無料電子ブック

学習

weka

Free unaffiliated eBook created from
Stack Overflow contributors.

#weka

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

1: wekaをいめる

このセクションでは、wekaのと、なぜがそれをいたいのかをします。

また、wekaのきなについてもし、するトピックにリンクするがあります。 wekaのドキュメンテーションはしいので、それらのトピックのバージョンをするがあるかもしれません。

Examples

インストールまたはセットアップ

Wekaは、データマイニングタスクのためのアルゴリズムのコレクションです。アルゴリズムは、データセットにすることも、のJavaコードからびすこともできます。 Wekaには、データの、、、クラスタリング、ルール、のためのツールがまれています。また、しいのにもしています。

Wekaのダウンロードとインストール

Wekaには2つのバージョンがあります。Weka 3.8はのバージョンで、Weka 3.9はバージョンです。のについては、スナップショットをダウンロードすることもです。

バージョンはバグのみをけり、バージョンはしいをけります。 Weka 3.8および3.9は、Wekaコミュニティがwekaにしいをするのをにするパッケージシステムをえています。パッケージシステムでは、パッケージをダウンロードしてインストールするためにインターネットがです。

Windows / MacOS / Linuxのアプリケーションはこちらからダウンロードできます。

あなたのコードにWEKAライブラリをする

pox.xml

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

グラデル

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

オンラインでwekaをいめるをむ https://riptutorial.com/ja/weka/topic/3699/wekaをいめる

2: KnowledgeFlowをするとにいがじます

き

Weka KnowledgeFlowKFはいやすいインターフェイスです。しかし、Wekaのマニュアルでは、KFのにするすべてのについてはれていません。がつにつれて、がやった、あるいはいからんださなトリックやをめるがあります。 Wekaのらしいをするために、Wekalistの々にMark Hall、Eibe Frankにします

TrainingSetMakerおよびTestSetMaker

1. `ClassAssigner`は、 `ArffLoader`と`TrainingSetMaker`または`TestSetMaker`でリンクするがあります。

ArffSaver

2. データセットをarffファイルにするには、ArffSaverので`relationNameForFilename`をFalseにするが`ArffSaver`です。

KnowledgeFlowでTimeSeriesForecastingをするには

1. knowledgeFlowをき、ArffLoaderでデータセットをロードする
2. にき、パースペクティブをチェックし、ArffLoaderをクリックしてすべてのパースペクティブにする
3. モデルをするのにく
4. モデルをし、モデルをクリップボードにコピーする
5. ctrl + vをクリックしてモデルをデータマイニングプロセスキャンバスにりけます
6. ArffSaverをしてのデータとともにを

Examples

ナレッジフローファイルをターミナルからく

1. のを`.bash_profile`してしてする

```
wekaflstart{
export R_HOME = /ライブラリ/ Frameworks / R.framework / Resources
java -Xss10M -Xmx4096M -cpweka.jar weka.gui.knowledgeflow.KnowledgeFlow "$ 1"
}
```

2. `weka.jar`ファイルをつディレクトリので、そのをき、 `wekastart "path to a knowledgeflow file"`し`wekastart "path to a knowledgeflow file"`

オンラインでKnowledgeFlowをするとにいがじますをむ
https://riptutorial.com/ja/weka/topic/8053/knowledgeflowをするとにいがじます

3: Wekaインターフェースのな

き

Wekaにはくのインターフェース、Explorer、KnowledgeFlow、Experimenter、SimpleCLI、Workbenchがあります。それらのすべてはになるとをもって、じタスクをすることができます。ここでは、さまざまなとについてします。

プロ

1. すべてのことをすばやくやる
2. データをすばやくに

cosプロセスをできません。

プロ

1. にいくつかのモデルをする、えば、5つのデータセットにして3つのなるをしてし、1つのでをる。
2. をすることができます

ナレッジフロー

プロ

1. Explorerができるすべてのことをう
2. プロセスをすることができます

cos

1. KFはループをサポートしていないため、Experimenterのをうことはできませんが、 ADAMS がちます。
2. KFはWeka APIのレベルのにアクセスできません。

シンプルな**CLI**

proコマンドラインをしてExplorerとのタスクをする

cosWeka API、Jython、Groovyスクリプトのすべてのにアクセスすることはできません。

ワークベンチ

プロのすべてのインターフェースをまとめて1つのにめる

Examples

シンプルなCLIとJythonの

シンプルなCLI

simpleCLIにし、のコードをします

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

Jythonの

Advanced Weka MOOCコースレッスン5.1のコード

```
# 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)
```

オンラインでWekaインターフェースのなをむ https://riptutorial.com/ja/weka/topic/8042/wekaインターフェースのな

4: WekaでCPython Scriptingをするには

WeykaにCPythonをインストールするには

wekaPythonをインストールする

1. `tools`にく、 `package manager`く
2. `wekaPython`し、してクリックしてインストールする

Pythonライブラリをインストールする

1. anacondaまたはcondaをインストールする
2. 4つのパッケージをインストールするnumpy、pandas、matplotlib、scikit-learn
3. フルインストールのdocについてはcondaをしてください

Examples

WekaのCPythonのHello Worldの

`Explorer`にき、 `iris.arff`データをき、 `CPython Scripting`にき、のコードをコピーして `Python Scripts`りけます

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

をるには、 `Python Variables`にき、 `hi`して、 `Get text`をクリックし`Get text`

オンラインでWekaでCPython Scriptingをするにはをむ
https://riptutorial.com/ja/weka/topic/7921/wekaでcpython-scriptingをするには-

5: WekaのRのい

なぜWekaでRをうのですか

1. Rはデータをするためのなツールです
2. Rにはくのがあり、をける
3. WekaのRは、にデータをし、し、Wekaにシームレスにすことができます

WekaでRをする

Macユーザーの

1. いinfo.PlistをMark Hallからえられたしいにきえてください
2. Rをダウンロードしてインストールする
3. Rのに`rJava`をインストールする

   install.packages 'rJava'

4. `Weka Package Manager` `Rplugin`して`Rplugin`をインストールする
5. `weka 3-8-0`フォルダしているバージョンのにし、そのをき、
6. の2のコードをしますMichael Hallのおかげで

   export R_HOME = /ライブラリ/ Frameworks / R.framework / Resources
   java -Xss10M -Xmx4096M -cpweka.jar weka.gui.GUIChooser

7. wekaをってしたいディレクトリので、のコードを`weka_r.sh`というのファイルにします
8. にするには、このディレクトリののでのコードをしてください

   chmod a + x weka_r.sh

9. `weka.jar` 3-8-0のweka.jarをディレクトリにりけ、のコードをします。

   ./weka_r.sh

、あなたはくができています。は、ディレクトリのにき、`../weka_r.sh`をしてWekaを`./weka_r.sh`てRをするがあります。

Wekaからデータをけるは

ターミナルからWekaをきます
Weka 3-8-0ディレクトリにWeka 3-8-0 、そのをいて、のコードをします

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

Weka Explorerによるデータ

1. preprocessパネルでopen fileクリックし、 weka data folderからデータファイルをしweka data folder。
2. R consoleパネルにし、 R console box Rスクリプトをし R console box。

Weka KnowledgeFlowによるデータ

1. Data mining processesパネルで、 Data mining processes DataSourcesをクリックしてArffLoaderをし、キャンバスにクリックします。
2. ArffLoaderをダブルクリックしてデータファイルをロードします
3. Scriptingパネルで、 RscriptExecutorをキャンバスにクリックします
4. option + ArffLoaderクリックし、 datasetしてから、 RScript Executorをクリックしてリンクします
5. [RScript Executor]をダブルクリックしてRスクリプトをするか、
6. Settingsをクリックし、 R Scriptingをしてwekaのデータでコンソールをします

Rコードをする

1. ExplorerまたはKnowledgeFlowでiris.arffをロードします。
2. のPlotting inside R ConsoleでPlotting inside R Consoleみてください

Examples

Rコンソールでのプロット

Wekaコースからのコードをつけることができます

iris.arffがweka、Weka ExplorerのR consoleまたはWeka KnowledgeFlowのR Scriptingのにロードされていると、のコードをってしいプロットをることができます

```
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)
```

オンラインでWekaのRのいをむ https://riptutorial.com/ja/weka/topic/7916/wekaのrのい

6: WeyyのJython

き

なぜ**Weka**の**Jython**をうのでしょうか 1.あなたがExplorer、Experimenter、KnowledgeFlow、simpleCLIのようなものにれていない、wekaのきなをするものをしてください。 2. Jythonをすると、Wekaのすぐの Weka APIによってされるすべてのにアクセスできます。 3.そのはPythonのようなもので、にやさしいスクリプトとえられています。

wekaでJythonをする

1. Wekaパッケージマネージャから`Jython`と`JFreeChart`ライブラリをインストールします。
2. ホームディレクトリのにし、 `nano .bash_profile`とします。
3. `.bash_profile` 、のようなコードをします

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

4. して
5. `source .bash_profile`

その、Wekaをし、 `tools`にアクセスして`Jython console`をクリックします。のをすことができます

Examples

データのみみとフィルタリング

```
# 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)
```

クラシファイアをする

```
# 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)
```

クロスバリデーション･クラシファイア

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

をする

```
# 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)
```

クラシファイアエラーバブルをする

```
# 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)
```

グラフの

```
# 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()
```

オンラインでWeyyのJythonをむ https://riptutorial.com/ja/weka/topic/8046/weyyのjython

7: インスタンスのロード

Examples

ARFFファイル

ARFFファイルファイルは、Wekaでされるデータのもなです。ARFFファイルには、データインスタンスがどのようなものであるべきかをするヘッダーがです。できるはのとおりです。

-

または。

- の

は、なのセットをしなければならない。えば

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

-

のをします。、で`StringToWordVector`フィルタをしてされます。

-

をすることができます。 Javaの`SimpleDateFormat`とに、このもフォーマットできます。 ISO-8601にデフォルトされます。

ヘッダーのをにします。

```
@RELATION iris

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

ヘッダーにいて、インスタンスはしいのインスタンスとともにリストされなければなりません。インスタンスのがわからないは`?`わりにすることができます。に、ARFFファイルのインスタンスのセットのをします。

```
@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
```

ARFFファイルのロード

されているWekaのバージョンにじて、ARFFファイルをロードするさまざまなをするがあります。

Weka <3.5.5

のサンプルコードは、ARFFファイルをロードするをしています。

```
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);
```

クラスインデックスは、にどのをすべきかをします。ほとんどのARFFファイルでは、これがので、なぜ`data.numAttributes() - 1`されているのですか。 `buildClassifier`などのWekaをしているは、クラスインデックスをするがあります。

Weka> = 3.5.5

Wekaのバージョンでは、ARFFファイルをみむのがにです。このメソッドは、CSVファイルやWekaができるのファイルもみむことができます。

```
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);
}
```

データベースからのロード

Wekaではくのデータベースをできます。まず、データベースにわせてDatabaseUtils.propsファイルをするがあります。には、データベースの、、ポート、およびしいドライバをするがあります。

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

に、なコードをしてデータベースをロードできます。

```
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();
```

データベースからのみみについての

- しいJDBCドライバがクラスパスにあることをしてください。
- Microsoft Accessをしているは、JDKにのJDBC-ODBCドライバをできます。
- `InstanceQuery`メソッドは、VARCHARをにし、TEXTをにします。などのフィルタ、`NominalToString`または`StringToNormal` 、しいタイプにすをすることができます。

オンラインでインスタンスのロードをむ https://riptutorial.com/ja/weka/topic/5928/インスタンスのロード

8: テキストの

Examples

LibLinearによるテキスト

- .arffファイルからトレーニングインスタンスをする

```
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);
```

- **StringToWordVector**をして、をにします。

  *このフィルターのな

  1. tf-idf
  2. ステミング
  3. のwrods
  4. ストップワード
  5. nグラム*

```
    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);
```

- trainingDataにフィルタをします`trainingData = Filter.useFilter(trainingData, filter);`
- LibLinearクラシファイアをする

    1. のSVMType 0はL2ロジスティックにする
    2. をするには、 `setProbabilityEstimates(true)` をします。

```
        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);
```

- モデルをする

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

- `.arff`ファイルからテストインスタンスをする

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

-

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

- のものとじ**StringToWordVector**フィルタをするか、または**testingData**にしい**StringToWordVector**フィルタをしますが、このコマンドの**trainingData**をれずにしてください **`filter.setInputFormat(trainingData);`** これにより、トレーニングとテストのインスタンスがをつようになります。わりに、 *`InputMappedClassifier`* することもできます
- testingDataにフィルタをします`testingData = Filter.useFilter(testingData, filter);`
- する

    1.テストセットのすべてのインスタンスのクラスをする

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

*res*は、 *.arff*ファイルでされているクラスにする*double*です。クラスをするには、*testintData.classAttribute().value((int)res)*します*testintData.classAttribute().value((int)res)*

2.すべてのインスタンスのをる

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

*dist*は、 *.arff*ファイルでされたすべてのクラスのをむ*double*です

。クラシファイアはをサポートし、それらをにするがあります

myClassifier.setProbabilityEstimates(true);

オンラインでテキストのをむ https://riptutorial.com/ja/weka/topic/7753/テキストの

クレジット

S. No		Contributors
1	wekaをいめる	Community, Gal Dreiman
2	KnowledgeFlowをするとにいがじます	Daniel
3	Wekaインターフェースのな	Daniel
4	WekaでCPython Scriptingをするには	Daniel
5	WekaのRのい	Daniel
6	WeyyのJython	Daniel
7	インスタンスのロード	SJB
8	テキストの	xro7