

Weka

Uso de APIs

2023

Conteúdo

- Weka
- Interface e APIs
- Fontes de dados
- Formato de arquivos de entrada
- Principais classes usadas
- Fluxo de análise de dados



Weka

- Waikato Environment for Knowledge Analysis
 - Waikato University
- Principais objetivos
 - Tornar técnicas de ML disponíveis de maneira geral
 - Aplicar essas técnicas a problemas pragmáticos
 - Em particular para a indústria da Nova Zelândia
 - Desenvolver novos algoritmos de ML e disponibilizar de maneira geral
 - Contribuir para um framework teórico do campo
- Weka e APIs implementadas nesse projeto



Weka

- Material de apoio e documentação
 - Projeto com décadas
 - Documentação e material de apoio abundante
- Livros disponíveis para download no site do projeto
- Manuais
- Javadocs
- Documentação para criação de novos classificadores e clusterizadores

Machine Learning Courses

We have put together several free online courses that teach machine learning and data mining using Weka. The videos for the courses are available on Youtube. The courses are hosted on the FutureLearn platform.

Data Mining with Weka



Course material

More Data Mining with Weka.



Course material

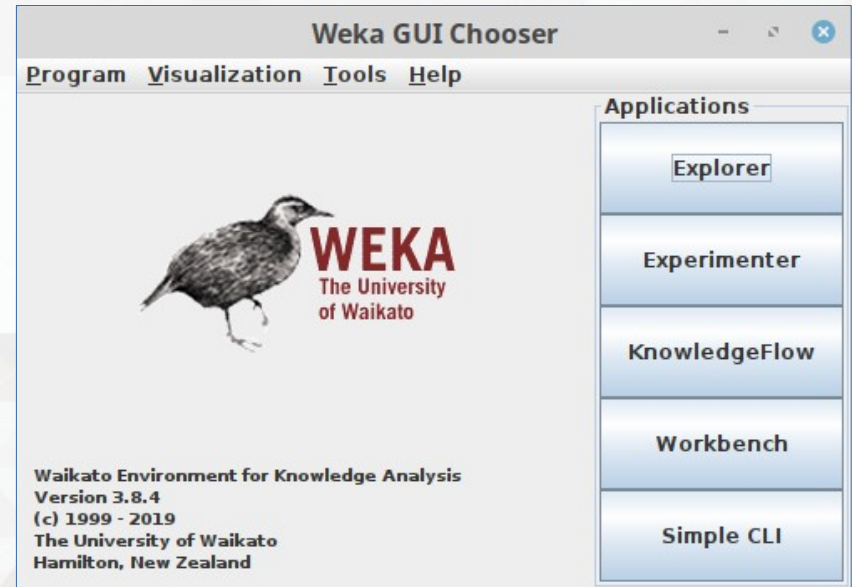
Advanced Data Mining with Weka



Course material

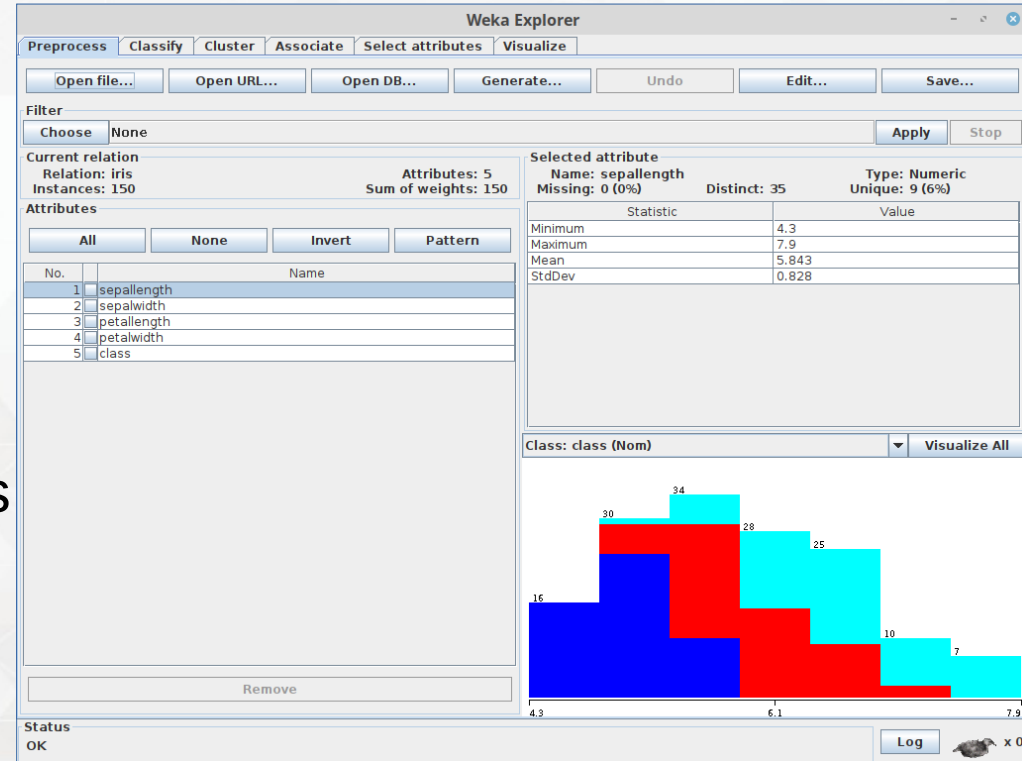
Weka - Interfaces

- Interface gráfica
- Linha de comando
 - Simple CLI
- APIs em Java
 - Programação de modelos
- Integração com outras ferramentas
 - Execução de jobs em clusters Hadoop e Spark
 - distributedWekaBase e distributedWekaSpark



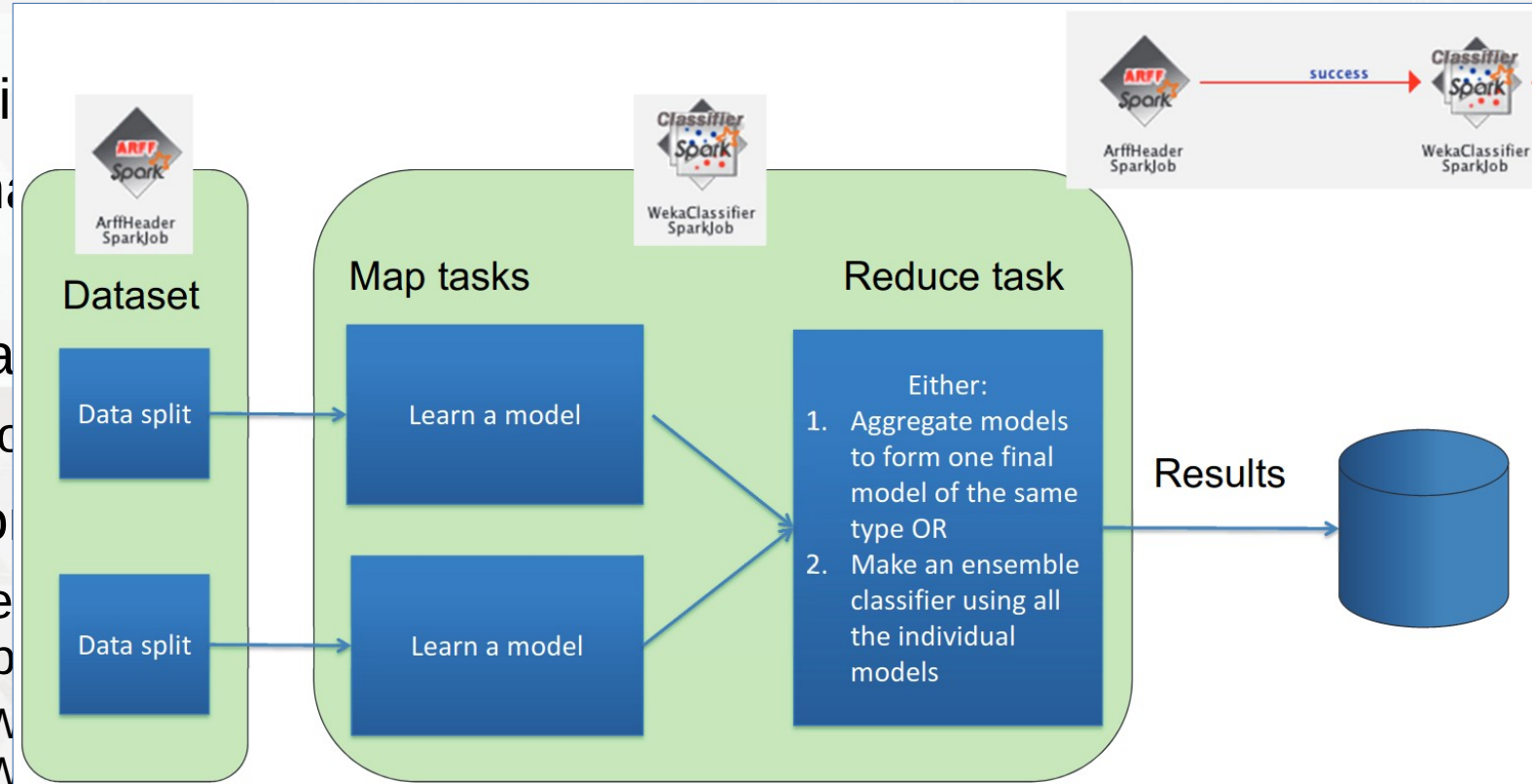
Weka - Interfaces

- Interface gráfica
- Linha de comando
 - Simple CLI
- APIs em Java
 - Programação de modelos
- Integração com outras ferramentas
 - Execução de jobs em clusters Hadoop e Spark
 - distributedWekaBase e distributedWekaSpark



Weka - Interfaces

- Interface gráfica
- Linha de comando
 - Simple CLI
- APIs em Java
- Programação
- Integração com
 - Execução de Hadoop e Spark
 - distributedW
 - distributedW

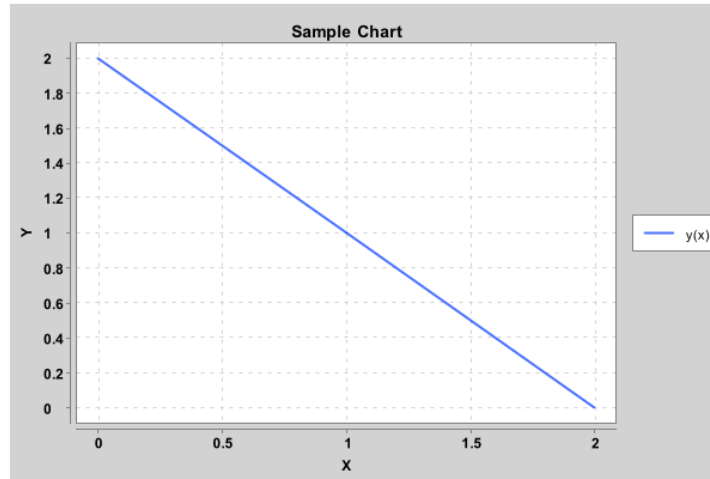


Weka - Interfaces

- Notebooks em Java
 - Jupyter Notebook
 - Apache Zeppelin
- IJava
 - Kernel Java para notebooks
 - <https://github.com/SpencerPark/IJava>

```
In [1]: 1 %maven org.knowm.xchart:xchart:3.5.2
        2
        3 import org.knowm.xchart.*;
        4
        5 double[] xData = new double[] { 0.0, 1.0, 2.0 };
        6 double[] yData = new double[] { 2.0, 1.0, 0.0 };
        7
        8 XYChart chart = QuickChart.getChart("Sample Chart", "X", "Y", "y(x)", xData, yData);
        9
       10 BitmapEncoder.getBufferedImage(chart);
```

Out[1]:



Formato de arquivos de entrada

- ARFF
 - Attribute-Relation File Format
 - Conversores de e para ARFF
- CSV
 - Com formato determinado
- Bancos de dados
 - Loaders para bancos de dados com drivers JDBC
 - Tipos devem ser convertidos aos tipos do Weka
 - Arquivo DatabaseUtils.props
 - Exemplos de configuração em weka.experiment



Arquivos ARFF

- Attribute-Relation File Format
 - Lista de instâncias com um conjunto de atributos
- Duas seções
 - Header
 - Nome, lista de atributos e tipos
 - Numeric, string, date, nominal, relational
 - Data
 - Dados conforme o header
 - Densos ou esparsos

```
@RELATION iris
```

```
@ATTRIBUTE sepallength REAL
```

```
@ATTRIBUTE sepalwidth REAL
```

```
@ATTRIBUTE petallength REAL
```

```
@ATTRIBUTE petalwidth REAL
```

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

```
@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
```

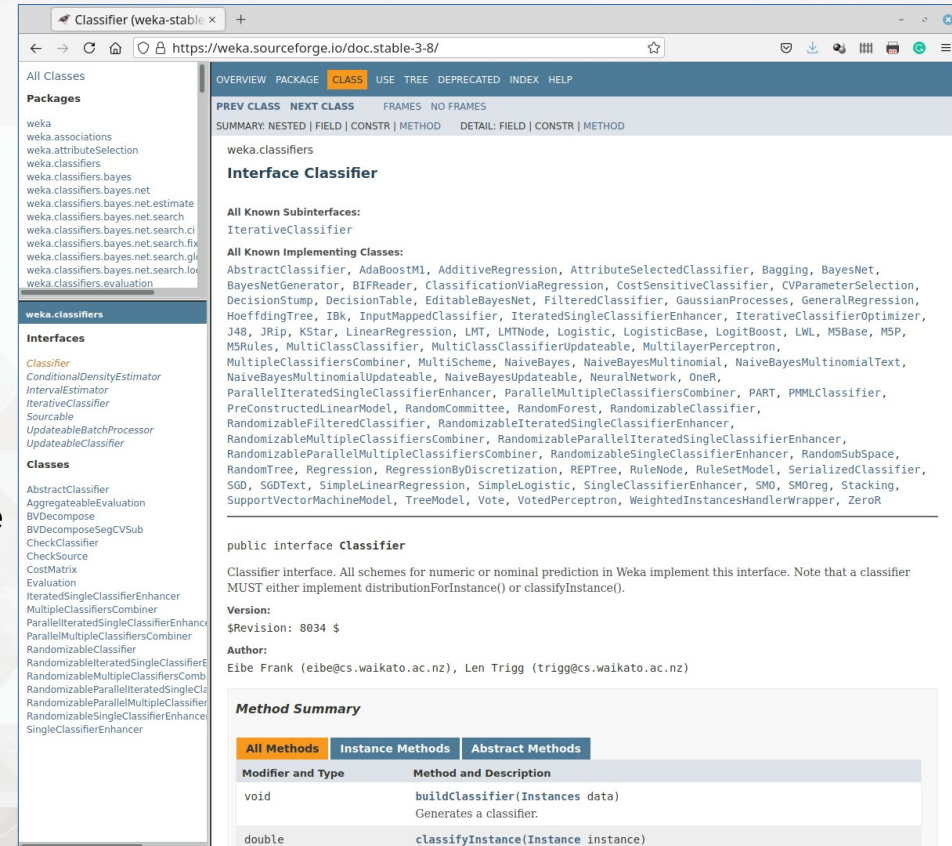
Weka API

- Tarefas comuns
 - Configurando opções
 - Criando datasets em memória
 - Carregando e salvando dados
 - Filtering
 - Classificação
 - Clustering
 - Selecionando atributos
 - Visualização
 - Serialização



Weka API

- Classes mais comuns
 - Instance
 - Item a ser classificado
 - Instances
 - Conjunto de itens para treinamento ou classificação
 - DataSet
 - Classifier / Clusterer
 - Interfaces (classes 100% abstratas)
 - Propagam comportamento para todos os classificadores e clusters
 - Evaluation
 - Avaliação de modelos
 - Cross-folding, random splits, métricas
 - Attribute selection / filtering



The screenshot shows the Weka API documentation page for the `Classifier` interface. The page is titled "Classifier (weka-stable)" and shows the URL `https://weka.sourceforge.io/doc.stable-3-8/`. The left sidebar lists the "All Classes" and "Packages" available in the API. The main content area shows the "Interface Classifier" with a list of "All Known Subinterfaces" and "All Known Implementing Classes". Below this, the "Method Summary" table is displayed, showing the methods `buildClassifier` and `classifyInstance`.

Interface Classifier

All Known Subinterfaces:

- `IterativeClassifier`

All Known Implementing Classes:

- `AbstractClassifier`, `AdaBoostM1`, `AdditiveRegression`, `AttributeSelectedClassifier`, `Bagging`, `BayesNet`, `BayesNetGenerator`, `BIFReader`, `ClassificationViaRegression`, `CostSensitiveClassifier`, `CVParameterSelection`, `Decision stump`, `DecisionTable`, `EditableBayesNet`, `FilteredClassifier`, `GaussianProcesses`, `GeneralRegression`, `HoeffdingTree`, `IBk`, `InputMappedClassifier`, `IteratedSingleClassifierEnhancer`, `IterativeClassifierOptimizer`, `J48`, `JRip`, `KStar`, `LinearRegression`, `LMT`, `LMTNode`, `Logistic`, `LogisticBase`, `LogitBoost`, `LWL`, `M5Base`, `MSP`, `M5Rules`, `MultiClassClassifier`, `MultiClassClassifierUpdateable`, `MultiLayerPerceptron`, `MultipleClassifiersCombiner`, `MultiScheme`, `NaiveBayes`, `NaiveBayesMultinomial`, `NaiveBayesMultinomialText`, `NaiveBayesMultinomialUpdateable`, `NaiveBayesUpdateable`, `NeuralNetwork`, `OneR`, `ParallelIteratedSingleClassifierEnhancer`, `ParallelMultipleClassifiersCombiner`, `PART`, `PMMLClassifier`, `PreConstructedLinearModel`, `RandomCommittee`, `RandomForest`, `RandomizableClassifier`, `RandomizableFilteredClassifier`, `RandomizableIteratedSingleClassifierEnhancer`, `RandomizableMultipleClassifiersCombiner`, `RandomizableParallelIteratedSingleClassifierEnhancer`, `RandomizableParallelMultipleClassifiersCombiner`, `RandomizableSingleClassifierEnhancer`, `RandomSubSpace`, `RandomTree`, `Regression`, `RegressionByDiscretization`, `REPTree`, `RuleNode`, `RuleSetModel`, `SerializedClassifier`, `SGD`, `SGDText`, `SimpleLinearRegression`, `SimpleLogistic`, `SingleClassifierEnhancer`, `SMD`, `SMDreg`, `Stacking`, `SupportVectorMachineModel`, `TreeModel`, `Vote`, `VotedPerceptron`, `WeightedInstancesHandlerWrapper`, `ZeroR`

public interface Classifier

Classifier interface. All schemes for numeric or nominal prediction in Weka implement this interface. Note that a classifier MUST either implement `distributionForInstance()` or `classifyInstance()`.

Version:
\$Revision: 8034 \$

Author:
Eibe Frank (eibe@cs.waikato.ac.nz), Len Trigg (trigg@cs.waikato.ac.nz)

Method Summary

Modifier and Type	Method and Description
void	<code>buildClassifier(Instances data)</code> Generates a classifier.
double	<code>classifyInstance(Instance instance)</code>

Weka API

- Carregando dados
 - ARFFs são carregados de forma simples
 - CSVs são usados através de Converters
 - Bancos de dados podem ser lidos
 - Com driver JDBC
 - Em memória ou de maneira incremental
 - InstanceQuery
 - DatabaseLoader

```
// carregando um arquivo ARFF  
  
DataSource source = new  
DataSource("/datasets/dados.arff");  
  
Instances data = source.getDataSet();
```

Weka API

- Carregando dados
 - ARFFs são carregados de forma simples
 - CSVs são usados através de Converters
 - Bancos de dados podem ser lidos
 - Com driver JDBC
 - Em memória ou de maneira incremental
 - InstanceQuery
 - DatabaseLoader

```
// carregando um arquivo CSV

DataSource source1 = new
DataSource("/datasets/dados.csv");

Instances data1 = source1.getDataSet();

// customizando o carregamento de CSV

CSVLoader loader = new CSVLoader();

loader.setSource(new
File("/datasets/dados.csv"));

Instances data2 = loader.getDataSet();
```


Weka API

- JDBC
 - Driver deve estar no CLASSPATH
- Em memória ou de maneira incremental
 - InstanceQuery
 - DatabaseLoader
 - Incremental com MySQL ou HSQLDB
- Arquivo de configuração
 - DatabaseUtils.props
 - Deve estar no diretório home do usuário ou no projeto da IDE
 - Conversão de tipos, palavras reservadas, etc
 - Exemplos no package weka.experiment

```
// carregando um dataset de uma tabela para a memória

InstanceQuery query = new InstanceQuery();

query.setDatabaseURL("jdbc:postgresql://localhost/datasets");
query.setUsername("usuariodb");
query.setPassword("senha");

query.setQuery("select * from diabetesdiag");

Instances instancias = query.retrieveInstances();
```

Weka API

- JDBC
 - Driver deve estar no CLASSPATH
- Em memória ou de maneira incremental
 - InstanceQuery
 - DatabaseLoader
 - Incremental com MySQL ou HSQLDB
- Arquivo de configuração
 - DatabaseUtils.props
 - Deve estar no diretório home do usuário ou no projeto da IDE
 - Conversão de tipos, palavras reservadas, etc
 - Exemplos no package weka.experiment

```
// carregando um dataset de uma tabela para a memória

DatabaseLoader dbLoader = new DatabaseLoader();

dbLoader.setSource("jdbc:postgresql://localhost/datasets",
"usuario", "senha");

dbLoader.setQuery("select * from diabetesdiag");

Instances instancias = dbLoader.getDataSet();
```

Weka API

- Classificação

- Interface Classifier

- Diversos packages com diferentes algoritmos
 - Novos classificadores podem ser facilmente implementados

- Classificação em batch

- Todo o dataset é carregado em memória
 - Método `buildClassifier(Instances)`

- Classificação incremental

- Treinamento incremental, conservando memória
 - Os dados não precisam estar todos em memória ao mesmo tempo
 - Métodos `buildClassifier(Instances)` e `updateClassifier(Instance)`

Packages de classificadores:

`Weka.classifiers`

- Bayes
- Lazy
- Pmml
- Rules
- trees

Weka API

- Classificação

- Interface Classifier

- Diversos packages com diferentes algoritmos
 - Novos classificadores podem ser facilmente implementados

- Classificação em batch

- Todo o dataset é carregado em memória
 - Método buildClassifier(Instances)

- Classificação incremental

- Treinamento incremental, conservando memória
 - Os dados não precisam estar todos em memória ao mesmo tempo
 - Métodos buildClassifier(Instances) e updateClassifier(Instance)

```
// árvore de decisão
```

```
// carregar dataset
```

```
DataSource ds = new
```

```
    DataSource("/datasets/iris.csv");
```

```
Instances instancias = ds.getDataSet();
```

```
// configurar atributo target
```

```
instancias.setClassIndex(4);
```

```
J48 arvore = new J48();
```

```
// treinar modelo
```

```
arvore.buildClassifier(instancias);
```

Weka API

- Tratamento de opções
 - As opções de cada algoritmo usadas na interface do Weka estão disponíveis na API
 - Maneira simples de configurar comportamento de modelos
 - Embora potencialmente confusa
- Interface `weka.core.OptionHandler`
 - Implementada pelos classificadores e clusterizadores
 - Método `setOptions(String[] opc)`

```
// configurando opções

String[] opc = new String[2];
opc[0] = "- R";
opc[1] = "1";

// configurar classificador
classif.setOptions(opc);
```

Weka API

- Filtragem
 - Filter
 - Remoção ou tratamento de atributos do dataset
 - Configurado via Options
 - Array de Strings
 - “-R”, “1” - por exemplo
 - Remove o primeiro atributo
 - Filtering on-the-fly
 - Filtra enquanto o dataset é processado pelo modelo
- Sempre bom pesar o uso de feature engineering usando recursos como esse
 - Principalmente com o uso de APIS
 - Onde é potencialmente mais simples tratar o dataset previamente

Weka API

- Classificando novas instâncias
 - Novo objeto Instance
 - Com dados da instância candidata
 - Sabendo qual é o dataset originário
- Resultado com índice da classe ou distribuição estatística
 - Dependendo do tipo de algoritmo escolhido
 - Resultados são retornados em qualquer caso

```
// classificando instância

Instance novo = new DenseInstance(inst.numAttributes());

novo.setDataset(inst);
novo.setValue(0, 4.9);
novo.setValue(1, 3.0);
novo.setValue(2, 1.4);
novo.setValue(3, 0.2);

double result[] = modelo.distributionForInstance(novo);

double resultClass = modelo.classifyInstance(novo);
```

Weka API

- Avaliação

- Classe Evaluation
- Cross-validation ou Train/test set

- Métricas

- Gerais
 - correct(), incorrect()
 - pctCorrect(), pctIncorrect()
- Para cada categoria do target
 - precision(int index)
 - recall(int index)
 - fMeasure(int index)
 - areaUnderROC(int index)

```
// avaliação de um modelo de árvore
```

```
Evaluation eval = new Evaluation(instancias);
```

```
eval.crossValidateModel(arvore, instancias, 10, new  
Random());
```

```
System.out.println(eval.pctCorrect());  
System.out.println(eval.toMatrixString());
```

Serialização de Modelos

- Salvar e carregar modelos
- Serialization API
 - API padrão de Java
 - ObjectOutputStream
 - writeObject
 - ObjectInputStream
 - readObject
 - Typecast necessário na leitura
- Maioria das classes da API Weka pode ser serializada, não só modelos
 - Implementam a interface Serializable
 - Algumas tem atributos transient
- Classe de auxílio SerializationHelper
 - Package weka.core
 - Métodos read e write

```
Classifier arvMdl = new J48();  
SerializationHelper.write("arvore.model", arvMdl);
```

Então

- Weka possui dois lados
 - Interface e API
- Ambos procuram tornar o fluxo de trabalho de Machine Learning mais simples
 - AutoML
- APIs são estáveis, testadas e de bom desempenho
 - E ainda podem ser executadas sobre um cluster, se necessário
- Abordagem é mais técnica em programação, entretanto
 - Alguns recursos mais voltados para programadores
 - Implementação de novas características e algoritmos



Obrigado

leandro@utfpr.edu.br

<http://lapti.ct.utfpr.edu.br>