
Weka

Fernando Anselmo

<http://fernandoanselmo.orgfree.com/wordpress/>

Versão 1.0 em 18 de julho de 2020

Resumo

Weka é um projeto com o objetivo de disseminar as técnicas de *Machine Learning* através da disponibilização de um software para utilização de pesquisadores, alunos. Sem utilizar qualquer linguagem de programação é possível resolver problemas reais e isso pode auxiliar ao estudante iniciante em Ciência de Dados a compreender melhor como funciona esse novo mundo que se abre. *Data Mining* não precisa ser um domínio exclusivo de grandes empresas através de um software com custos exorbitantes e tem o objetivo de facilitar o trabalho no tratamento de dados com *Big Data* através de ferramentas para *pre-processing*, *classification*, *regression*, *clustering*, regras de associação dos atributos e visualização.

1 Parte inicial

Criado pela Universidade de **Waikato** (Nova Zelândia) e implementado pela primeira vez em 1997. Utiliza a *GNU General Public License* (GPL). Foi reescrito da linguagem C para Java como forma de permitir uma melhor portabilidade e uma GUI (*Graphic Universal Interface*) para interagir com arquivos de dados e produzir resultados visuais. Permite pré-processar o *Big Data*, aplicar diferentes algoritmos de *Machine Learning* e comparar as várias saídas através de relatórios estatísticos e de modo gráfico.

O tipo de algoritmo que aplicamos tem por base principalmente um conhecimento de domínio dentro de um tipo, por exemplo, **classificação**, existem diversos algoritmos disponíveis. Podemos testar diferentes tipos para criar um modelo eficiente de *Machine Learning*. E ao fazer isso, normalmente, preferimos visualizar os dados processados de forma gráfica e, portanto, também precisamos de ferramentas para visualização.



Figura 1: Logo do Weka

Weka é um software de código aberto e fornece várias ferramentas para o pré-processamento de dados, implementação de vários algoritmos de *Machine Learning* e ferramentas de visualização de

modo que podemos aplicar as técnicas de aprendizado de máquina a problemas reais de mineração de dados. Como o aplicativo é totalmente desenvolvido usando a linguagem de programação Java. Permite o acesso a vários SGBD, através do uso de *drivers Java Database Connectivity* (JDBC), que podem ser utilizados como fonte de dados. Contém ainda algoritmos incrementais que usamos para processar grandes conjuntos de dados.

Fornece implementações de *Machine Learning* que podemos aplicar facilmente em qualquer conjunto de dados. Também inclui uma variedade de ferramentas para transformação, como algoritmos para amostragem, alimentar um esquema de aprendizado e analisar o classificador resultante de seu desempenho - tudo sem escrever nenhum código.

Além disso tudo, por ser criado utilizando *Open Source* a comunidade Weka é muito ativa e constantemente produz *plugins* (chamados de pacotes) que podem ser incorporados e fornece funcionalidades não previstas originalmente.

2 Dentro do Programa

O processo de instalação do Weka é simplesmente um arquivo compactado que após descompactar (trataremos essa pasta criada pelo arquivo de **WEKA_HOME**), na raiz da pasta existe o arquivo **weka.sh** (no Windows esse arquivo é um **.bat**) que é utilizado para executar o programa.



Figura 2: Tela Principal do Weka

As aplicações disponíveis são:

- **Explorer:** é a opção mais utilizada, com a possibilidade de trabalhar de forma interativa. É onde iremos explorar as funções da ferramenta. Focaremos principalmente nessa, pois nas outras opções realizamos muitas das funções contidas aqui, porém de forma automatizada. Esta opção permite importar dados, aplicar algoritmos de filtragem para transformar as características quantitativas em discretas ou excluir de acordo com critérios definidos.
- **Experimenter:** automatizar os processos, por exemplo, para rodar vários algoritmos de *Machine Learning* de forma automática e visualizar todos os resultados de uma só vez.
- **KnowledgeFlow:** criar fluxos de processos através de uma interface com base em fluxo de dados. É possível arrastar componentes como filtros ou classificadores e conecta-los para obter novos fluxos de processamento.
- **Workbench:** combinar todas as outras aplicações mostradas em uma única interface. Essa pode ser interessante caso tenhamos que alternar entre as opções *Explorer* e *Experimenter* por

várias vezes.

- **Simple CLI:** uma interface de linha de comando (*shell*) destinada a criação de *scripts shell* que podem ser utilizados com a API do Weka. Com esta API é possível rodar experimentos, algoritmos de *Machine Learning* e diversas outras tarefas. Podemos automatizar tarefas e integrar aplicações.

ATENÇÃO: Pasta de Trabalho do Usuário

Quando executamos o Weka pela primeira vez, no diretório raiz do usuário é criada uma pasta chamada `/wekafiles`. Nesta pasta podemos colocar os arquivos **.props** para conexão com banco de dados SQL.

3 Análise Exploratória dos Dados

Antes de partirmos para os algoritmos propriamente ditos, primeiro devemos abrir e entender um conjunto de dados. Em estatística, *Exploratory Data Analysis* é uma abordagem sobre um determinado conjunto de dados para compreendermos suas características principais. Clicar em **Explorer**:

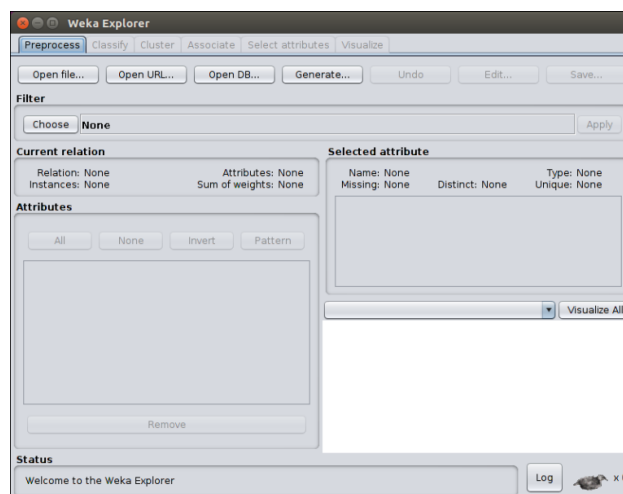


Figura 3: Opção Explorer

Abaixo das guias temos os seguintes botões para a leitura de dados:

- **Open file...** por padrão o Weka lê arquivos em formato ARFF (*Attribute-Relation File Format*), o programa traz como exemplo algumas bases bem conhecidas.
- **Open URL...** no qual podemos obter os dados diretamente da Web.
- **Open DB...** pela disponibilização de um driver JDBC podemos ler qualquer SGBD relacional.
- **Generate...** gerar aleatoriamente uma base de dados, útil para procedermos testes com os algoritmos.

Como exemplo clicamos em **Open file...** e selecionar **WEKA_HOME/data**:

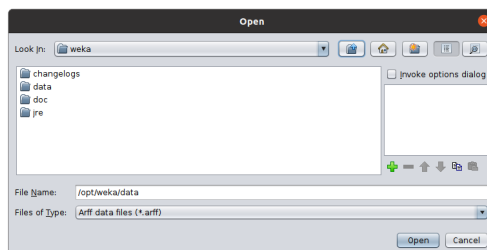


Figura 4: Pastas contidas no Weka

Selecionar o dataset **iris.arff** do diretório **data** e pressionar **Open**:

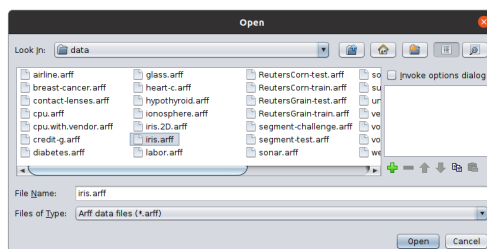


Figura 5: Selecionar o DataSet

ATENÇÃO: Não encontrou os arquivos

Não se preocupe, na página <https://waikato.github.io/weka-wiki/datasets/> existem vários exemplos e nesta outra <https://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html> os mesmos arquivos ARFF disponibilizados. Criar uma pasta /data (no seu diretório de trabalho) e baixar os arquivos para esta.

Após carregar o *DataSet* na tela teremos uma visão geral na tela principal, são 5 atributos (colunas) e 150 instâncias (linhas). As classes ou rótulos são os tipos reais das flores. São quatro atributos preditores: *SepalLength*, *SepalWidth*, *PetalLength*, *PetalWidth* e um atributo alvo *class* que identifica a espécie da Flor Iris (*Setosa*, *Versicolor* e *Virginica*). Essa classificação foi feita por pessoas especialistas no assunto:

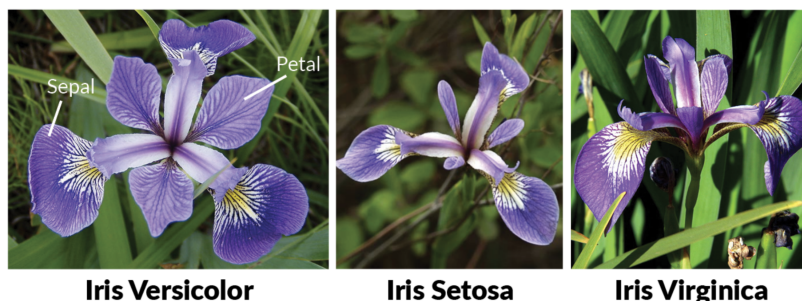


Figura 6: Tipos de Flores Iris

No botão **Edit...** podemos visualizar e modificar os dados. Como por exemplo, excluir uma coluna, ou modelar em um formato diferente. Neste momento vamos deixar como está. O objetivo aqui

é usar esse *DataSet* para aplicar os algoritmos de *Machine Learning* que seja capaz de classificar corretamente a espécie de alguma flor com base nos atributos preditivos.

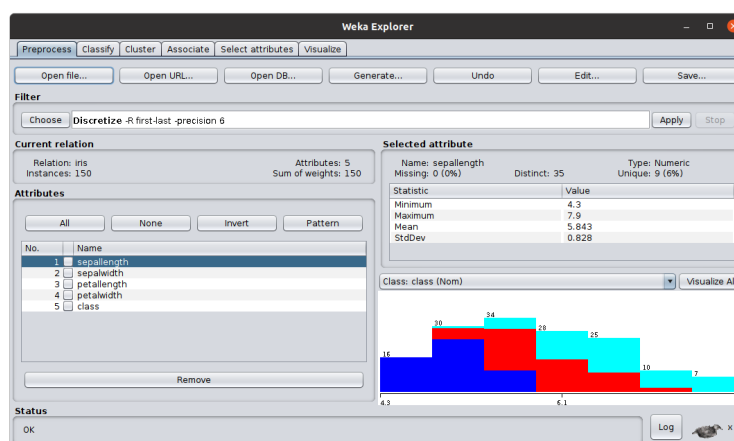


Figura 7: Tela Principal com o DataSet Iris carregado

Essa janela é excelente para verificar os dados que temos a disposição uma vez que estão carregados, seus atributos são mostrados no box **Attributes**. Acima temos **Current Relation** que exibe o nome da relação (tabela), número de instâncias, de atributos e seus pesos. Durante a varredura dos dados, é calculado algumas estatísticas básicas para cada atributo selecionado mostradas no box **Selected attribute**.

As ferramentas de pré-processamento no WEKA são chamadas de *Filters*. São filtros para discretização, normalização, re-amostragem, transformação seleção e combinação de atributos. Algumas técnicas, como a mineração de regras de associação, só podem ser executadas em dados categóricos. Isso requer a discretização de atributos numéricos ou contínuos. Para aplicar esse filtro pressionar o botão **Choose** que mostra uma lista de filtros disponíveis. Selecionar **Supervised** ▷ **Attribute** ▷ **Discretize** e pressionar o botão **Apply**. O filtro converterá os valores numéricos em nominal.

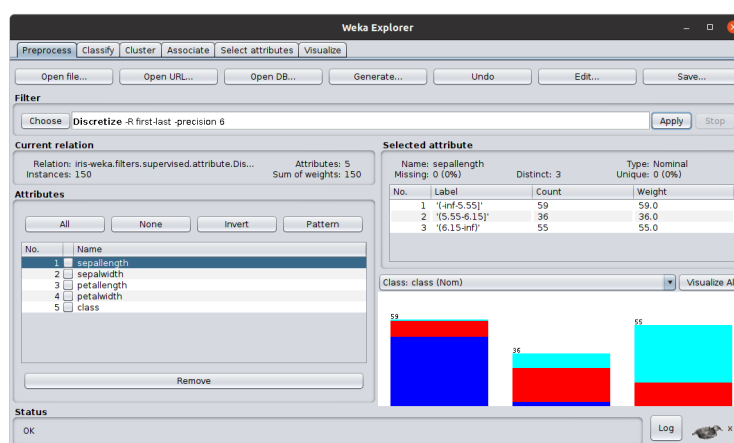


Figura 8: Discretização Aplicada

Na parte de baixo temos uma distribuição dos dados para cada atributo selecionado ou comparação unitária quando selecionamos 2 ou mais. Podemos também visualizar todas as comparações com o botão **Visualize All**.

Porém a melhor maneira de visualizar esses relacionamentos e na guia superior **Visualize**.

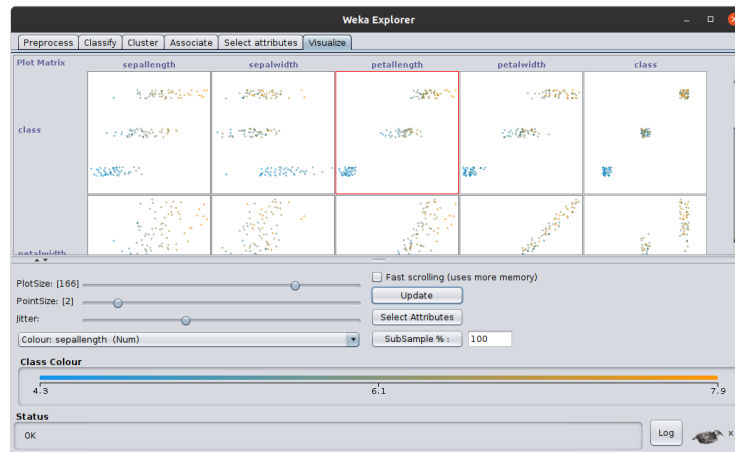


Figura 9: *Relacionamento entre os atributos*

A visualização mostra um gráfico da relação entre os atributos e ajuda a determinar um problema quanto a dificuldade de aprendizagem. O botão **Select Attributes** é ideal para limitarmos a quantidade de atributos que desejamos visualizar e o botão **SubSample %** permite definir a quantidade de dados a visualizar.

Antes de sairmos para testar os modelos, devemos conhecer bem estas janelas para explorar melhor os dados e compreendermos o que temos a nossa disposição pois esta é uma fase extremamente importante e muitas vezes negligenciada.

4 Executar um Algoritmo

No Weka os algoritmos de *Machine Learning* estão concentrados nas seguintes guias:

- **Classify:** os classificadores aprendem a atribuir um rótulo de classe através dos dados que foram fornecidos como exemplos. Um classificador pode reconhecer e-mails como "spam" ou "não spam". Temos a nossa disposição Regressão Linear e Logística, *Support Vector Machines*, Árvore de Decisão, *RandomForest* e *Naive Bayes*.
- **Cluster:** a clusterização é a separação dos dados em categorias de similaridades, são disponibilizados algoritmos como: *SimpleKMeans*, *FilteredClusterer* e *HierarchicalClusterer*.
- **Associate:** o aprendizado por regras de associação é um método utilizado para descobrir relações interessantes entre variáveis em grandes bancos de dados. Destina-se a identificar regras fortes descobertas em bancos de dados e se utiliza de algumas medidas de interesse. Nesta categoria temos: *Apriori*, *FilteredAssociator* e *FPGrowth*.

O modo de executar qualquer algoritmo (Classificação, Clusterização ou Associação) é exatamente o mesmo, assim não ficaremos nesta tratando das diversas opções existentes e ao entender um único conjunto seremos capazes de realizar o mesmo processo para outros algoritmos.

Árvore de Decisão será nossa porta de entrada para esse mundo, esta é uma representação de uma tabela de decisão sob a forma de uma árvore. Para executar este com o Weka, clicar na aba **Classify**, pressionar **choose**, navegar até **trees** e selecionar o algoritmo **J48**:

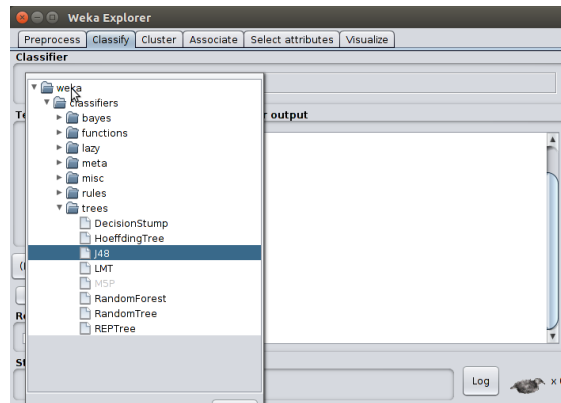


Figura 10: Visão do J48

J48 é uma implementação do algoritmo C4.5 na linguagem Java que utiliza uma árvore de decisão construída a partir dos dados de treino. Essas árvores são hipóteses usadas para prever probabilidades de uma instância ser de um determinado tipo. Cada algoritmo possui uma descrição de como trabalham, para vê-la, posicione o mouse no nome deste e uma janela amarela será aberta.

Na aba **Test Options** está selecionado a opção **Cross Validation** com **10 folds**. Isso significa que o *DataSet* será treinado 9 vezes com partes distintas (*folds*) dos dados. Uma parte do dado será usada como teste para classificação enquanto outras 9 partes serão usadas para treinar o modelo. Dessa forma o algoritmo é testado com todas as partes dos dados evitando erros de variância. Clicar em **Start** para executar o algoritmo.

As principais opções de teste disponíveis são:

- *Use training set*. Avalia o classificador em relação ao quão bem prediz a classe das instâncias em que foi treinado.
- *Supplied test set*. Permite adicionar o um conjunto de testes independente. Clicar no botão **Set** para adicionar o arquivo.
- *Cross-validation*. Avalia o classificador por validação cruzada, usando o número de dobras inseridas no campo **Folds**.
- *Percentage split*. Selecionar uma certa porcentagem dos dados, que é mantida para o teste. A quantidade de dados retidos depende do valor digitado no campo **%**.

Existem outras opções que podem ser acessadas através do botão **More options...**. Para esses dados provavelmente o resultado aparece em segundos, obviamente que se usamos uma base muito grande o tempo de espera será maior.

Na parte inferior da janela temos a linha de status do que aconteceu. À direita temos o ícone do Weka. O número ao lado indica os vários processos que estão em execução simultaneamente. Por exemplo, ao carregar um arquivo, o pássaro se senta, isso significa que não há processos em execução (o sistema está ocioso). Ao rodar um algoritmo observe o pássaro, ele se levanta e começa a se mover, isso indica que um processo foi iniciado.

4.1 Resultados dos Classificadores

Após pressionarmos o botão **Start**, o algoritmo inicia sua tarefa, após a conclusão sua tarefa esta é marcada na **Result list** de modo que podemos alternar entre os vários relatórios armazenados. Em **Classifier output** mostra o relatório do resultado:

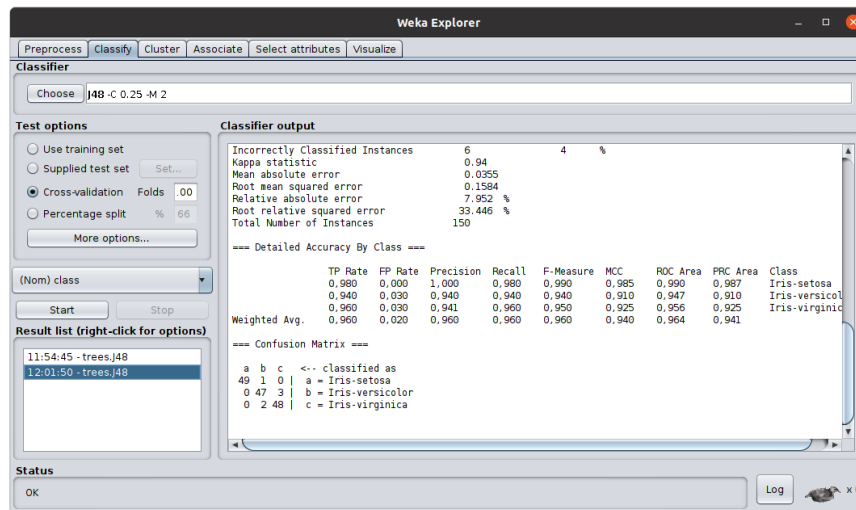


Figura 11: Relatório de Resultados

Na seção **Summary**, verificamos que das 150 instâncias que foram testadas, o algoritmo acertou 144 (*Correctly Classified Instances*). Isso significa que 96% das instâncias testadas foram corretamente classificadas. Esta métrica é chamada de **Acurácia**, ou seja, nosso algoritmo teve 96% de acertos. Na seção **Detailed Accuracy By Class** temos a acurácia por classe.

Na seção **Confusion Matrix** é possível investigar como se saiu a classificação de cada classe comparada aos valores reais:

- Classe **A** (*Iris-setosa*) foi classificada 1 vez como se fosse Classe **B** (*Iris-versicolor*).
- Classe **B** foi classificada 3 vezes como se fosse classe **C** (*Iris-virginica*).
- Classe **C** foi classificada 2 vezes como se fosse classe **B**.

Essa matriz permite visualizar quais classes estão sendo classificadas erroneamente, ou seja, aonde o modelo está errando e precisa de mais dados para ser melhor treinado.

Escolher outros classificadores e veja o maior número de resultados possíveis, comparar os modelos, permita-se explorar bem esta tela.

ATENÇÃO: Conversão de CSV para ARFF

Podemos converter facilmente um arquivo CSV para o formato nativo do Weka chamado ARFF, basta executar o seguinte comando:

```
java -cp weka.jar weka.core.converters.CSVLoader arq.csv > arq.arff
```

Weka fornece acesso direto à biblioteca de algoritmos implementados. Esse recurso possibilita a aplicação de algoritmos criados em diferentes sistemas. Por exemplo, podem ser facilmente trazidos do MATLAB e a ferramenta para acessar os algoritmos é implementada em pacotes de *Machine Learning* como **Spider** e **MATLABArsenal**.

Um modelo utiliza um conjunto de dados para treinar através de cálculos a melhor maneira de mapear exemplos de dados de entrada para o resultado. Como tal, o conjunto de dados de treinamento deve ser suficientemente representativo do problema e ter muitos exemplos consistentes do atributo alvo.

5 Gerenciador de Pacotes

Ao retornar para a tela principal do Weka (quando iniciamos o programa) na opção **Tools** > **Package Manager**, ganhamos acesso a todos os Classificadores que o Weka trabalha. Alguns desses foram criados e adicionados pela comunidade. O Gerenciador de Pacotes nos permite localizar e instalar o que desejamos trabalhar.

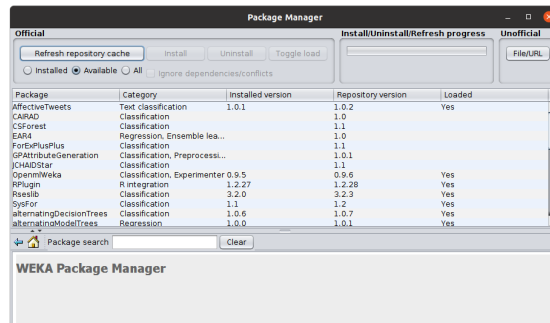


Figura 12: Gerenciador de Pacotes

Fornece uma lista de pacotes na parte superior da janela e um painel na parte inferior que exibe as informações sobre o pacote atualmente selecionado na lista. Podemos optar por exibir os pacotes disponíveis, apenas os instalados ou todos. A lista apresenta o nome de cada pacote, a categoria a que pertence, a versão atualmente instalada (se estiver), a versão mais recente do pacote disponível que é compatível com a versão do WEKA em uso e um campo que, para pacotes instalados, indica se o pacote foi carregado com sucesso pelo WEKA ou não.

Para atualizar qualquer pacote basta localizar um pacote aonde a coluna *Installed version* seja diferente de *Repository version*. Clicar neste pacote e pressionar o botão **Install**.

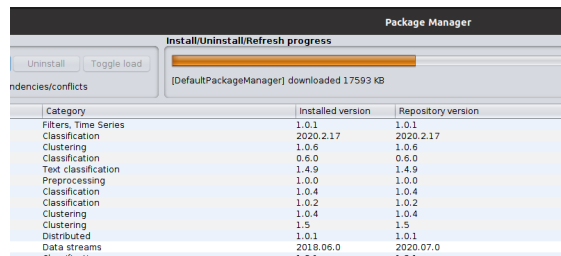


Figura 13: Atualizar o Pacote Data Streams

Por padrão, WEKA carrega todos os pacotes instalados, se um pacote não puder ser carregado por algum motivo, uma mensagem é exibida na coluna *Loaded*. Para impedir que um pacote específico seja carregado, basta selecioná-lo e clicar no botão **Toggle load**. O pacote ficará marcado para a próxima vez que o WEKA for iniciado. Isso pode ser útil se um pacote instável estiver com erros, em conflito com outro (talvez devido a bibliotecas de terceiros) ou impedindo o funcionamento adequado do WEKA.

6 Conectar a um banco MySQL

Podemos consultar dados diretamente através de diversos SGBDs por um driver JDBC (*Java Database Connectivity*). Isso facilita o acesso e a manipulação desses dados que estão armazenados em um banco de dados relacional como o MySQL, Oracle ou PostgreSQL.

Através de uma *Query* podemos importar os dados retornados para o Weka e trabalhar normalmente com todos os filtros, métodos e classificadores sem a necessidade de gerar um arquivo de dados.

ATENÇÃO: Nível Hard

O processo para obter acesso a base de dados SQL não é simples e exige que tenhamos a noção sobre como funciona a conexão JDBC e a configuração de variáveis e a URL de conexão. Nos *drivers* existe a documentação do seu funcionamento, então quando for baixá-los não menosprezar sua leitura.

Passo 1. Realizar o download do driver JDBC desejado para o SGBD e colocá-lo em uma pasta comum (por exemplo /libs).

Passo 2. Adicionar este arquivo (normalmente um **.jar**) a variável de ambiente CLASSPATH (no Linux isso está em um arquivo chamado /etc/environment). Por exemplo para o driver JDBC tipo 4 do SGBD MySQL:

```
CLASSPATH=".:/home/fernando/libs/mysql-connector-java-5.1.49.jar"
```

Passo 3. Criar um arquivo de propriedades **DatabaseUtils.prop** que define as diversas configurações específicas sobre o SGBD que iremos nos conectar. Por exemplo, este é para o SGBD MySQL:

```
1 jdbcDriver=com.mysql.jdbc.Driver
2 jdbcURL=jdbc:mysql://localhost:3306/conta?user=root&password=root
3
4 char=0
5 varchar=0
6 longvarchar=0
7 binary=0
8 varbinary=0
9 longvarbinary=0
10 bit=1
11 numeric=2
12 decimal=2
13 tinyint=3
14 mallint=4
15 INT=5
16 integer=5
17 bigint=6
18 real=7
19 float=2
20 double=2
21 date=8
22 time=10
23 timestamp=8
24
25 CHAR=0
26 TEXT=0
27 VARCHAR=0
```

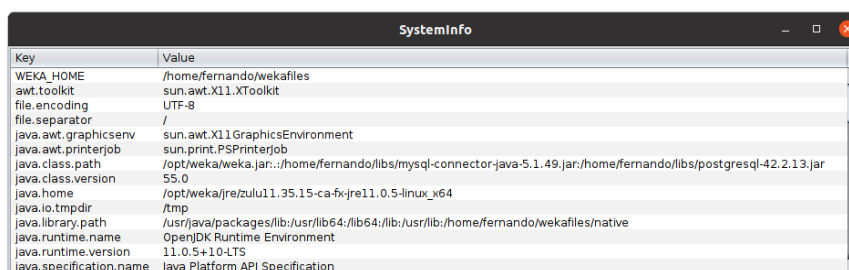
```

28 LONGVARCHAR=9
29 BINARY=0
30 VARBINARY=0
31 LONGVARBINARY=9
32 BIT=1
33 NUMERIC=2
34 DECIMAL=2
35 FLOAT=2
36 DOUBLE=2
37 TINYINT=3
38 SMALLINT=4
39 SHORT=4
40 SHORT=5
41 INTEGER=5
42 BIGINT=6
43 LONG=6
44 REAL=7
45 DATE=8
46 TIME=10
47 TIMESTAMP=8
48 DATETIME=8
49
50 # Tabela de Criacao
51 CREATE_STRING=TEXT
52 CREATE_INT=INT
53 CREATE_DOUBLE=DOUBLE
54 CREATE_DATE=DATETIME
55 DateFormat=yyyy-MM-dd HH:mm:ss
56
57 # Opcoes do Database
58 checkUpperCaseNames=false
59 checkLowerCaseNames=false
60 checkForTable=true
61 setAutoCommit=true
62 createIndex=false
63
64 # Palavras reservadas
65 Keywords=AND, ASC, BY, DESC, FROM, GROUP, INSERT, ORDER, SELECT, UPDATE, WHERE

```

A primeira linha está o driver de conexão com o banco e a segunda a URL padrão, de resto mostra apenas como o Weka deve converter os tipos de campos do MySQL para seus tipos. Visitar esse endereço para maiores referências: https://waikato.github.io/weka-wiki/weka_experiment_database_utils.props/.

Passo 4. Ao entrar no Weka a partir do Menu Principal acessar a opção **Help > SystemInfo**.



Key	Value
WEKA_HOME	/home/fernando/wekafiles
awt.toolkit	sun.awt.X11.XToolkit
file.encoding	UTF-8
file.separator	/
java.awt.graphicsenv	sun.awt.X11GraphicsEnvironment
java.awt.printerjob	sun.print.PSPrinterJob
java.class.path	/opt/weka/weka.jar:/home/fernando/libs/mysql-connector-java-5.1.49.jar:/home/fernando/libs/postgresql-42.2.13.jar
java.class.version	55.0
java.home	/opt/weka/jre/zulu11.35.15-ca-fx-jre11.0.5-linux_x64
java.io.tmpdir	/tmp
java.library.path	/usr/java/packages/lib:/usr/lib64:/lib64:/lib:/usr/lib:/home/fernando/wekafiles/native
java.runtime.name	OpenJDK Runtime Environment
java.runtime.version	11.0.5+10-LTS
java.specification.name	Java Platform API Specification

Figura 14: Janela *SystemInfo*

Nesta janela estão contidas todas as variáveis e valores de ambientes lidos pelo programa. Observar que a variável **java.class.path** contém o arquivo jar do Driver JDBC. Caso isso não aconteça verifique se a variável **CLASSPATH** está definida corretamente, outra opção pode ser editar o arquivo **weka.sh** (ou .bat) e verificar se está inserida corretamente:

```
CLASSPATH="$DIR/weka.jar:$CLASSPATH"
```

Passo 5. No Menu Principal acessar a opção **Tools** ▸ **SqlViewer**.

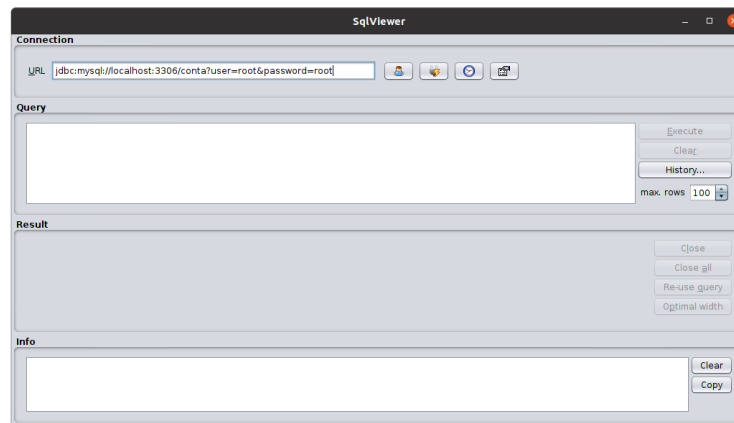


Figura 15: *Janela SqlViewer*

A URL que colocamos no arquivo PROPS já deve estar disponível, é possível trocá-la para acessar outra base de dados. Na frente temos uma série de botões: o primeiro permite alterar o usuário e senha (se for realizar tal ação remova essa propriedade da URL). O segundo conectar com o banco de dados. O terceiro trazer uma conexão anteriormente realizada (um histórico de conexões). O quarto ler um outro arquivo PROPS para a conexão com outro SGBD. Sendo assim podemos ter vários deles para os diferentes tipos de bases de dados.

Uma vez que tudo está sem problemas, podemos sair e entrar novamente na opção Explorer, porém ao invés de ler os dados por **Open File...** selecionamos a opção **Open DB...** e procedemos a consulta (*Query*) para selecionar os dados. Ao pressionar o botão OK estes serão trazidos para que possamos proceder qualquer análise.

7 Conclusão

Weka fornece a saída estatística do processamento do modelo, também uma ferramenta de visualização para inspecionar os dados. Na realidade, existe um software que faz as mesmas coisas que os programas caros — este software se chama Weka. Através de sua interface é possível incorporar ao Weka, como qualquer outra biblioteca, a seus próprios aplicativos para fazer coisas como tarefas de mineração de dados automatizadas no lado do servidor.

Weka é um aplicativo de código aberto disponível gratuitamente sob o contrato da GNU. Originalmente escrito em C e completamente reescrito em Java, tornou-se compatível com todos os sistemas operacionais e SGBD's mais conhecidos do mercado. É amigável com uma interface gráfica que permite rápida configuração e operação principalmente para os que não possuem nenhum conhecimento de linguagem de programação. Os vários modelos podem ser aplicados ao mesmo conjunto de dados. Podemos assim comparar as saídas e selecionar o melhor que atenda ao objetivo. Isso resulta em um desenvolvimento mais rápido de modelos para *Machine Learning* em geral.

Sou um entusiasta do mundo **Open Source** e novas tecnologias. Qual a diferença entre Livre e Open Source? Livre significa que esta apostila é gratuita e pode ser compartilhada a vontade. Open Source além de livre todos os arquivos que permitem a geração desta (chamados de arquivos fontes) devem ser disponibilizados para que qualquer pessoa possa modificar ao seu prazer, gerar novas, complementar ou fazer o que quiser. Os fontes da apostila (que foi produzida com o LaTeX) está disponibilizado no GitHub [4]. Veja ainda outros artigos que publico sobre tecnologia através do meu Blog Oficial [2].

Referências

- [1] Página do Weka
<https://www.cs.waikato.ac.nz/ml/weka/>
- [2] Fernando Anselmo - Blog Oficial de Tecnologia
<http://www.fernandoanselmo.blogspot.com.br/>
- [3] Encontre essa e outras publicações em
<https://cetrex.academia.edu/FernandoAnselmo>
- [4] Repositório para os fontes da apostila
<https://github.com/fernandoans/publicacoes>