oder Details sammeln, die ich aus den Fehlern gelernt habe, die ich je gemacht habe oder machen werde. Vielen Dank an die Leute von Wekalist (insbesondere Mark Hall, Eibe Frank) für den Aufbau einer wunderbaren Lernumgebung für Weka!

Bemerkungen

TrainingSetMaker und TestSetMaker

1. Ein `ClassAssigner` muss zwischen `ArffLoader` und `TrainingSetMaker` oder `TestSetMaker` .

ArffSaver

2. Um Dataset erfolgreich in einer arff-Datei zu speichern, ist es sicherer, `relationNameForFilename` in der Konfiguration von `ArffSaver` auf False zu `ArffSaver` .

Wie nutze ich TimeSeriesForecasting in KnowledgeFlow?

1. KnowledgeFlow öffnen, Dataset mit ArffLoader laden
2. Klicken Sie mit der rechten Maustaste auf ArffLoader, um an alle Perspektiven zu senden
3. Gehen Sie zur Prognoseperspektive für Zeitreihen, um ein Modell einzurichten
4. Führen Sie das Modell aus und kopieren Sie das Modell in die Zwischenablage
5. Strg + V, und klicken Sie, um das Modell in den Data Mining-Prozessbereich einzufügen
6. Speichern Sie die Vorhersage zusammen mit den Originaldaten mit ArffSaver

---

Examples

So öffnen Sie die KnowledgeFlow-Datei direkt vom Terminal aus

1. `.bash_profile` Sie die folgende Funktion in `.bash_profile` , speichern Sie und `.bash_profile` Sie das `.bash_profile`

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

2. `weka.jar` in einem Verzeichnis mit einer Datei `weka.jar` das Terminal und führen `wekastart "path to a knowledgeflow file"`

Bei der Verwendung von KnowledgeFlow können leicht Fehler gemacht werden online lesen: https://riptutorial.com/de/weka/topic/8053/bei-der-verwendung-von-knowledgeflow-konnen-leicht-fehler-gemacht-werden

Kapitel 3: Einfacher Vergleich der Weka-Schnittstellen

Einführung

Weka hat viele Schnittstellen, Explorer, KnowledgeFlow, Experimenter, SimpleCLI, Workbench. Alle Benutzer können meistens die gleichen Aufgaben mit unterschiedlichen Fokus und Flexibilität erledigen. Hier werden wir ihre unterschiedlichen Schwerpunkte und Flexibilitäten untersuchen.

Bemerkungen

Forscher

Profi:

1. mach alles schnell
2. Geben Sie einen schnellen und umfassenden Überblick über die Datenstruktur

cos: kann den Prozess nicht speichern;

Experimentator

Profi:

1. mehrere Modelle gleichzeitig vergleichen, z. B. drei verschiedene Klassifizierer gegen 5 Datensätze ausführen und das Vergleichsergebnis an einer Stelle anzeigen;
2. Experiment kann gespeichert werden

KnowledgeFlow

Profi:

1. fast alles tun, was Explorer kann
2. kann den Prozess speichern

cos:

1. KF kann die Arbeit von Experimenter nicht ausführen, da es keine Schleifen unterstützt. ADAMS kann jedoch helfen.
2. KF kann nicht auf einfache Funktionen innerhalb der Weka API zugreifen.

simpleCLI

pro: Ausführen ähnlicher Aufgaben des Explorers mithilfe der Befehlszeile

cos: Es kann nicht auf alle Funktionalitäten der Weka API zugegriffen werden. Jython oder Groovy-Skripting wird für diese Aufgabe empfohlen.

Werkbank

pro: es sammelt alle anderen Schnittstellen an einem Ort

Examples

simpleCLI- und Jython-Beispiele

simpleCLI

Gehen Sie zu simpleCLI und geben Sie den folgenden Code ein

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

Jython-Beispiel

Codes aus dem Advanced Weka MOOC Kursstunde 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)
```

Einfacher Vergleich der Weka-Schnittstellen online lesen:
https://riptutorial.com/de/weka/topic/8042/einfacher-vergleich-der-weka-schnittstellen

Kapitel 4: Erste Schritte mit Jython in Weka

Einführung

Warum sollten wir Jython in Weka verwenden? 1. Wenn Sie mit dem, was Explorer, Experimenter, KnowledgeFlow, SimpleCLI unzufrieden sind, nicht zufrieden sind, können Sie dies tun und nach etwas suchen, um die größere Kraft von weka zu entfesseln. 2. Mit Jython können wir auf alle von Weka API bereitgestellten Funktionen direkt in Weka zugreifen. 3. Ihre Syntax ist Python-artig, was als Anfänger-freundliche Skriptsprache gilt.

Bemerkungen

So richten Sie Jython in weka ein

1. Installieren Sie die `Jython` und `JFreeChart` Bibliothek über den Weka Package Manager.
2. Gehen Sie zum Terminal des Home-Verzeichnisses und geben Sie `nano .bash_profile`
3. `.bash_profile` in `.bash_profile` eine Codezeile wie `.bash_profile` hinzu

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

4. speichern und schließen
5. In der Terminal-Ausführungsquelle `source .bash_profile`

Starten Sie dann Weka neu, gehen Sie zu `tools` und klicken Sie auf `Jython console` . Sie können diese Beispiele oben ausprobieren

Examples

Daten laden und filtern

```
# 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)
```

Erstellen Sie einen Klassifizierer

```
# 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)
```

Kreuzklassifizierer

```
# 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) ==="))
```

Eine Vorhersage machen

```
# 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)
```

Cross-validate Classifier Error Bubble

```
# 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)
```

Grafik anzeigen

```
# 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()
```

Erste Schritte mit Jython in Weka online lesen: https://riptutorial.com/de/weka/topic/8046/erste-

schritte-mit-jython-in-weka

Kapitel 5: Instanzen laden

Examples

ARFF-Dateien

ARFF-Dateien (Attribute-Relation File Format) sind die am häufigsten verwendeten Daten für Weka-Daten. Jede ARFF-Datei muss einen Header haben, der beschreibt, wie jede Dateninstanz aussehen sollte. Die Attribute, die verwendet werden können, lauten wie folgt:

- Numerisch

Reelle oder ganze Zahlen.

- Nominal

Nominalattribute müssen einen Satz möglicher Werte bereitstellen. Zum Beispiel:

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

- String

Erlaubt beliebige String-Werte. Wird normalerweise später mit dem `StringToWordVector` Filter verarbeitet.

- Datum

Erlaubt die Angabe von Datumsangaben. Wie bei Java `SimpleDateFormat` kann dieses Datum auch formatiert werden. Standardmäßig wird das ISO-8601-Format verwendet.

Ein Beispielkopf sieht wie folgt aus:

```
@RELATION iris

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

Nach dem Header muss jede Instanz mit der richtigen Anzahl von Instanzen aufgelistet sein. Ist ein Attributwert für eine Instanz nicht bekannt `?` kann stattdessen verwendet werden. Das folgende Beispiel zeigt ein Beispiel für die Menge von Instanzen in einer ARFF-Datei:

```
@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
```

Laden von ARFF-Dateien

Abhängig von der verwendeten Weka-Version sollten verschiedene Methoden zum Laden von ARFF-Dateien verwendet werden.

Weka <3.5.5

Der folgende Beispielcode zeigt, wie eine ARFF-Datei geladen wird:

```
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);
```

Der Klassenindex zeigt, welches Attribut zur Klassifizierung verwendet werden soll. In den meisten ARFF-Dateien ist dies das letzte Attribut, weshalb es auf `data.numAttributes() - 1` . Wenn Sie eine Weka-Funktion wie `buildClassifier` , müssen Sie den Klassenindex festlegen.

Weka> = 3,5,5

In der neuesten Version von Weka ist es sehr einfach, eine ARFF-Datei zu laden. Diese Methode kann auch CSV-Dateien und andere Dateien laden, die Weka versteht.

```
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);
}
```

Laden aus der Datenbank

In Weka können viele Datenbanken verwendet werden. Zunächst muss die Datei DatabaseUtils.props so bearbeitet werden, dass sie zu Ihrer Datenbank passt. Insbesondere müssen Sie den Namen, den Ort, den Port und den korrekten Treiber Ihrer Datenbank angeben.

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

Dann kann die Datenbank mit einfachem Code geladen werden.

```
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();
```

Einige Hinweise zum Laden aus einer Datenbank:

- Stellen Sie sicher, dass sich der richtige JDBC-Treiber in Ihrem Klassenpfad befindet.
- Wenn Sie Microsoft Access verwenden, kann der mit dem JDK gelieferte JDBC-ODBC-Treiber verwendet werden.
- Die `InstanceQuery` Methode konvertiert VARCHAR in Nominalattribute und TEXT in Zeichenfolgenattribute. Ein Filter wie `NominalToString` oder `StringToNormal` kann die Attribute wieder in den richtigen Typ konvertieren.

Instanzen laden online lesen: https://riptutorial.com/de/weka/topic/5928/instanzen-laden

Kapitel 6: Textklassifizierung

Examples

Textklassifizierung mit LibLinear

- Erstellen Sie Trainingsinstanzen aus der .arff-Datei

```
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);
```

- Verwenden Sie **StringToWordVector,** um Ihre Zeichenfolgenattribute in eine Zahlendarstellung umzuwandeln:

  * Wichtige Merkmale dieses Filters:

    1. tf-idf Darstellung
    2. Stemming
    3. Kleinbuchstaben
    4. Stoppwörter
    5. N-Gramm-Darstellung *

```
    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);
```

- Wenden Sie den Filter auf trainingData an: `trainingData = Filter.useFilter(trainingData, filter);`
- Erstellen Sie den LibLinear-Klassifizierer

    1. SVMType 0 unten entspricht der L2-regulierten logistischen Regression
    2. Setzen Sie `setProbabilityEstimates(true)` , um die Ausgabefunktionen zu drucken

```
        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);
```

- Modell speichern

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

- Erstellen Sie `.arff` aus der `.arff` Datei

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

- Klassifizierer laden

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

- **Verwenden Sie den gleichen StringToWordVector-Filter wie oben oder erstellen Sie einen neuen für testingData. Denken Sie jedoch daran, die trainingData für diesen Befehl zu verwenden:** **`filter.setInputFormat(trainingData);`** *Dadurch werden Schulungs- und Testinstanzen kompatibel. Alternativ können Sie* *`InputMappedClassifier`*
- Wenden Sie den Filter auf testingData an: `testingData = Filter.useFilter(testingData, filter);`
- Klassifizieren!

    1.Geben Sie den Klassenwert für jede Instanz im Testsatz ab

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

`res` ist ein doppelter Wert, der der nominalen Klasse entspricht, die in der `.arff` Datei definiert ist. Um die nominelle Klasse zu erhalten, verwenden Sie:
`testintData.classAttribute().value((int)res)`

---

2. Ermitteln Sie die Wahrscheinlichkeitsverteilung für jede Instanz

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

`dist` ist ein doppeltes Array, das die Wahrscheinlichkeiten für jede in der `.arff` Datei definierte Klasse `.arff`

Hinweis. Der Klassifizierer sollte Wahrscheinlichkeitsverteilungen unterstützen und diese aktivieren: `myClassifier.setProbabilityEstimates(true);`

Textklassifizierung online lesen: https://riptutorial.com/de/weka/topic/7753/textklassifizierung

Kapitel 7: Wie verwende ich CPython Scripting in Weka?

Bemerkungen

Wie installiere ich CPython in Weka?

Installieren Sie WekaPython

1. Gehen Sie zu `tools` , öffnen Sie den `package manager`
2. Suchen Sie `wekaPython` , wählen Sie und klicken `wekaPython` , um zu installieren

Installieren Sie Python-Bibliotheken

1. Anaconda oder Conda installieren
2. Installieren Sie vier Pakete: numpy, pandas, matplotlib, scikit-learn
3. für die vollständige installation doc siehe conda

Examples

Hallo Weltbeispiel für CPython von Weka

> Gehen Sie zum `Explorer` , öffnen `iris.arff` Daten `iris.arff` , und gehen Sie zu `CPython Scripting` . Kopieren Sie die folgenden `iris.arff` und `iris.arff` in `Python Scripts` :

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

> Um die Ausgabe anzuzeigen, gehen Sie zu `Python Variables` , wählen Sie beispielsweise `hi` aus und klicken `Get text`

Wie verwende ich CPython Scripting in Weka? online lesen:
https://riptutorial.com/de/weka/topic/7921/wie-verwende-ich-cpython-scripting-in-weka-

Kapitel 8: Wie verwende ich R in Weka?

Bemerkungen

Warum R in Weka verwenden?

1. R ist ein leistungsfähiges Werkzeug für die Vorverarbeitung von Daten
2. R hat eine große Anzahl von Bibliotheken und wächst weiter
3. R in Weka kann Daten problemlos abrufen, verarbeiten und nahtlos an Weka übergeben

---

So richten Sie R in Weka ein

Für Mac-Benutzer

1. ersetze die alte info.Plist durch die neue von Mark Hall
2. R herunterladen und installieren
3. Installieren Sie `rJava` in R mit

   Installationspakete ('rJava')
4. Installieren Sie `Rplugin` mit `Weka Package Manager`
5. Gehen Sie zum Ordner `weka 3-8-0` (wenn es sich um die verwendete Version handelt) und öffnen Sie das Terminal und
6. Führen Sie die folgenden zwei Codezeilen aus (Dank an Michael Hall)

   export R_HOME = / Library / Frameworks / R.framework / Resources
   java -Xss10M -Xmx4096M -cp.: weka.jar weka.gui.GUIChooser
7. Um das Leben zu erleichtern, speichern Sie den obigen Code in einem Verzeichnis, in dem Sie mit weka arbeiten möchten, in einer Datei namens `weka_r.sh`
8. Führen Sie im Terminal dieses Verzeichnisses den folgenden Code aus:

   chmod a + x weka_r.sh
9. `weka.jar` Sie `weka.jar` aus weka 3-8-0 in das Verzeichnis ein und führen Sie den folgenden Code aus:

   ./weka_r.sh

Nun können Sie loslegen. Nächstes Mal müssen Sie nur zum Terminal des Verzeichnisses gehen

und `./weka_r.sh` , um R mit Weka zu starten.

Wie erhalte ich Daten von Weka?

Weka vom Terminal aus öffnen :
Gehen Sie in das Verzeichnis von `Weka 3-8-0` , öffnen Sie das Terminal und führen Sie den folgenden Code aus:

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

Daten über Weka Explorer :

1. `preprocess` Panel, klicken Sie auf `open file` , wählen Sie aus einer Datendatei `weka data folder` ;
2. Gehen Sie zum `R console` und geben Sie R-Skripts in das `R console box` .

Daten über Weka KnowledgeFlow :

1. Klicken Sie im Bereich `Data mining processes` auf `DataSources` , um beispielsweise `ArffLoader` auszuwählen, klicken Sie auf die Leinwand.
2. Doppelklicken Sie auf `ArffLoader` , um eine `ArffLoader` zu laden
3. `Scripting` - Panel, klicken `RscriptExecutor` auf Leinwand
4. `option` + Klicken Sie auf `ArffLoader` , wählen Sie das `dataset` und klicken `RScript Executor` dann auf `RScript Executor` , um diese zu verknüpfen
5. Doppelklicken Sie auf `RScript Executor` um ein R-Skript `RScript Executor` , oder
6. Klicken Sie auf `Settings` und wählen Sie `R Scripting` , um die R-Konsole mit den Daten von Weka zu verwenden

---

R-Codes abspielen

1. Laden Sie `iris.arff` mit Explorer oder KnowledgeFlow.
2. Versuchen Sie das `Plotting inside R Console` oben

Examples

Plotten in der R-Konsole

> Die folgenden Codes finden Sie im Weka-Kurs

Da `iris.arff` in weka geladen ist, können Sie in Weka Explorer's `R console` oder Weka KnowledgeFlows `R Scripting` mit den folgenden Codes schöne Plots erstellen:

```
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)
```

Wie verwende ich R in Weka? online lesen: https://riptutorial.com/de/weka/topic/7916/wie-verwende-ich-r-in-weka-

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Weka	Community, Gal Dreiman
2	Bei der Verwendung von KnowledgeFlow können leicht Fehler gemacht werden	Daniel
3	Einfacher Vergleich der Weka-Schnittstellen	Daniel
4	Erste Schritte mit Jython in Weka	Daniel
5	Instanzen laden	SJB
6	Textklassifizierung	xro7
7	Wie verwende ich CPython Scripting in Weka?	Daniel
8	Wie verwende ich R in Weka?	Daniel