Бесплатная электронная книга

учусь

weka

Free unaffiliated eBook created from
Stack Overflow contributors.

#weka

Около

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

глава 1: Начало работы с weka

замечания

В этом разделе представлен обзор того, что такое weka, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в рамках weka и ссылки на соответствующие темы. Поскольку документация для weka новая, вам может потребоваться создать начальные версии этих связанных тем.

Examples

Установка или настройка

Weka - это набор алгоритмов машинного обучения для задач интеллектуального анализа данных. Алгоритмы могут быть применены непосредственно к набору данных или вызваны из вашего собственного кода Java. Weka содержит инструменты для предварительной обработки данных, классификации, регрессии, кластеризации, правил ассоциации и визуализации. Он также хорошо подходит для разработки новых схем машинного обучения.

Загрузка и установка Weka

Существуют две версии Weka: Weka 3.8 - последняя стабильная версия, а версия Weka 3.9 - версия для разработки. Для края кровотечения также можно загружать ночные снимки.

Стабильные версии получают только исправления ошибок, а версия разработки получает новые функции. Weka 3.8 и 3.9 имеют систему управления пакетами, которая позволяет сообществу Weka добавлять новые функции в Weka. Для загрузки и установки пакетов система управления пакетами требует подключения к Интернету.

Вы можете скачать приложение для Windows / MacOS / Linux здесь .

Интегрируйте библиотеку WEKA в свой код:

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'
```

Прочитайте Начало работы с weka онлайн: https://riptutorial.com/ru/weka/topic/3699/начало-работы-с-weka

глава 2: Загрузка экземпляров

Examples

Файлы ARFF

Файлы ARFF (формат файла атрибутов) являются наиболее распространенным форматом данных, используемых в Weka. Каждый файл ARFF должен иметь заголовок, описывающий, каков должен быть каждый экземпляр данных. Атрибуты, которые можно использовать, следующие:

- числовой

Реальные или целые числа.

- номинальный

Номинальные атрибуты должны обеспечивать набор возможных значений. Например:

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

- строка

Позволяет произвольные строковые значения. Обычно обрабатывается позже с использованием фильтра `StringToWordVector` .

- Дата

Позволяет указать даты. Как и в случае `SimpleDateFormat` Java, эта дата также может быть отформатирована; он будет по умолчанию соответствовать стандарту ISO-8601.

Пример заголовка можно увидеть следующим образом:

```
@RELATION iris

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

После заголовка каждый экземпляр должен быть указан с правильным количеством экземпляров; если значение атрибута для экземпляра неизвестно `?` может использоваться вместо этого. Ниже приведен пример набора экземпляров в файле 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
```

Загрузка файлов ARFF

В зависимости от используемой версии Weka следует использовать различные методы загрузки файлов ARFF.

Weka <3.5.5

Следующий пример кода показывает, как загрузить файл 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);
```

Индекс класса показывает, какой атрибут следует использовать для классификации. В большинстве файлов ARFF это последний атрибут, поэтому он установлен в `data.numAttributes() - 1` . Если вы используете функцию Weka, такую как `buildClassifier` , вы должны установить индекс класса.

Weka> = 3.5.5

В последней версии Weka очень легко загрузить файл ARFF. Этот метод также может загружать файлы CSV и любые другие файлы, которые Weka может понять.

```
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);
}
```

Загрузка из базы данных

Многие базы данных могут использоваться в Weka. Во-первых, файл DatabaseUtils.props должен быть отредактирован в соответствии с вашей базой данных; в частности, вы должны указать имя, местоположение, порт и правильный драйвер своей базы данных.

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

Затем базу данных можно загрузить с помощью простого кода.

```
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();
```

Некоторые замечания о загрузке из базы данных:

- Убедитесь, что правильный драйвер JDBC находится в вашем пути к классам.
- Если вы используете Microsoft Access, тогда можно использовать JDBC-ODBC-драйвер, который поставляется с JDK.
- Метод `InstanceQuery` преобразует VARCHAR в номинальные атрибуты и TEXT в строковые атрибуты. Фильтр, такие как `NominalToString` или `StringToNormal` , может преобразовывать атрибуты обратно в их правильный тип.

Прочитайте Загрузка экземпляров онлайн: https://riptutorial.com/ru/weka/topic/5928/загрузка-экземпляров

глава 3: Как использовать скрипты CPython в Weka?

замечания

Как установить CPython в Weka?

Установить wekaPython

1. перейти к `tools` , открыть `package manager`
2. поиск `wekaPython` , выберите и нажмите, чтобы установить

Установка библиотек Python

1. установить анаконду или конду
2. установить четыре пакета: numpy, pandas, matplotlib, scikit-learn
3. для полной установки doc см. conda

Examples

Пример Hello World для CPython of Weka

> Перейдите в `Explorer` , откройте данные `iris.arff` , затем перейдите на `CPython Scripting` , Скопируйте и вставьте следующие строки кодов в `Python Scripts` :

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

> Чтобы просмотреть выходные данные, перейдите к `Python Variables` , выберите `hi` , например, и нажмите `Get text`

Прочитайте Как использовать скрипты CPython в Weka? онлайн:
https://riptutorial.com/ru/weka/topic/7921/как-использовать-скрипты-cpython-в-weka-

глава 4: Как пользоваться R в Weka

замечания

Зачем использовать R в Weka?

1. R - мощный инструмент для предварительной обработки данных
2. R имеет огромное количество библиотек и продолжает расти
3. R в Weka, может легко получать данные, обрабатывать их и беспрепятственно переходить на Weka

---

Как настроить R в Weka

Для пользователя Mac

1. замените старый info.Plist на новый, указанный Марк Холл
2. загрузить и установить R
3. установить `rJava` внутри R с

   install.packages ('rJava')

4. установить `Rplugin` с `Weka Package Manager`
5. перейдите в папку `weka 3-8-0` (если это версия, которую вы используете), и откройте ее терминал, и
6. выполните следующие 2 строки кодов (спасибо Майклу Холу)

   экспорт R_HOME = / Библиотека / Рамки / R.framework / Ресурсы
   java -Xss10M -Xmx4096M -cp.: weka.jar weka.gui.GUIChooser

7. чтобы сделать жизнь проще, внутри каталога, где вы хотите работать с weka, сохраните код выше в файл с именем `weka_r.sh`
8. сделайте его исполняемым, внутри терминала этого каталога запустите следующий код:

   chmod a + x weka_r.sh

9. вставьте `weka.jar` из weka 3-8-0 в каталог и запустите следующий код:

./weka_r.sh

Теперь вы готовы идти. В следующий раз вам просто нужно перейти на терминал каталога и запустить `./weka_r.sh` чтобы запустить R с Weka.

Как получить данные от Weka?

открыть Weka из терминала :
перейдите в каталог `Weka 3-8-0` , откройте свой терминал, запустите следующий код:

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

данные через Weka Explorer :

1. панель `preprocess` , нажмите « `open file` , выберите файл данных из `weka data folder` ;
2. перейдите в панель `R console` , введите R-скрипты внутри `R console box` .

данные через Weka KnowledgeFlow :

1. Панель `Data mining processes` , щелкните `DataSources` чтобы выбрать `ArffLoader` например, щелкните его на холсте;
2. дважды щелкните `ArffLoader` чтобы загрузить файл данных
3. Панель `Scripting` , нажмите `RscriptExecutor` на холст
4. `option` + щелкните `ArffLoader` , выберите `dataset` , затем нажмите « `RScript Executor` чтобы связать их
5. дважды нажмите « `RScript Executor` чтобы напечатать сценарий типа R, или
6. нажмите « `Settings` и выберите « `R Scripting` чтобы использовать консоль R с данными Weka

---

Воспроизведение кодов R

1. загрузите `iris.arff` с помощью Explorer или KnowledgeFlow;
2. попробуйте `Plotting inside R Console` примере `Plotting inside R Console` выше

Examples

Построение внутри консоли R

Следующие коды можно найти по курсу Weka

Учитывая, что `iris.arff` загружен в weka, внутри `R console` Weka Explorer или `R Scripting`

Weka KnowledgeFlow, вы можете играть со следующими кодами, чтобы сделать красивые графики:

```
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)
```

Прочитайте Как пользоваться R в Weka онлайн: https://riptutorial.com/ru/weka/topic/7916/как-пользоваться-r-в-weka

глава 5: Начало работы с Jython в Weka

Вступление

Почему мы используем Jython внутри Weka? 1. Если вы недовольны тем, что Explorer, Experimenter, KnowledgeFlow, simpleCLI позволяют вам делать и искать что-то, чтобы развязать большую силу weka; 2. С Jython мы можем получить доступ ко всем функциям, предоставленным Weka API, прямо в Weka; 3. Его синтаксис - Python-подобный, который считается начинающим языком сценариев;

замечания

Как настроить Jython в weka

1. установить библиотеку `Jython` и `JFreeChart` из менеджера пакетов Weka;
2. перейдите на домашний каталог, введите `nano .bash_profile`
3. внутри `.bash_profile` , добавьте строку кода, как показано ниже.

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

4. Сохранить и выйти
5. внутри терминала запустить `source .bash_profile`

Затем перезапустите Weka, зайдите в `tools` и нажмите `Jython console` , и вы можете попробовать эти примеры выше

Examples

Данные загрузки и фильтрации

```
# 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)
```

Построение классификатора

```
# 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)
```

Классификатор перекрестной проверки

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

Сделать прогноз

```
# 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)
```

Перекрестно проверять классификатор 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)
```

Отображаемый график

```
# 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()
```

Прочитайте Начало работы с Jython в Weka онлайн: https://riptutorial.com/ru/weka/topic/8046/начало-работы-с-jython-в-weka

глава 6: Ошибки, легко возникающие при использовании KnowledgeFlow

Вступление

Weka KnowledgeFlow (KF) - отличный интерфейс для использования. Однако руководство Weka не охватывает все мелкие детали использования KF. Здесь будет место для сбора этих маленьких трюков или деталей, которые я узнал из тех ошибок, которые я сделал или сделаю с течением времени. Большое спасибо людям Wekalist (особенно Mark Hall, Eibe Frank) за создание замечательной учебной среды для Weka!

замечания

TrainingSetMaker и TestSetMaker

1. `ClassAssigner` должен быть связан между `ArffLoader` и `TrainingSetMaker` или `TestSetMaker` .

ArffSaver

2. Чтобы успешно сохранить набор данных в файл arff, безопаснее установить `relationNameForFilename` в False внутри конфигурации `ArffSaver` .

Как использовать TimeSeriesForecasting в KnowledgeFlow?

1. Откройте KnowledgeFlow, загрузите набор данных с помощью ArffLoader
2. перейдите к настройке, просмотрите перспективу прогнозирования временных рядов, щелкните правой кнопкой мыши ArffLoader, чтобы отправить все перспективы
3. перейти к прогнозированию временных рядов для создания модели
4. запустить модель и скопировать модель в буфер обмена
5. ctrl + v, и нажмите, чтобы вставить модель в процесс обработки данных.
6. сохранить предсказание вместе с исходными данными с помощью ArffSaver

---

Examples

Как открыть файл KnowledgeFlow непосредственно с терминала

1. добавьте следующую функцию в `.bash_profile` , сохраните и выйдите

```
функция wekaflstart () {
экспорт R_HOME = / Библиотека / Рамки / R.framework / Ресурсы
java -Xss10M -Xmx4096M -cp: weka.jar weka.gui.knowledgeflow.KnowledgeFlow "$ 1"
}
```

2. внутри каталога с файлом `weka.jar` , откройте его терминал, запустите `wekastart "path to a knowledgeflow file"`

Прочитайте Ошибки, легко возникающие при использовании KnowledgeFlow онлайн: https://riptutorial.com/ru/weka/topic/8053/ошибки--легко-возникающие-при-использовании-knowledgeflow

глава 7: Простое сравнение интерфейсов Weka

Вступление

У Weka много интерфейсов, Explorer, KnowledgeFlow, Experimenter, SimpleCLI, Workbench. Все они в основном могут выполнять одни и те же задачи с различной направленностью и гибкостью. Здесь мы рассмотрим их различные фокусы и гибкости.

замечания

исследователь

про:

1. делать все быстро
2. дать быстрый и всеобъемлющий обзор структуры данных

cos: не может сохранить процесс;

Экспериментатор

про:

1. сравните сразу несколько моделей, например, запустите 3 разных классификатора против 5 наборов данных вместе и посмотрите результат сравнения в одном месте;
2. эксперимент может быть сохранен

KnowledgeFlow

про:

1. делать почти все, что может сделать Explorer
2. может сохранить процесс

сов:

1. KF не может выполнять работу Experimenter, поскольку он не поддерживает циклы, но ADAMS может помочь;
2. KF не может получить доступ к низкоуровневым функциям внутри Weka API;

simpleCLI

pro: выполните аналогичные задачи того, что Explorer делает с помощью командной строки

cos: он не может получить доступ ко всем функциям API Weka, для этой задачи рекомендуется использовать сценарии Jython или Groovy.

верстак

pro: он объединяет все другие интерфейсы в одном месте

Examples

Примеры простых CLI и Jython

simpleCLI

перейдите в simpleCLI, введите следующий код

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

Пример Jython

коды из расширенного курса Уэска 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)
```

Прочитайте Простое сравнение интерфейсов Weka онлайн:
https://riptutorial.com/ru/weka/topic/8042/простое-сравнение-интерфейсов-weka

глава 8: Текстовая классификация

Examples

Текстовая классификация с LibLinear

- Создавать учебные экземпляры из файла .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);
```

- Используйте **StringToWordVector,** чтобы преобразовать ваши строковые атрибуты в числовое представление:

  * Важные особенности этого фильтра:

    1. представление tf-idf
    2. вытекающие
    3. нижние строчки
    4. игнорируемые слова
    5. n-граммовое представление *

```
    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);
```

- Применить фильтр к trainingData: `trainingData = Filter.useFilter(trainingData, filter);`
- Создайте классификатор LibLinear

    1. SVMType 0 ниже соответствует L2-регуляризованной логистической регрессии
    2. Установите `setProbabilityEstimates(true)` чтобы напечатать выходные возможности

```
        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);
```

- Сохранить модель

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

- Создавать тестовые экземпляры из файла `.arff`

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

- Классификатор нагрузки

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

- **Используйте тот же фильтр StringToWordVector, как указано выше, или создайте новый для тестированияData, но не забудьте использовать команду trainingData для этой команды:** `filter.setInputFormat(trainingData);` *Это совместит учебные и тестовые экземпляры. В качестве альтернативы вы можете использовать* `InputMappedClassifier`
- Применить фильтр к testData: `testingData = Filter.useFilter(testingData, filter);`
- Классифицировать!

1. Получите значение класса для каждого экземпляра в наборе тестирования

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

`res` - это двойное значение, которое соответствует номинальному классу, который определен в файле `.arff`. Чтобы получить номинальное использование класса: `testintData.classAttribute().value((int)res)`

---

2. Получить распределение вероятности для каждого экземпляра

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

`dist` - это двойной массив, содержащий вероятности для каждого класса, определенного в файле `.arff`

Заметка. Классификатор должен поддерживать распределения вероятности и включать их с: `myClassifier.setProbabilityEstimates(true);`

Прочитайте Текстовая классификация онлайн: https://riptutorial.com/ru/weka/topic/7753/текстовая-классификация

кредиты

S. No	Главы	Contributors
1	Начало работы с weka	Community, Gal Dreiman
2	Загрузка экземпляров	SJB
3	Как использовать скрипты CPython в Weka?	Daniel
4	Как пользоваться R в Weka	Daniel
5	Начало работы с Jython в Weka	Daniel
6	Ошибки, легко возникающие при использовании KnowledgeFlow	Daniel
7	Простое сравнение интерфейсов Weka	Daniel
8	Текстовая классификация	xro7