大略使用方法

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import weka.classifiers.bayes.NaiveBayes;

import weka.core.Instance;

import weka.core.Instances;

import weka.experiment.InstanceQuery;

import weka.filters.Filter;

import weka.filters.unsupervised.attribute.NumericToNominal;

import weka.filters.unsupervised.attribute.Remove;

public class javaproj {

/\*\*

\* @param args

\* @throws Exception

\*/

public static void main(String[] args) throws Exception {

final String url = "jdbc:mysql://localhost/lightdb";

final String user = "root";

final String password = "";

InstanceQuery instanceQuery = null;

Instances data = null;//instances為資料叢集

double result = 0;

//SQL連接

try {

instanceQuery = new InstanceQuery();

instanceQuery.setUsername(user);

instanceQuery.setPassword(password);

instanceQuery.setDatabaseURL(url);

instanceQuery.setQuery("SELECT \* FROM light");

data = instanceQuery.retrieveInstances();

//System.out.println(data);

} catch (Exception e) {

e.printStackTrace();

}

//先移除從資料庫撈出來的資料中 位於每筆資料中第1 4項的資料(不需要ID和TIME) : remover ::option 移除1,,4(INDEX = 1 2 3 4 非0開始)

Remove remove = new Remove();

String[] options = new String[2];

options[0] = "-R";

options[1] = "1,4";

remove.setOptions(options);

remove.setInputFormat(data);

Instances dataClusterer = Filter.useFilter(data, remove);

//修改撈出來的資料中第2項資料的屬性為Norminal

NumericToNominal convert = new NumericToNominal();

String[] optionsc = new String[2];

optionsc[0] = "-R";

optionsc[1] = "2";

convert.setOptions(optionsc);

convert.setInputFormat(dataClusterer);

Instances Dataset = Filter.useFilter(dataClusterer, convert);

//System.out.println(Dataset);

//讀取已經複製到(data資料夾內的.arff訓練檔案)

BufferedReader reader = new BufferedReader(new FileReader("./data/training.arff"));

Instances training = new Instances(reader);

reader.close();

//System.out.print(training);

//此時training.attribute{0:@attribute value numeric 1:@attribute state {0,1}}

//需要訓練的欄位是state{0,1}

training.setClassIndex(training.numAttributes() -1); //未設定會看到arrayoutofbooundexception

//實作分類器 <貝氏分類器>

NaiveBayes nb = new NaiveBayes();

nb.buildClassifier(training);

//或直接用cls代替nb成為分類器

Classifier cls = (Classifier)weka.core.SerializationHelper.read("已訓練完成.arff檔案路徑");

//同樣的已經處理過後的Dataset也需要經過設定欄位於state (index = 1);

Dataset.setClassIndex(Dataset.numAttributes()-1);//未設定會看到arrayoutofbooundexception

//System.out.println(nb);

//for(int i = 0; i < Dataset.numInstances(); i++){//小於資料叢集的大小

//result = nb.classifyInstance(Dataset.instance(i));

//System.out.println(Dataset.instance(i) + " , predict : " + Dataset.classAttribute().value((int)result));

//}

//updateSQL

//針對每一行資料做分類(instance)

//後直接塞回SQL DB

for(int i = 0; i < Dataset.numInstances(); i++){//小於資料叢集的大小

result = nb(cls).classifyInstance(Dataset.instance(i));

instanceQuery.execute("UPDATE light SET status = " +

(Dataset.classAttribute().value((int)result)) + " WHERE id = " + String.valueOf((i+1)));}

//SQL cmd: <UPDATE light SET status 1|0 WHERE id = 1>

}

}