Mineração de Dados na Prática com a Weka API

Article · January 2013

CITATIONS
2

READS
241

1 author:

Eduardo Corrêa Gonçalves
Instituto Brasileiro de Geografia e Estatística
32 PUBLICATIONS **91** CITATIONS

Weka API

Criação de Programas para Mineração de Dados na Linguagem Java

Eduardo Corrêa Gonçalves – SQL Magazine v. 107

De que se trata o artigo:

Mineração de dados é um processo que emprega algoritmos sofisticados para analisar grandes bases de dados, procurando extrair das mesmas informações que estejam implícitas, que sejam previamente desconhecidas e potencialmente úteis para as empresas. Este artigo descreve o passo-a-passo para a execução de processos reais de mineração de dados com o uso da API Java fornecida pela ferramenta livre Weka.

Em que situação o tema útil:

Weka é uma das mais populares ferramentas de mineração de dados. Apesar de muito conhecida, ela é tipicamente utilizada apenas para fins didáticos (em cursos mineração de dados) ou em pequenos experimentos. No entanto, a ferramenta fornece uma API Java bastante poderosa e flexível que permite a sua integração a qualquer tipo de sistema Java. Este artigo destina-se aos profissionais e estudantes de mineração de dados que possuem conhecimento básico sobre a ferramenta Weka e desejem agora utilizá-la em projetos reais de mineração de dados.

Resumo

Este artigo aborda a utilização da Weka API em sistemas desenvolvidos na linguagem Java. O artigo é dividido em duas partes. A primeira apresenta as instruções para obter, instalar a configurar a API. São introduzidas duas maneiras básicas para utilizá-la: (i) integração da biblioteca "weka.jar" ao projeto Java e (ii) integração do código-fonte ao projeto Java. A segunda parte do artigo dedica-se a apresentação de um exemplo onde a Weka API é utilizada em um programa Java que executa as principais etapas de um projeto real de mineração de dados: importação da base de dados, criação do modelo de mineração de dados e aplicação do mesmo sobre novos objetos.

De uma maneira simples, a mineração de dados (*data mining*) pode ser definida como um processo automático que tem por objetivo a descoberta de **conhecimento valioso** em **grandes bases de dados**. Ela baseia-se principalmente na utilização de **algoritmos** que são capazes de vasculhar grandes bases de dados de modo eficiente, revelando **padrões interessantes** que estejam escondidos dentro da "montanha de dados". A Figura 1 ilustra a ideia apresentada.

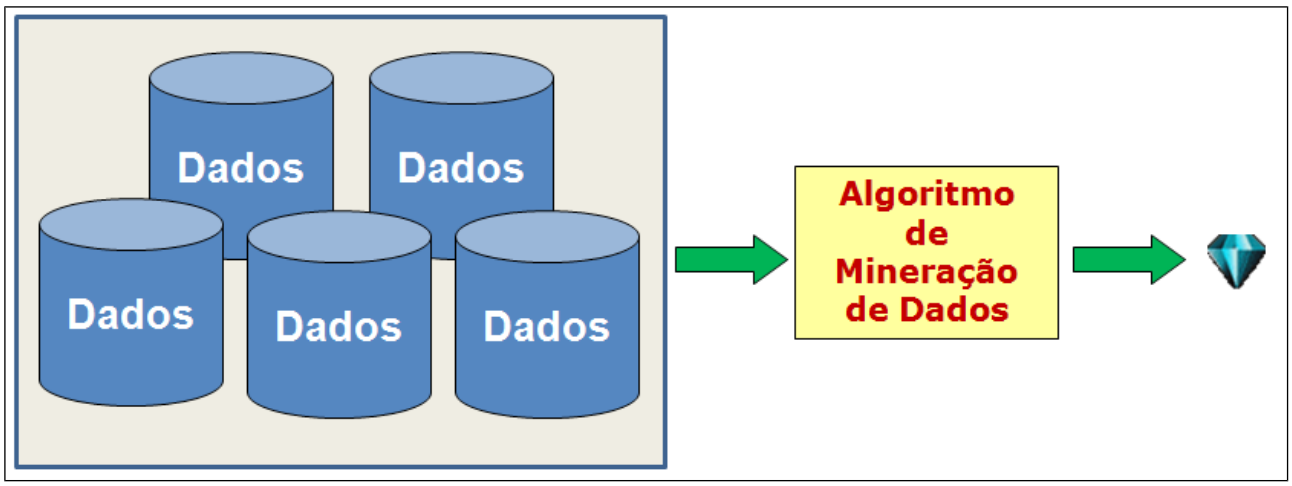

Figura 1. Mineração de dados: o pequeno diamante de informação é extraído de uma "montanha de dados".

Ao longo dos últimos anos, o crescente emprego de processos de mineração de dados pelas empresas motivou o surgimento de algumas dezenas de ferramentas comerciais e livres para este fim. Um dos sistemas que acabou alcançando grande destaque chama-se **Weka**, criado por uma universidade da Nova Zelândia (*The University of Waikato*). Trata-se de um *software* livre do tipo *open source*, desenvolvido em **Java**, dentro das especificações da GPL (*General Public License*). As suas características, bem como os algoritmos nela implementadas são descritas de forma detalhada em [1], cujos autores são os principais responsáveis pela implementação da ferramenta.

A Weka se consolidou como a ferramenta de mineração de dados mais utilizada em ambiente acadêmico. Ela é aplicada não apenas em pesquisas científicas, mas principalmente para **fins didáticos**. Foi exatamente a sua adequação para este último tipo de aplicação que a tornou popular. Professores responsáveis por ministrar cursos de mineração de dados ou *business intelligence* em universidades de todo o mundo, costumam empregar a Weka como instrumento de apoio para o ensino de conceitos básicos sobre *data mining*. Através de sua interface gráfica (conhecida como Weka Explorer) é possível conduzir processos de mineração de pequenas bases de dados, realizando a avaliação dos resultados obtidos e a comparação de algoritmos. Além disso, é possível executar tarefas relacionadas ao pré-processamento de dados como, por exemplo, a seleção e a transformação de atributos.

Embora a Weka tenha um grande número de usuários, a maior parte desconhece que ela disponibiliza uma **API** que torna possível a utilização de suas classes dentro de programas Java. Trata-se do que os autores da ferramenta chamam de "**forma programática**" de utilizar a Weka. Na prática, esta forma programática revela-se bem mais interessante e útil, pois abre possibilidades para a utilização da Weka em **projetos reais** de mineração de dados (e não apenas em projetos acadêmicos/didáticos). Embora seja possível encontrar muitos tutoriais sobre a Weka na Internet, a maioria possui enfoque didático, ou seja, explica apenas como utilizar a interface gráfica da Weka (Weka Explorer) para realizar pequenos experimentos com o intuito de apresentar conceitos básicos de mineração de dados. É muito difícil encontrar textos que abordem Weka sob uma perspectiva mais sólida, mostrando como integrar a sua API a sistemas desenvolvidos em Java, com o intuito de resolver problemas práticos do mundo real.

Este artigo tem por objetivo cobrir esta lacuna. O artigo apresenta as principais informações necessárias para que você possa utilizar a Weka API dentro de seus próprios programas Java, podendo assim aplicar a mineração de dados de forma prática. O artigo está dividido em duas partes. A primeira descreve os passos necessários para configurar a Weka API em seu ambiente de desenvolvimento. A segunda parte dedica-se à apresentação dos passos necessários para criar um programa Java que utiliza a Weka API para executar um processo completo de mineração de dados (o exemplo em questão envolve a tarefa de **classificação**, talvez a mais utilizada na mineração de dados). Através dos exemplos apresentados, você aprenderá a importar e manipular bases de dados ARFF, criar modelos através da execução de algoritmos de mineração de dados e aplicar estes modelos a novos objetos.

Weka API – Configuração

Esta seção descreve como realizar as configurações necessárias para possibilitar a utilização da Weka API em seus programas Java. Inicialmente, apresenta-se o roteiro para o download e instalação da ferramenta. Em seguida, são mostradas duas maneiras distintas para trabalhar com a API: (i) incorporar diretamente a biblioteca "weka.jar" ao seu projeto e (ii) configurar o código-fonte da Weka em seu projeto (neste caso, o exemplo envolve a utilização da IDE Eclipse).

Download e Instalação

A seguir são apresentadas as instruções para o download e instalação da ferramenta Weka. Para utilizar a ferramenta é preciso que você possua a **versão 1.6** ou **superior** do **Java** instalada em seu computador.

Passo 1: Acesse o endereço http://www.cs.waikato.ac.nz/ml/weka/ e clique no link "Download", que encontra-se destacado na Figura 2 (canto esquerdo da tela).

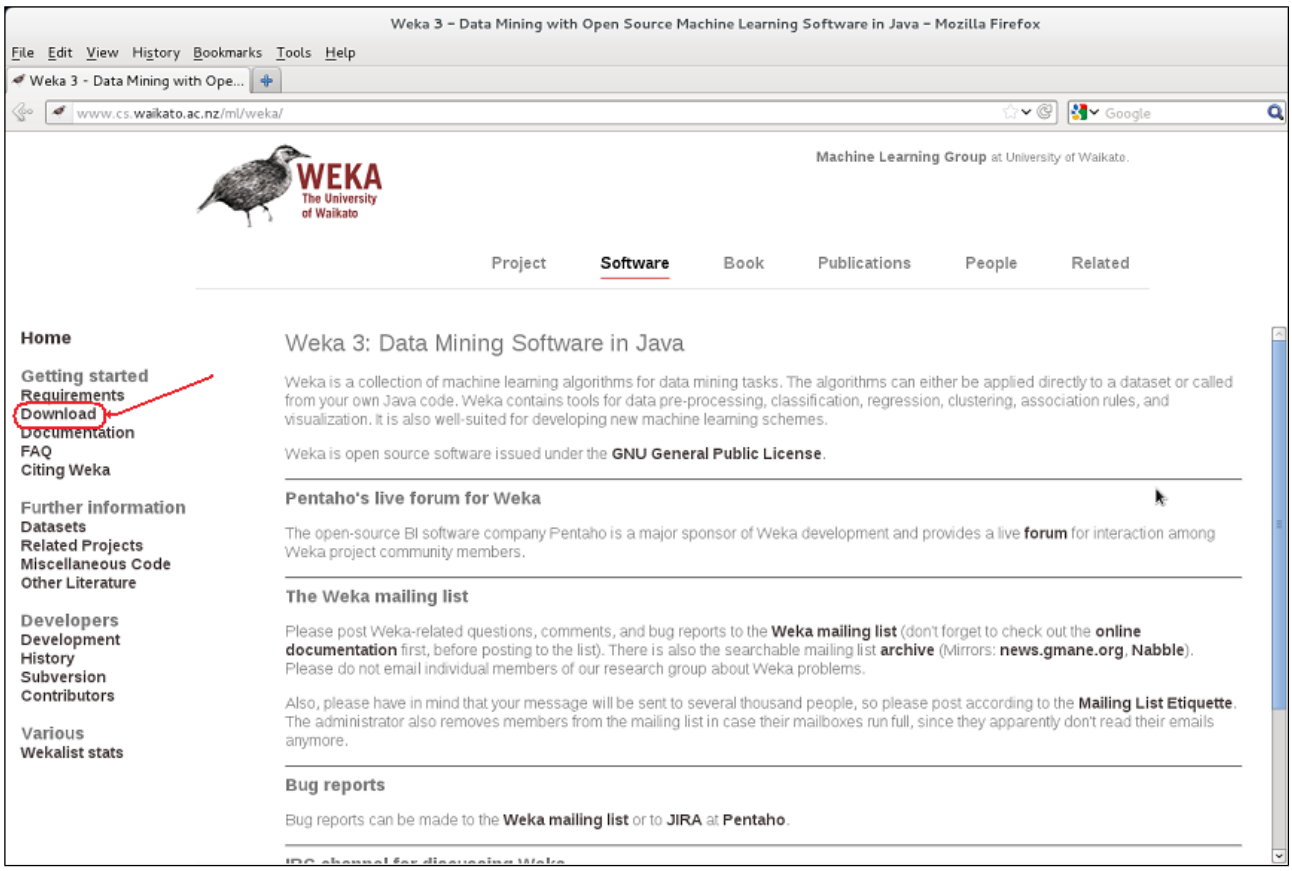

Figura 2. Web site da ferramenta Weka.

Passo 2: Você será levado para a página onde existem opções de download para diferentes versões da ferramenta. Com relação à plataforma, estão disponibilizadas versões em 32 e 64 bits para Windows, Linux e Mac. Além disso, também há uma divisão entre **versão estável** ("Stable book 3rd ed. Version") e **versão de desenvolvimento** ("Developer Version"). **Faça o download da versão estável** referente à sua plataforma (Figura 3), pois ela é mais simples para se trabalhar e suficiente para todos os exemplos apresentados neste artigo. O problema da versão de desenvolvimento é que normalmente ela irá conter código que ainda não foi completamente testado, além de requerer um ambiente de trabalho mais "complicado" (ex: uso de sistemas de controle de versão como o Subversion).

OBS: no caso da plataforma Windows, existe a possibilidade de fazer o download de um instalador que inclui a JVM 1.6. No entanto, você não fará o download deste instalador. Como o artigo trata do uso Weka API em programas Java, assume-se que você já possui o ambiente de desenvolvimento Java (JDK, versão 6 ou superior) instalado em sua máquina.

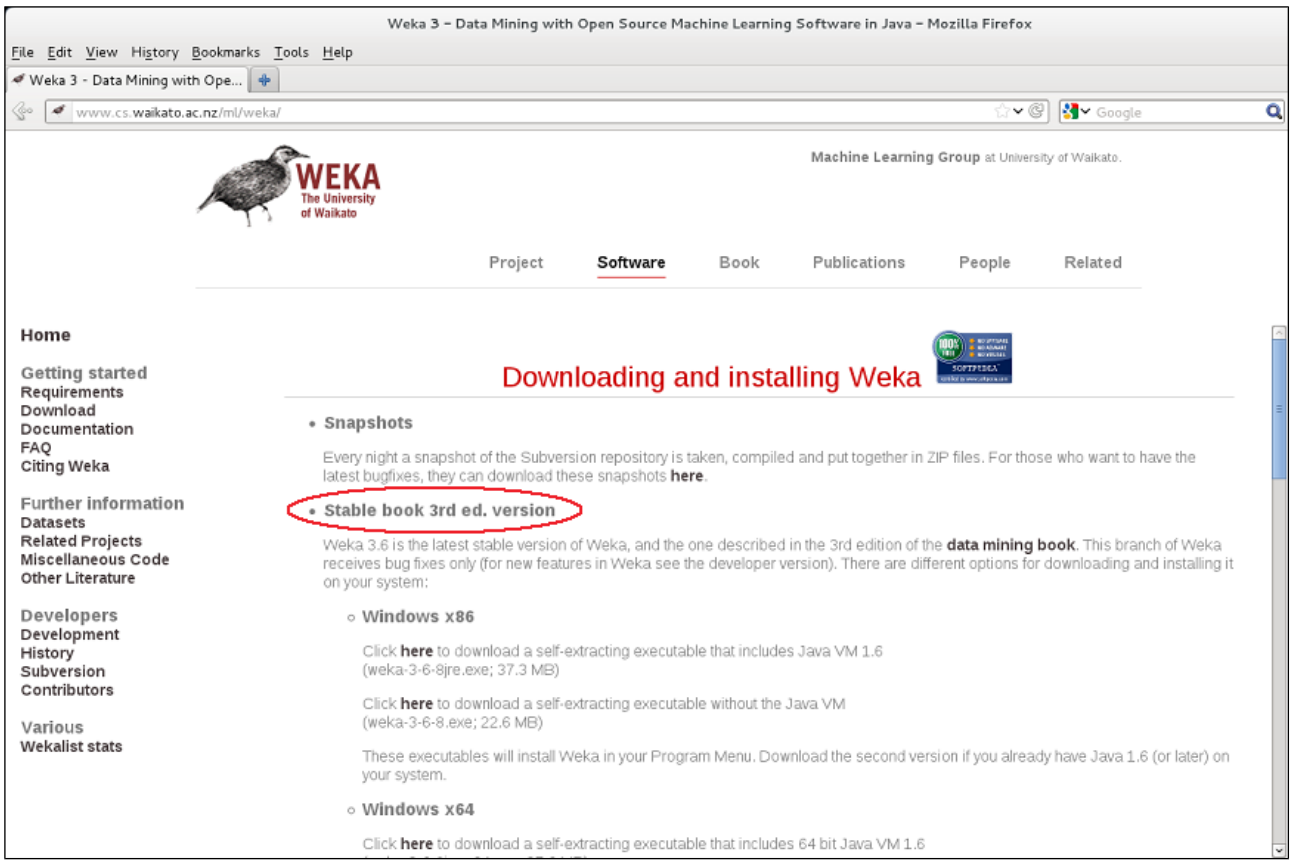

Figura 3. Opção de download da versão estável da ferramenta.

Passo 3: Após clicar na opção de versão desejada, você será redirecionado para uma página do sourceforge e poderá efetuar o download do arquivo de instalação. No momento em que este artigo era desenvolvido, a versão 3-6-8 correspondia à versão estável mais recente da ferramenta. Para fazer a instalação no Windows, basta dar dois cliques no executável e seguir as instruções apresentadas na tela. Normalmente, o programa será instalado na pasta "Arquivos de Programas\Weka-3-6-8". Já para realizar a instalação no Linux, basta apenas descompactar o arquivo ".zip" em algum local do seu computador (ex: dentro do home, crie uma pasta "Weka-3-6-8").

Testando a Instalação

Dentre os vários arquivos que armazenados na pasta da Weka, os dois mais importantes são:

- **weka.jar**: biblioteca que contém os executáveis da Weka (arquivos ".class").
- **weka-src.jar**: arquivo com os códigos-fonte da Weka (arquivos ".java").

Para testar se a instalação foi bem-sucedida, basta fazer uma chamada para execução do arquivo "weka.jar". Tanto no ambiente Windows como no Linux, você pode abrir uma janela de prompt de comando, navegar até a pasta onde a Weka foi instalada e digitar o comando abaixo (se você usa Windows, também é possível obter o mesmo resultado efetuando o duplo-clique sobre o arquivo "RunWeka.bat").

```
C:\Program Files\Weka-3-6-8> java -Xmx1000M -jar weka.jar
```

Esta linha de comando fará com que a interface Weka Gui Chooser seja disparada (Figura 4).

Figura 4. Weka GUI Chooser

Para prosseguir com o teste, clique no botão "Explorer" que fará a execução da Weka Exporer – a interface interativa de mineração de dados da Weka (Figura 5).

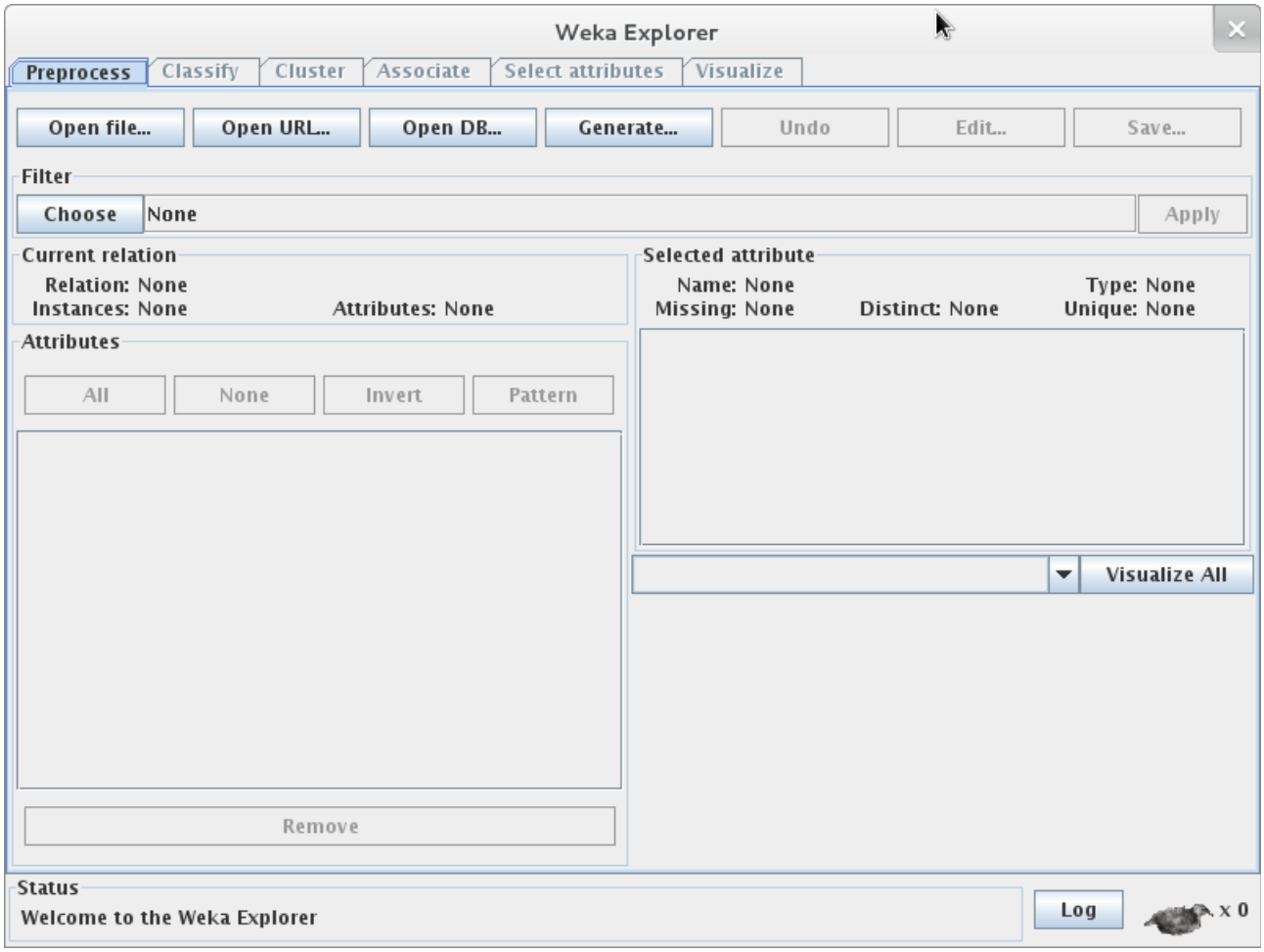

Figura 5. Weka Explorer

Dentro da pasta da Weka existe uma subpasta denominada "data" que possui diversas bases de dados exemplo. Estas bases possuem a extensão ".ARFF", que é o formato de trabalho padrão da Weka (ver Nota 1). Clique no botão "Open File...", navegue até a pasta "data" e abra, por exemplo, a base de dados "vote.arff". Isto fará com que a base seja importada e as suas propriedades sejam exibidas na tela, conforme

mostrado na Figura 6.

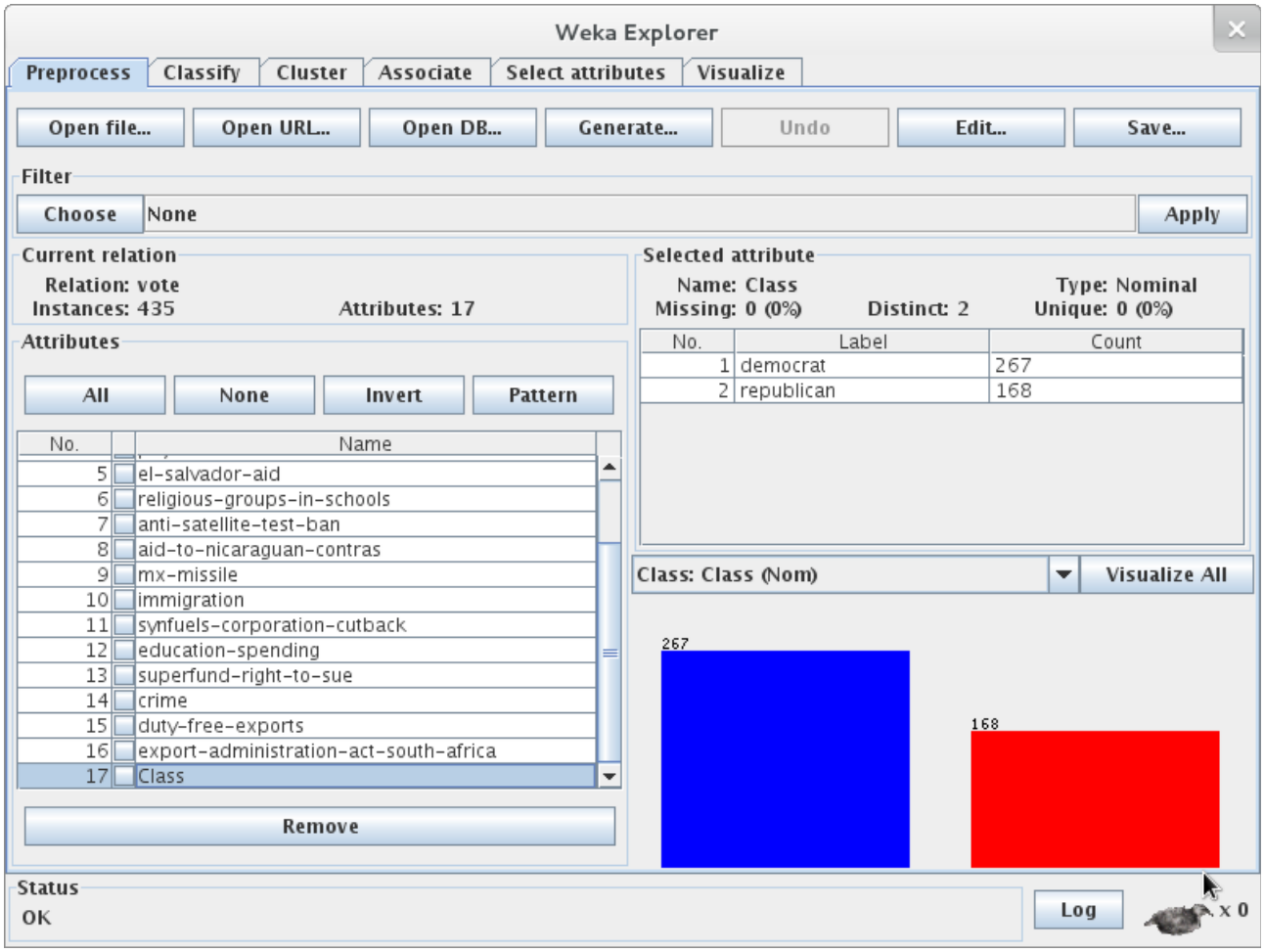

Figura 6. Weka Explorer após a importação da base de dados "vote.arff"

Conforme mencionado na introdução deste artigo, existem inúmeros tutoriais na Internet que ensinam como utilizar o aplicativo Weka Explorer para realizar experimentos interativos de mineração de dados. Este artigo **não** abordará este tema, pois o nosso objetivo é ensinar como utilizar Weka API em programas Java. Para conhecer mais sobre a Weka Explorer, consulte os textos disponibilizados em [2], [3] e [4].

Nota 1. O Formato ARFF

A ferramenta Weka trabalha preferencialmente com arquivos de entrada no formato ARFF (Attribute Relation File Format), que corresponde a um arquivo texto contendo um conjunto de instâncias, precedido por um pequeno cabeçalho. O cabeçalho é utilizado para fornecer informações a respeito dos atributos que compõem o conjunto de instâncias. Abaixo apresenta-se um exemplo de arquivo ARFF, contendo dados de clientes de uma seguradora de veículos hipotética, contendo um cabeçalho e um conjunto de 6 instâncias.

```
@relation Seguradora
@attribute sexo {F, M}
```

```
@attribute faixa_etaria {<25, 25-30, 31-40, >40}
@attribute acidente {Sim, Nao}
@data
F,>40,Nao
M,<25,Sim
F,<25,Nao
F,31-40,Nao
M,25-30,Nao
M,<25,Sim
```

O cabeçalho contém a declaração da relação que o arquivo representa (tag @relation), uma lista de atributos (tag @attribute) e a relação de valores que que cada atributo pode assumir (se o atributo for numérico seus valores não são especificados). No exemplo, os atributos são "sexo", "faixa_etaria" e "acidente" (indica se o cliente já sofreu algum tipo de acidente com o veículo). Os dados propriamente ditos são precedidos por uma tag @data. Cada instância é representada em uma linha. Os valores dos campos dentro de uma instância devem ser separados utilizando a vírgula.

As subseções a seguir descrevem dois métodos básicos para utilizar a API da Weka. O método 1 é bem mais simples, consistindo apenas em acoplar a biblioteca "weka.jar" diretamente em seus projeto. O método 2 é menos simples, porém mais interessante, consistindo em estruturar o código fonte da Weka em um projeto da IDE Eclipse. Neste último caso, embora o exemplo envolva o Eclipse, os passos podem ser facilmente adaptados para outra IDE (ou até para quem não utiliza IDE nenhuma).

Método 1: incorporando a biblioteca "weka.jar" em seu sistema Java

O método mais elementar para utilizar a API da Weka consiste simplesmente em **incorporar a biblioteca** da Weka ("weka.jar") ao seu sistema e utilizá-la como uma "caixa-preta", ou seja, chamando os algoritmos e obtendo os resultados. Mostraremos como fazer isso através da configuração de um ambiente exemplo e da confecção de um pequeno programa dentro deste ambiente, programa este que faz uso da Weka API. Execute os seguintes passos:

Passo 1: Crie uma pasta em seu computador (ex: "C:\mining") e copie a biblioteca "weka.jar" para dentro desta pasta. Para facilitar, também copie para o mesmo local a base de dados "weather.arff", localizada dentro da subpasta "data".

Passo 2: Digite o programa "TesteWeka.java", mostrado na Listagem 1, e salve-o no mesmo local onde estão os arquivos "weka.jar" e "weather.arff". O programa "TesteWeka.java" apresenta o método padrão para importar arquivos ARFF com o uso de duas classes da API da Weka: DataSource e Instances. Maiores informações sobre estas classes são apresentadas na parte 2 deste artigo ("Um Exemplo Prático").

```
import weka.core.converters.ConverterUtils.DataSource;
import weka.core.Instances;

public class TesteWeka {

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

    //importa a base de dados ARFF utilizando classes da Weka
    DataSource source = new DataSource("weather.arff");
    Instances D = source.getDataSet();

    //imprime informações associadas à base de dados
    System.out.println("Num. instancias:" + D.numInstances());
    System.out.println("Num. atributos:" + D.numAttributes());

    //imprime o conteúdo da base
    System.out.println("Base de Dados:");
    System.out.println(D.toString());
  }

}
```

Listagem 1. TesteWeka.java

Passo 3: Agora devemos **compilar** o programa Java. Como tanto a biblioteca "weka.jar" como o programa "TesteWeka.java" estão armazenados na mesma pasta, isto pode ser feito através da linha de comando a seguir (se você estiver utilizando o ambiente Linux, troque o ponto-e-vírgula ";" por dois pontos ":").

```
C:\mining> javac -cp .;weka.jar TesteWeka.java
```

Passo 4: Para **executar** o programa, o processo é análogo, bastando digitar a linha de comando apresentada a seguir:

```
C:\mining> java -cp .;weka.jar TesteWeka
```

O resultado da execução do programa é apresentado na Figura 7. O número de instâncias da base de dados "weather.arff" e o número de atributos da mesma são impressos. A seguir, o programa também imprime todo o conteúdo da base de dados na tela.

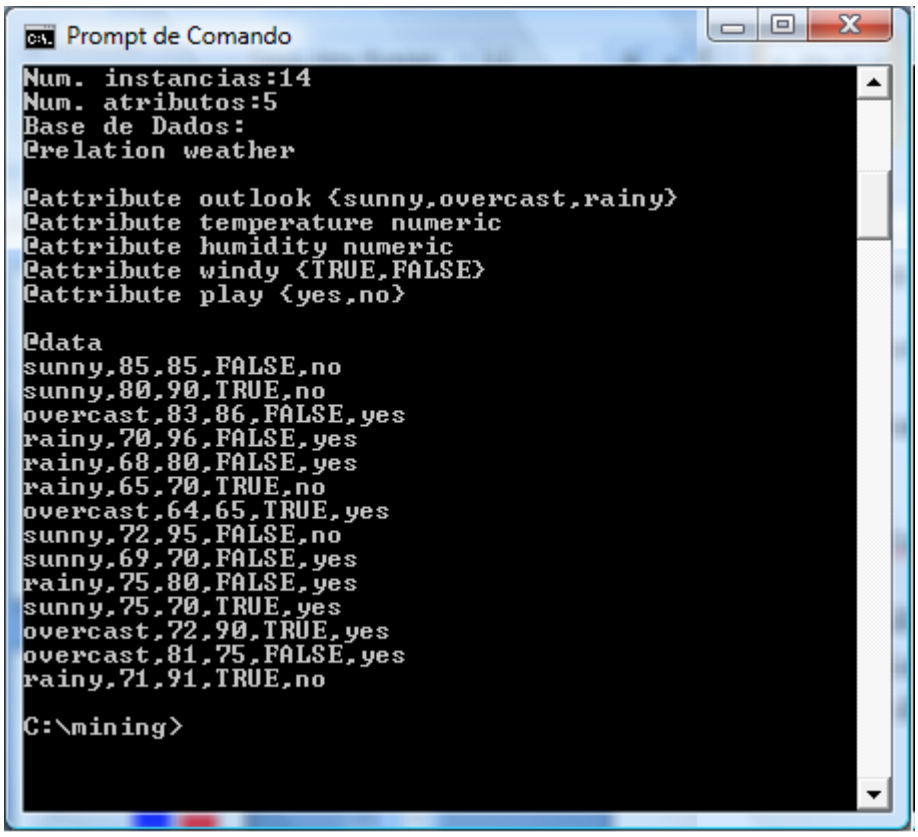

Figura 7. Resultado da execução do programa "TesteWeka.java"

Nota 2. Classpath

No exemplo apresentado, o parâmetro –cp (ou –classpath) utilizado para a compilação e execução do programa "TesteWeka" serve para indicar ao Java quais são as pastas onde ele deve procurar pelas classes que compõem o seu programa. Após o –cp você deve especificar uma lista de diretórios ou de arquivos JAR separados pelo caractere " ; " (ponto-e-vírgula) caso o seu sistema seja Windows ou " : " (dois pontos) no ambiente Linux).

Observe com calma as linhas de comando mostradas nos Passos 3 e 4 do exemplo. Veja que o primeiro item após o –cp é " . " (um ponto!), que significa o diretório corrente. Já o segundo item é "weka.jar", o arquivo JAR com a Weka API. O resultado disto é: o Java irá procurar por classes tanto no diretório corrente, como "dentro" do arquivo "weka.jar". O uso do ponto " . " como primeira entrada após o parâmetro –cp assume que o arquivo "TesteWeka.java" está no diretório corrente ("C:\mining") e faz com que o mesmo possa localizado pelo compilador.

Método 2: Estruturando o código-fonte da Weka no Eclipse

Outra forma de trabalhar com a Weka de maneira programática consiste em **estruturar** todo o **código-fonte** da ferramenta dentro do projeto de um sistema Java. Nesta seção mostraremos como isto é possível, utilizando um exemplo que envolve um projeto criado na IDE **Eclipse** [5]. A opção

se deve ao fato de esta IDE ser a mais utilizada para o desenvolvimento de sistemas em Java. No entanto, as instruções podem ser facilmente adaptadas para o NetBeans ou qualquer outra IDE. E ainda mais: conhecendo a estrutura de pacotes da Weka, você pode até mesmo usar o código-fonte dela sem precisar estar trabalhando com uma IDE.

Trabalhar com o código-fonte da Weka pode ser mais interessante do que simplesmente acoplar a biblioteca "weka.jar" ao seu sistema, pois você não ficará limitado a apenas utilizar os algoritmos da Weka, mas poderá também **estudá-los** e, até mesmo, **modificá-los**. Esta forma de trabalho é recomendada especialmente para estudantes e profissionais de mineração de dados, pois estes normalmente precisam entender o funcionamento dos algoritmos utilizados em seus projetos / estudos. Para configurar o código-fonte da Weka no Eclipse você poderá executar os seguintes passos:

Passo 1: Abra o Eclipse e crie um novo projeto, que pode, por exemplo, se chamar "MyWeka" (File -> New –> Project). Com isto, o Eclipse criará automaticamente uma pasta chamada "MyWeka" (no workspace default de seu computador) e dentro dela duas subpastas: "bin" e "src".

Passo 2: Mantenha o Eclipse aberto, mas esqueça dele por um tempinho. Agora você irá até a pasta onde instalou a ferramenta Weka e abrirá o arquivo "weka-src.jar". Recorde que este arquivo contém o código-fonte da Weka. Você poderá examinar os arquivos contidos na "weka-src.jar" utilizando com algum utilitário descompactador (se você quiser, renomeie o arquivo para "weka-src.zip" para tornar a sua tarefa mais fácil e abra o arquivo com o WinRAR ou o WinZIP). Após abrir o arquivo, navegue até a subpasta "src/main/java/weka". Neste local, existem 10 subpastas, conforme mostra a Figura 8. Cada subpasta representa um diferente **pacote** da Weka API.

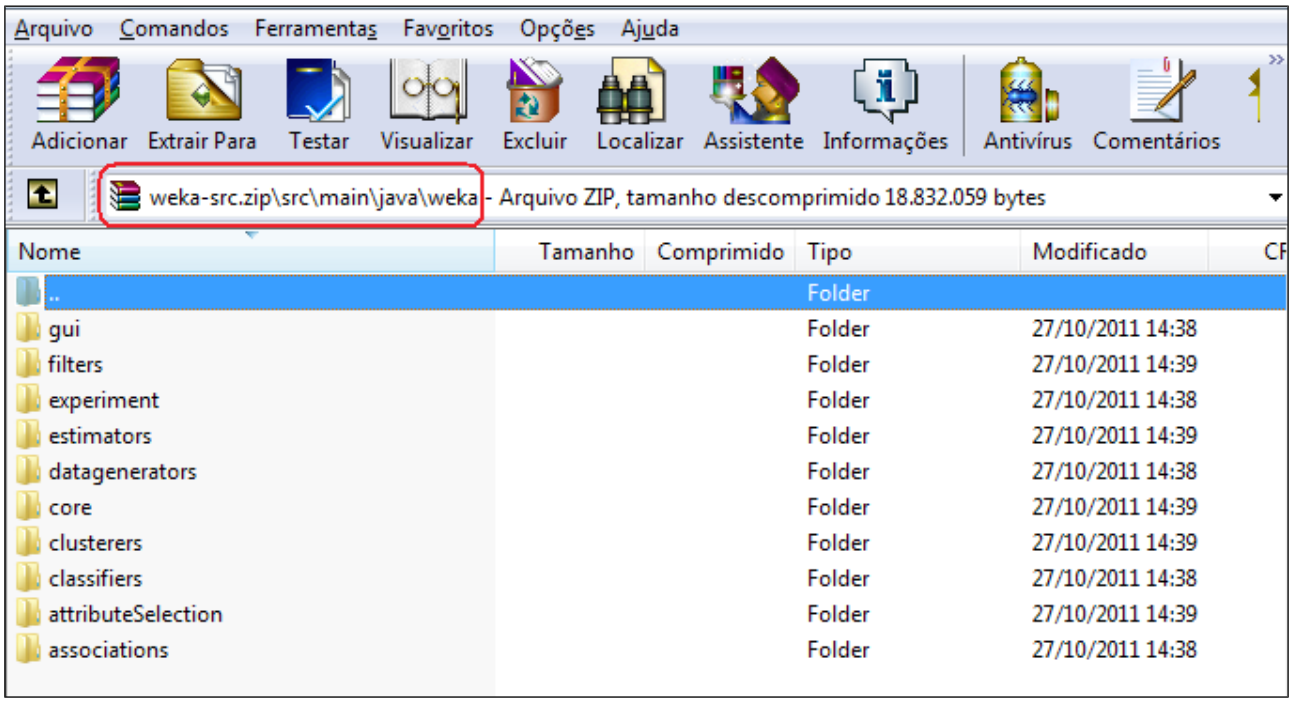

Figura 8. Pastas que contêm o código-fonte da Weka no arquivo "weka-src.jar"

Passo 3: Utilizando o programa descompactador, realize a extração dos arquivos **a partir da subpasta "src/main/java/weka"**. O **destino** deverá ser a pasta "**MyWeka/src**" que foi criada pelo Eclipse. IMPORTANTE: é a subpasta "weka" que deve ficar dentro da "src" - veja a Figura 9.

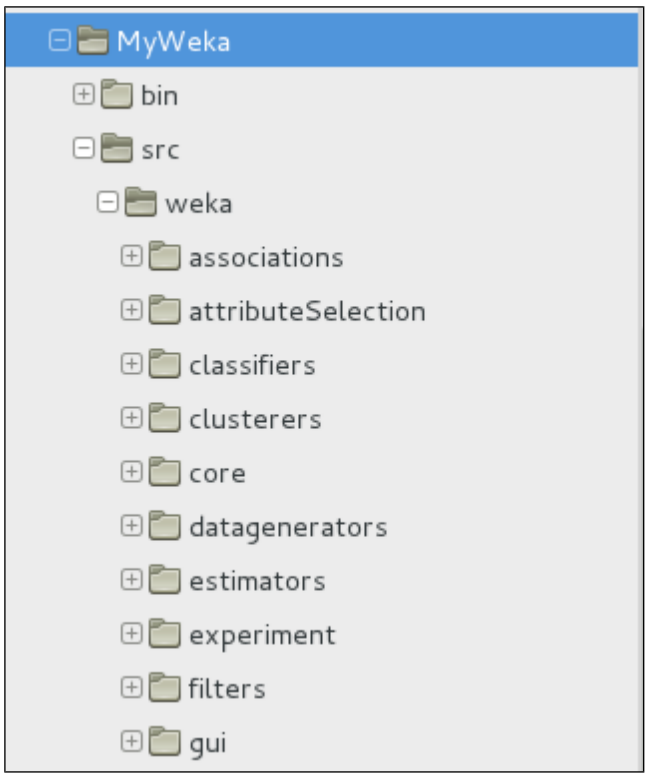

Figura 9. Estrutura de pastas do projeto “MyWeka” após a descompactação das pastas com o código-fonte.

Passo 4: Volte para o Eclipse, clique no projeto “MyWeka” e efetue o “refresh” (digite F5). Os pacotes da Weka serão exibidos, conforme mostrado na Figura 10. Porém você observará que o projeto vai acusar um erro. O problema é que alguns fontes fazem referência para bibliotecas contidas em um pacote chamado “weka.cup.runtime” que ainda não estão incluídas em seu projeto. Felizmente, esse problema é simples de corrigir, como veremos no passo a seguir.

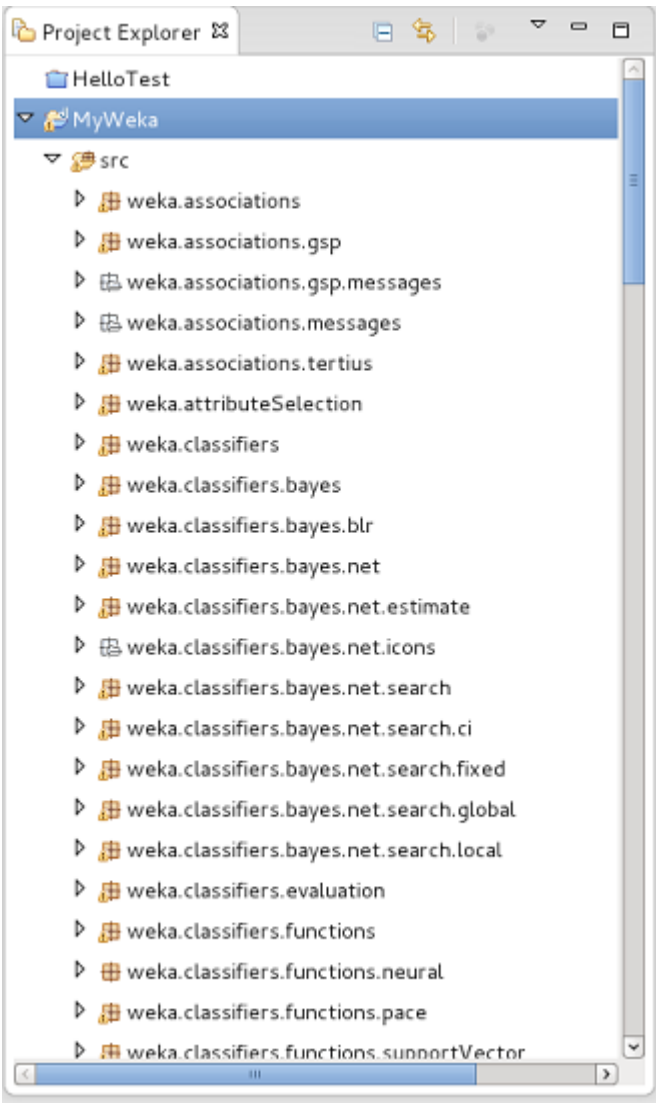

Figura 10. Código-fonte da Weka estruturado em um projeto do Eclipse (Pacotes)

Passo 5: Saia do Eclipse e volte ao arquivo "weka_src.jar". Nele, existe uma pasta "lib". Extraia essa pasta, inserindo-a na pasta que o Eclipse criou para o projeto, conforme indicado na Figura 11. Dentro de "lib", existem três bibliotecas: "java-cup.jar", "JFlex.jar" e "junit.jar".

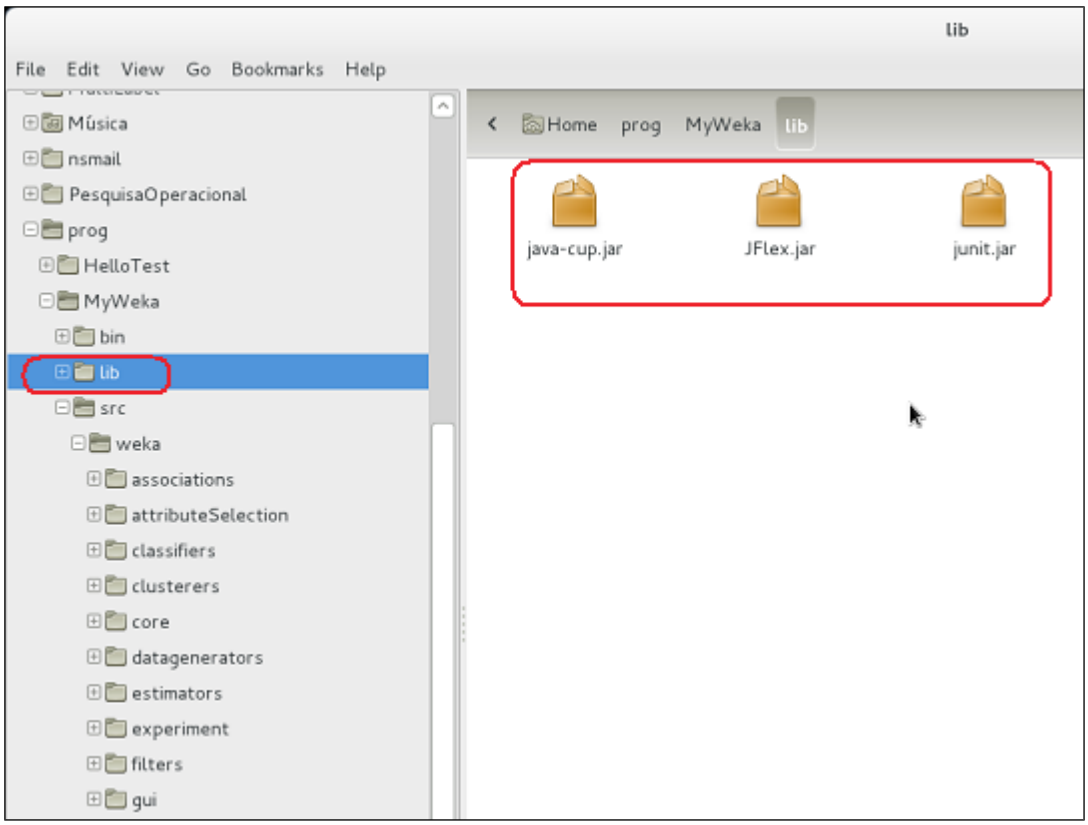

Figura 11. Inserindo a pasta “lib” no projeto

Passo 6: Volte para o Eclipse e faça um novo refresh no projeto. Será necessário agora acrescentar as bibliotecas mostradas na Figura 11 ao “Build-Path” do projeto “MyWeka”. Basta clicar com o botão direito sobre o projeto e selecionar: “Build-path -> Configure Build-path -> Add Jars...”. Adicione então os três arquivos JAR (Figura 12). Efetue um refresh no projeto e, após alguns segundos, estará tudo Ok.

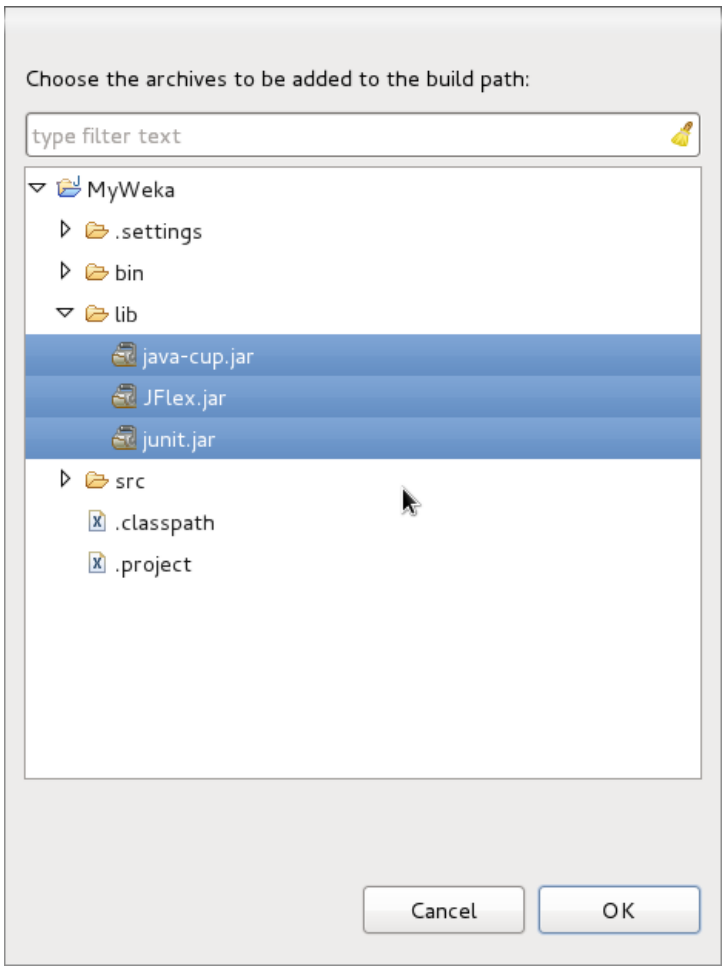

Figura 12. Adicionando bibliotecas ao build-path

Após a execução do Passo 6 a configuração estará finalizada. O código-fonte da Weka estará configurado sem erros em seu projeto. Para testar a configuração, você pode criar um pacote dentro do projeto "MyWeka" - por exemplo "weka.myprogs" - e digitar o programa da Listagem 1 ("TesteWeka.java"), salvando-o neste pacote. Execute o programa no Eclipse e veja o resultado.

Os profissionais de mineração de dados certamente acharão muito interessante navegar pelos pacotes mostrados na Figura 10 para poder examinar as implementações da Weka para diversos algoritmos de mineração de dados. Veja alguns exemplos de pacotes:

- O pacote "**classifiers**" contém implementações de diversos algortimos de classificação. Por exemplo: o pacote "classifiers.bayes" possui implementações de classificadores bayesianos, como o Naïve Bayes ("NaiveBayes.java"); No pacote "classifiers.lazy", encontra-se a implementação do famoso algoritmo k-NN ("IBk.java"); Já em "classifiers.trees" estão os algoritmos de árvore de decisão, como ID3 ("Id3.java") e Cart ("SimpleCart.java").

- O pacote "**associations**" contém as implementações de algoritmos para mineração de regras de associação e padrões sequenciais. Um exemplo é o famoso algoritmo Apriori ("Apriori.java").

- O pacote "**clusteres**" contém diversos algoritmos para análise de agrupamentos (clustering), incluindo o básico algoritmo k-Means ("SimpleKMeans.java").

- O pacote "**core**" não contém algoritmos que executam tarefas de mineração de dados, mas sim as classes que formam o núcleo da Weka. Por exemplo, as classes para importação e manipulação de bases de dados ("DataSource", "Instances", "Instance", "Attributes", etc.).

É importante que você saiba que existem outras formas de utilizar o código-fonte da Weka em seus projetos. Existe, por exemplo, a possibilidade de obter o código a partir de um repositório do Subversion. Neste caso, as instruções de utilização encontram-se apresentadas no link [6].

Um Exemplo Prático

Esta seção apresenta um exemplo prático de utilização da Weka API em um programa Java. Embora muito simples, o programa mostra como podemos realizar diferentes etapas da **tarefa de classificação**, uma das mais utilizadas na mineração de dados. Desta forma, o exemplo possui a capacidade de ser facilmente adaptado para resolver um problema real de mineração de dados.

A tarefa de classificação consiste em realizar a associação automática de um objeto a uma determinada classe, pertencente a um conjunto pré-definido de classes. De uma maneira simplificada, a execução do processo é composta pelas etapas descritas a seguir:

- **Treinamento** (Etapa 1): o algoritmo de classificação processa uma **base de treinamento**, composta por instâncias já classificadas. Como resultado, este algoritmo constrói (ou "aprende") um modelo capaz de classificar novos objetos (instâncias cuja classe é desconhecida).

- **Teste** (Etapa 2): após o treinamento, o modelo precisa ser testado para que sua acurácia seja avaliada. Isso pode ser feito com o uso de uma base de teste, que também possui instâncias classificadas. A acurácia do modelo de classificação representa a porcentagem de instâncias do conjunto de teste que são corretamente classificadas pelo modelo construído na fase de treinamento. Se a acurácia for alta, o classificador é considerado eficiente, podendo então ser colocado em produção (ou seja, poderá ser usado para classificar novos objetos).

Apresentaremos um exemplo de uso da Weka API que realiza a classificação com o uso da classe "IBk.java" (uma implementação simples do algoritmo k-NN). Para que o exemplo ficasse bem simples, optamos pela utilização da base de dados "weather.arff" que é uma das bases-exemplo localizada na pasta "data" da ferramenta Weka. Esta base é a mesma que foi utilizada no exemplo da Listagem 1. O seu conteúdo completo encontra-se apresentado na Figura 2.

A base de dados "weather.arff" possui 14 instâncias que indicam condições climáticas em diferentes dias. A partir desta base, torna-se possível minerar um classificador para decidir se é possível praticar um esporte em um local aberto (ex: jogar golfe) em função das condições climáticas de um determinado dia (ex: se está chovendo, ventando, etc.). Os atributos da base são: (i) *outlook* que indica se o tempo está ensolarado (*sunny*), nublado (*overcast*) ou chuvoso (*rainy*); (ii) *temperature* (valor numérico que indica a temperatura em fahrenheit); (iii) *humidity* (valor numérico, indicando a umidade); (iv) *windy* (valor booleano, indicando se está ou não ventando); (v) *play (*atributo classe, que indica se no dia em questão houve ou não jogo de golfe).

A Listagem 2, apresenta o programa-exemplo escrito em Java que realiza a classificação da base "weather.arff" com o uso Weka API. Conforme citado anteriormente, o algoritmo de classificação escolhido foi o IBk ("IBk.java"). A explicação detalhada do programa é apresentada em seguida ao código.

```
import weka.core.converters.ConverterUtils.DataSource;
import weka.core.Instances;
import weka.core.Instance;
import weka.classifiers.lazy.IBk;

public class TesteIBk {

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

      //-------------------------------------------------------
       // (1) importação da base de dados de treinamento
      //-------------------------------------------------------
         DataSource source = new DataSource("c:\\bases\\weather.arff");
         Instances D = source.getDataSet();

         // 1.1 - espeficicação do atributo classe
         if (D.classIndex() == -1) {
             D.setClassIndex(D.numAttributes() - 1);
         }

      //-------------------------------------------------------
       // (2) Construção do modelo classificador (treinamento)
      //-------------------------------------------------------
         IBk k3 = new IBk(3);
         k3.buildClassifier(D);

      //-------------------------------------------------------
       // (3) Classificando um novo exemplo
      //-------------------------------------------------------

         // 3.1 criação de uma nova instância
         Instance newInst = new Instance(5);
         newInst.setDataset(D);

         newInst.setValue(0, "sunny");
         newInst.setValue(1, 80);
         newInst.setValue(2, 75);
         newInst.setValue(3, "TRUE");

         // 3.2 classificação de uma nova instância
         double pred = k3.classifyInstance(newInst);

         // 3.3 imprime o valor de pred
         System.out.println("Predição: " + pred);

    }
}
```

Listagem 2. TesteIBk.java

Vamos à explicação do código do programa. Inicialmente, observe que o programa possui 3 seções distintas, cada qual exemplificando uma parte específica do processo de

classificação:

1. Importação da base de dados de treinamento.
2. Construção do modelo classificador (treinamento)
3. Classificação de um novo exemplo

Seção 1: Importação da Base de Dados de Treinamento (Linhas 10-19)

Esta primeira parte do programa, contém o método básico para realizar a importação da bases de dados que será alvo do processo de mineração de dados.

O primeiro passo consiste em utilizar a *helper class* **DataSource** para indicar a base de dados. Veja que, em nosso exemplo, estamos supondo que a base de dados está localizada no caminho "C:\bases\weather.arff" (modifique o caminho em seu programa, para apontar para o local certo de "weather.arff" em seu computador). Embora a Weka trabalhe prioritariamente com arquivos ARFF, ela também é suporta outros formatos, como CSV, XRFF (uma espécie de versão XML do formato ARFF), além de possibilitar a importação de dados provenientes de tabelas armazenadas em bancos de dados relacionais (via JDBC). A classe "DataSource" é responsável por examinar o formato da base de dados de entrada e utilizar o conversor adequado para possibilitar a importação da mesma para a memória.

Em seguida está a linha "Instances D = source.getDataSet();". Ela cria um objeto da classe **Instances** que, basicamente, gerencia o conjunto de instâncias da base de dados que foi importado para a memória. "Instances" é uma das classes mais importantes do núcleo da Weka. Ela possui diversos métodos onde você pode recuperar qualquer informação sobre a base de dados ou, até mesmo, alterar o seu conteúdo. Consulte o JavaDoc da Weka API para maiores informações [7].

Finalizando a seção de código 1, utilizamos o método "**setClassIndex**" de Instances para definirmos qual é o **atributo classe** em nossa base de dados. Quando uma base é importada para a memória, cada atributo recebe um índice de acordo com a ordem em que foi especificado no arquivo ARFF. Em nosso exemplo os índices são 0 – Outlook, 1 – Temperatura, 2 – Humidity, 3 – Windy e 4 – Play. O código das linhas 17 a 19 define que Play é o atributo classe indicando exatamente o seu índice (recorde que o método "numAttributes", mostrado no programa da Listagem 1, retorna o número de atributos da base de dados).

Seção 2: Criação do Modelo Classificador (Treinamento)

Esta segunda seção é incrivelmente simples, consistindo apenas em duas linhas de código.

Inicialmente, instanciamos um objeto chamado k3 do tipo "IBk". Isto significa criar uma instância do algoritimo classificador IBk em memória (o algoritmo que será usado para treinar o modelo classificador). É importante chamar a atenção para dois aspectos. O primeiro, é que, para utilizar o IBk, torna-se preciso preliminarmente fazer o **import** da respectiva classe para o seu programa (veja a linha 4, onde importa-se o IBk que está dentro do pacote "weka.classifiers.lazy"). Se você for utilizar outro classificador, como o Naïve Bayes, SVM, ID3, J48, etc., deverá fazer a importação do pacote adequado. O

segundo aspecto importante é que o IBk foi instanciado com o parâmetro 3 (valor entre parênteses). O que isto significa? O IBk nada mais é do que uma implementação do algoritmo k-NN que realiza a classificação de um objeto em função de seus k vizinhos mais próximos. Quando estudamos mineração de dados, aprendemos que este algoritmo requer que o valor de k seja especificado como entrada pelo usuário. Em nosso exemplo, simplesmente indicamos o valor de k como 3, pois o IBk possui um construtor que aceita receber este parâmetro. Mas como saber que um determinado algoritmo da Weka requer um parâmetro de entrada? Existem duas formas: você pode consultar o JavaDoc da Weka API [7] ou simplesmente entrar no código da classe para examiná-lo (ex: "fuçar" o código da classe "IBk.java").

Após instanciarmos o objeto do tipo IBk com o parâmetro adequado, podemos utilizá-lo para criar o modelo classificador. Para isto, basta fazer uma chamada ao método "**buildClassifier**", passando como entrada a base de dados de treinamento. Por padrão, os algoritmos de classificação da Weka sempre possuem um método chamado "buildClassifier" que tem exatamente a tarefa de realizar a construção do modelo classificador (ou seja, treinar o classificador). Isto que dizer que se, por exemplo, você decidir usar o Naïve Bayes, o J48, ou outro algoritmo de classificação ao invés do k-NN, a forma de treinar o classificador será a mesma: fazendo uma chamada ao método "buildClassifier".

Para que o exemplo ficasse simples, optamos por não implementar a etapa de teste do modelo classificador. No entanto, isto também pode ser feito de maneira simples na Weka. Consulte o link [8] para obter um exemplo.

Seção 3: Classificando um Novo Exemplo

Agora que o nosso modelo está construído, podemos utilizá-lo para classificar um novo exemplo. É exatamente isto que é feito na seção 3 de nosso código-exemplo.

Na primeira parte (3.1), instanciamos um objeto do tipo **Instance** chamado "newInst". "Instance" é a classe da Weka utilizada para a representação de **uma instância** de uma base de dados (ou seja um elemento de Instances). No momento que o objeto "newInst" é instanciado, utilizamos o parâmetro 5 entre parênteses para indicar que esse objeto possuirá 5 atributos (nesse momento ainda não se sabe que atributos serão esses).

Depois disso, a linha "newInst.setDataset(D);" realiza um processamento bastante interessante. Ela indica que a instância "newInst" possui a mesma estrutura das instâncias que compõem o objeto "D" (a nossa base de treinamento em memória, um objeto do tipo Instances). **Importante**: essa linha de código não insere "newInst" em "D" !!! Ela serve apenas para dizer que "newInst" tem especificação idêntica a das instâncias de "D".

Estando a especificação de "newInst" definida, podemos preenchê-la com os valores desejados. Em nosso exemplo, atribuirmos o valor "sunny" para o atributo "Outlook" (índice 0), o valor 80 para o atributo "Temperature" (índice 1), 75 para "Humidity" (índice 2) e "TRUE" para "Windy" (índice 3). Em função desse valores, qual será o valor que o modelo de classificação atribuirá para a variável classe "Play"?

Para obtermos a resposta a esta pergunta, basta chamar o método responsável por realizar a classificação: "**classifyInstance**" (seção 3.2 do código da Listagem 2). É preciso

passar o como entrada o objeto cuja classe queremos prever ("newInst"). Observe que, no exemplo, valor retornado é armazenado em uma variável chamada "pred" que é do tipo **double**. Isso parece estranho, tendo em vista que o atributo classe "Play" é do tipo discreto, possuindo os valores "yes" e "no" (veja a Figura 7). Então por que "pred" é do tipo double? A resposta encontra-se no fato de que, **internamente**, a Weka armazena os valores de todas as variáveis como um double. Sendo assim, para o atributo classe "Play", o valor "yes" é representado internamente como 0.0 e o valor "no" é representado internamente como 1.0 (o valor interno de armazenamento segue a ordem em que os valores de "Play" foram especificados no cabeçalho do arquivo ARFF: "yes" antes do "no").

Se você executar o programa, verá que o valor de pred será impresso como 1.0 (seção 3.3 do código-exemplo). Isso significa que o modelo classificador previu play = "no" para a nova instância "newPred" (Figura 13).

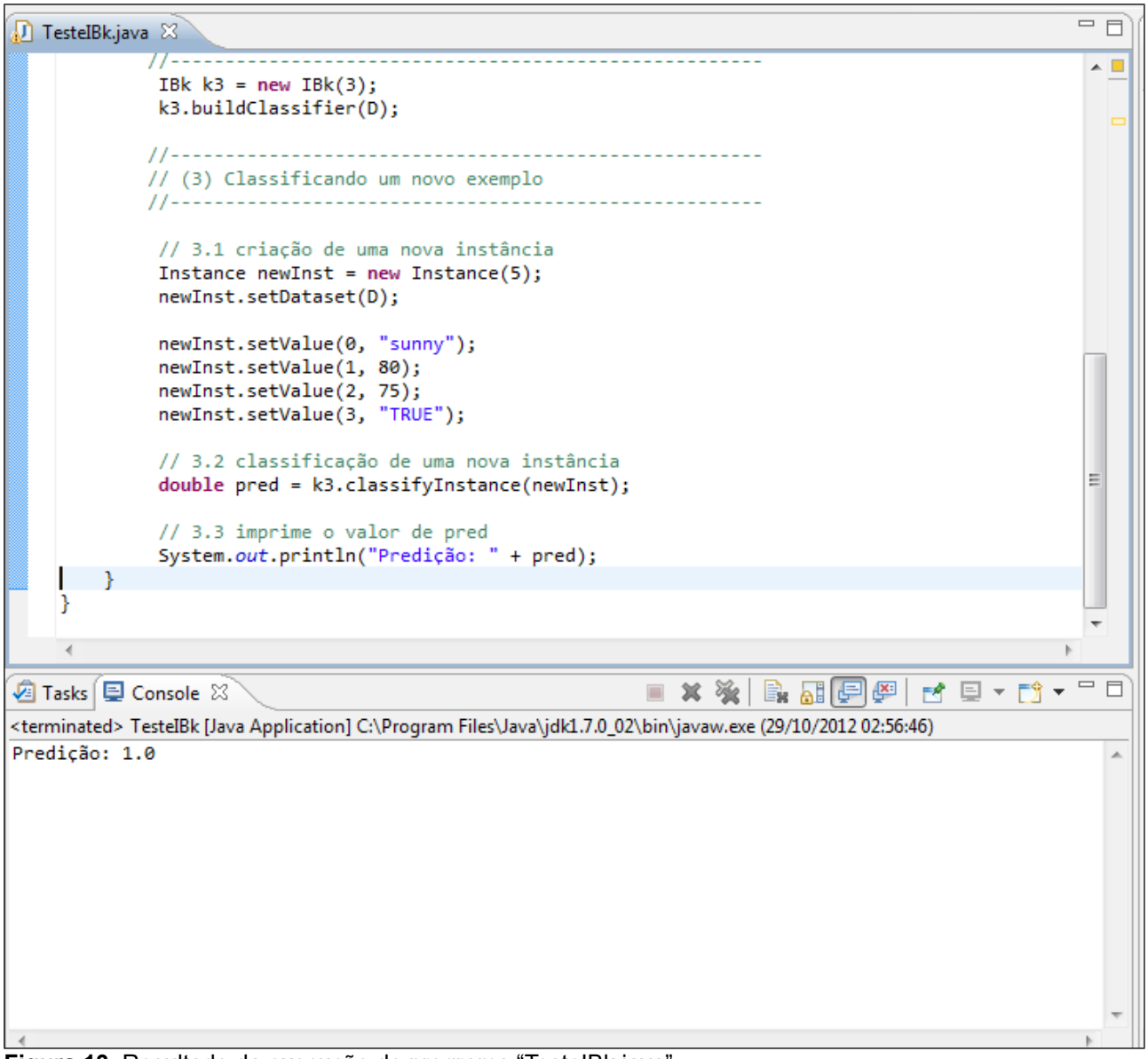

Figura 13. Resultado da execução do programa "TesteIBk.java"

Seria possível imprimir o valor “no” ao invés do número 1.0? Sim, é possível. Você pode ter certeza de que a Weka API é muito flexível e normalmente fornecerá quase tudo que você puder imaginar em matéria de recursos. Se você desejar ver o valor da classe como uma String, bastará acrescentar as linhas de código apresentadas na Listagem 3 no final do programa.

```
// 3.4 imprime a string correspondente ao valor de pred
Attribute a = D.attribute(4);
String predClass = a.value((int) pred);
System.out.println("Predição: " + predClass);
```

Listagem 3. Imprimindo a classe prevista como uma String

Acrescente também o seguinte import no início de seu programa:

```
import weka.core.Attribute;
```

Explicação: na linha “Attribute a = D.attribute(4)” estamos recuperando as informações do quarto atributo da base de dados de treinamento D (do tipo Instances) e armazenando-as na variável “a” do tipo “**Attribute**”. Como sabemos, este quarto atributo é “Play”. “Attribute” é uma importante classe core da Weka utilizada para manter informações a respeito dos atributos de uma base de dados.

A linha “String predClass = a.value((int) pred);” utiliza o método “**value**” de “Attribute” para retornar o valor String correspondente ao valor interno armazenado em pred (1.0 = “no”) e armazená-lo na variável “predClass”. A seguir, basta imprimir o valor de predClass. Veja o resultado na Figura 14.

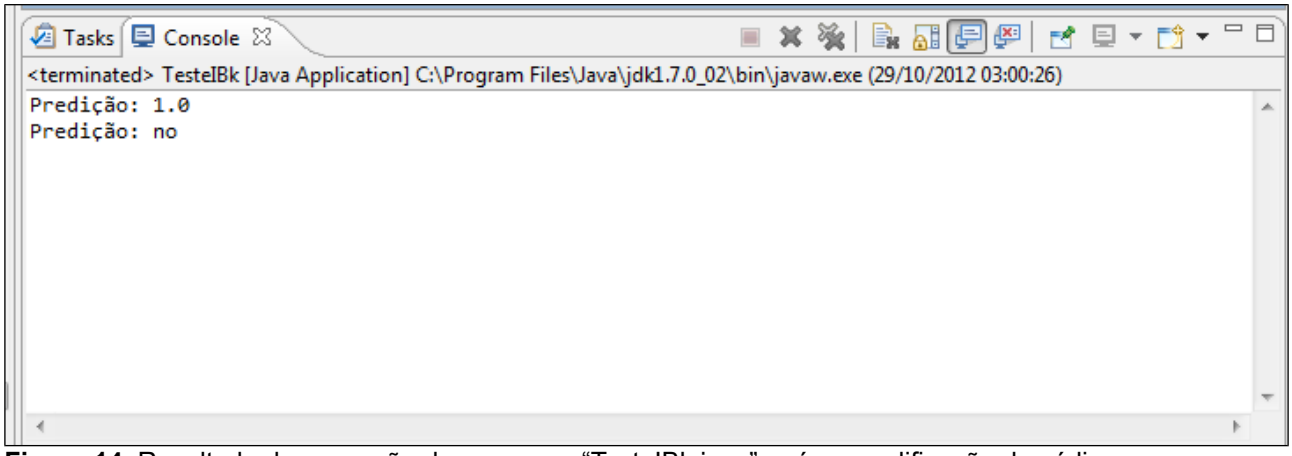

Figura 14. Resultado da execução do programa “TesteIBk.java” após a modificação do código.

Conclusões

Este artigo abordou a utilização da Weka API em programas Java para possibilitar a criação de programas capazes de lidar com projetos de mineração de dados reais. Inicialmente, foi mostrada a forma de obter a API e incorporá-la ao ambiente de desenvolvimento Java. A seguir, foi apresentado um pequeno programa Java que demonstrou como utilizar a API em um projeto prático. Neste projeto, um modelo de classificação foi criado com o uso do algoritmo k-NN e, a seguir, utilizado para classificar um novo objeto.

Embora o artigo tenha apresentado um programa muito simples, é importante deixar claro que o modelo básico para utilizar a Weka dentro de qualquer sistema será sempre similar. Primeiro será preciso importar a base de dados, depois criar o modelo de mineração de dados e depois utilizá-lo sobre novos objetos. Evidentemente, em um projeto real você precisará lidar com algumas questões adicionais, tais como: executar a fase de teste do modelo de classificação (por exemplo, usando um esquema de cross-validation) ou aplicar o modelo sobre diversos novos objetos e não apenas um (esses novos objetos podem estar armazenados em um arquivo ou em uma tabela de banco de dados).

Na verdade, essas questões podem ser resolvidas de forma simples através da Weka API, uma vez que esta é extremamente flexível e poderosa. Este artigo descreveu apenas os primeiros passos para quem desejar utilizar a API. O objetivo principal do artigo foi o de demonstrar que ela pode, de forma simples e prática, ser utilizada em projetos reais de mineração de dados (e não apenas para fins didáticos ou para execução de avaliações em ambiente acadêmico). O leitor que desejar se aprofundar no uso da Weka API poderá consultar os links [8], [9] e [10], que abordam temas importantes, como a implementação da fase de teste em processos de classificação (cross-validation, uso de arquivos de treino e teste, etc.), execução da tarefa de clusterização, acesso à bases de dados relacionais, criação de modelos de classificação utilizando outros algoritmos além do k-NN (Naïve Bayes, SVM, J48, etc.), passagem de parâmetros para os algoritmos, entre outros. Outra referência importantíssima é o JavaDoc da Weka API, disponível no link [7].

Referências Bibliográficas

[1] I. Witten, E. Frank, and H. Mark, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, 2011.

[2] E. C. Gonçalves, “Data Mining com a Ferramenta Weka.” [Online]. Available: http://forumsoftwarelivre.com.br/2011/arquivos/palestras/DataMining__Weka.pdf.

[3] E. C. Gonçalves, “Mineração de Regras de Associação com a Ferramenta de Data Mining Weka,” *DevMedia*, 2011. [Online]. Available: http://www.devmedia.com.br/mineracao-de-regras-de-associacao-com-a-ferramenta-de-data-mining-weka/20478.

[4] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse, *WEKA Manual for Version 3-7-6*. University of Waikato, 2012.

[5] “Eclipse.” [Online]. Available: www.eclipse.org.

[6] “Subversion (Weka Wikispaces).” [Online]. Available: http://weka.wikispaces.com/Subversion.

[7] “Weka API - Documentação (JavaDoc).” [Online]. Available: http://weka.sourceforge.net/doc.stable/.

[8] “Use WEKA in your Java code.” [Online]. Available: http://weka.wikispaces.com/Use+WEKA+in+your+Java+code.

[9] “Weka - Programmatic Use.” [Online]. Available: http://weka.wikispaces.com/Programmatic+Use.

[10] “Weka - Source Code.” [Online]. Available: http://weka.wikispaces.com/space/content?tag=source%20code.