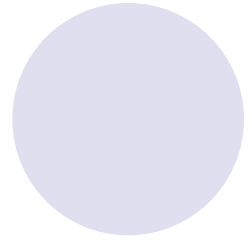
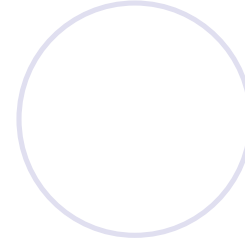
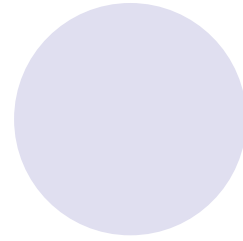


Weka



# A Machine Learning Framework



- Machine Learning

- Sub-discipline of AI to train computer programs to make predictions on future data

- Weka

- Provides algorithms and services to conduct ML experiments and develop ML applications

# History



- Received funding in 1993 from government of New Zealand
- First TCL/TK implementation released in 1996
- Rewritten in Java in 1999
- Updated Java GUI in 2003

# Main Services



- **Data Pre-Processing**

- Importing Data into Weka's Formats
- Filtering Data

- **Data Classification**

- Predict one attribute based on other attributes

- **Clustering**

- Breaking data into meaningful sub-groups



# Main Concept

## ¢ Two Main Features in WEKA framework.

- Machine-Learning Algorithms.
  - ¢ 76 classification/regression algorithms.
  - ¢ 8 clustering algorithms.
- Data Processing Tools.
  - ¢ 49 data processing tools.



# Main Concept

## ¢ Class and Package in WEKA

- Class in WEKA
  - Implementation of a particular machine learning algorithm
    - ex. J48 class in `weka.classifier.trees` package.
- Package in WEKA
  - Just a directory containing a collection of related classes.
    - ex. `weka.classifier.trees` package.



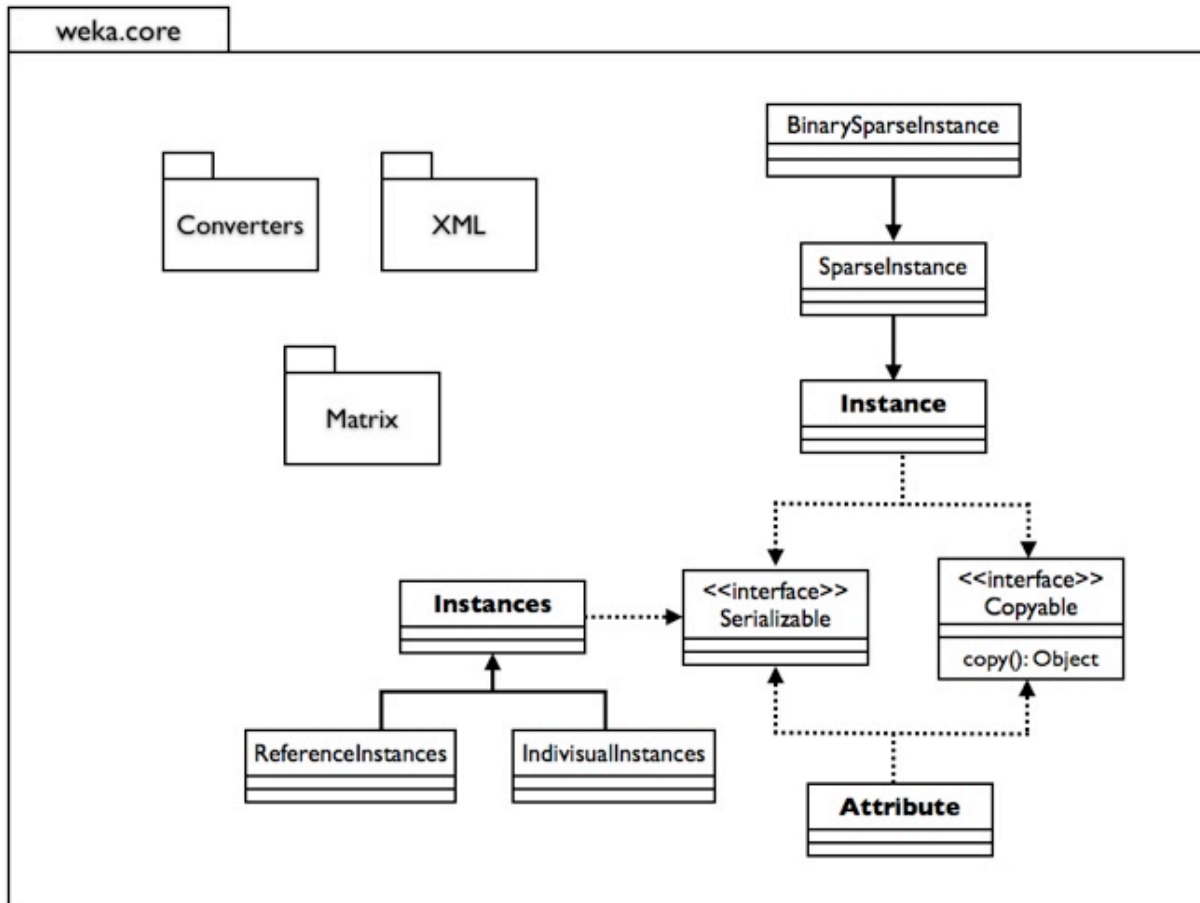
# Main Concept

## ¢ Main Packages

- weka.core package
- weka.classifiers package
- weka.filters package

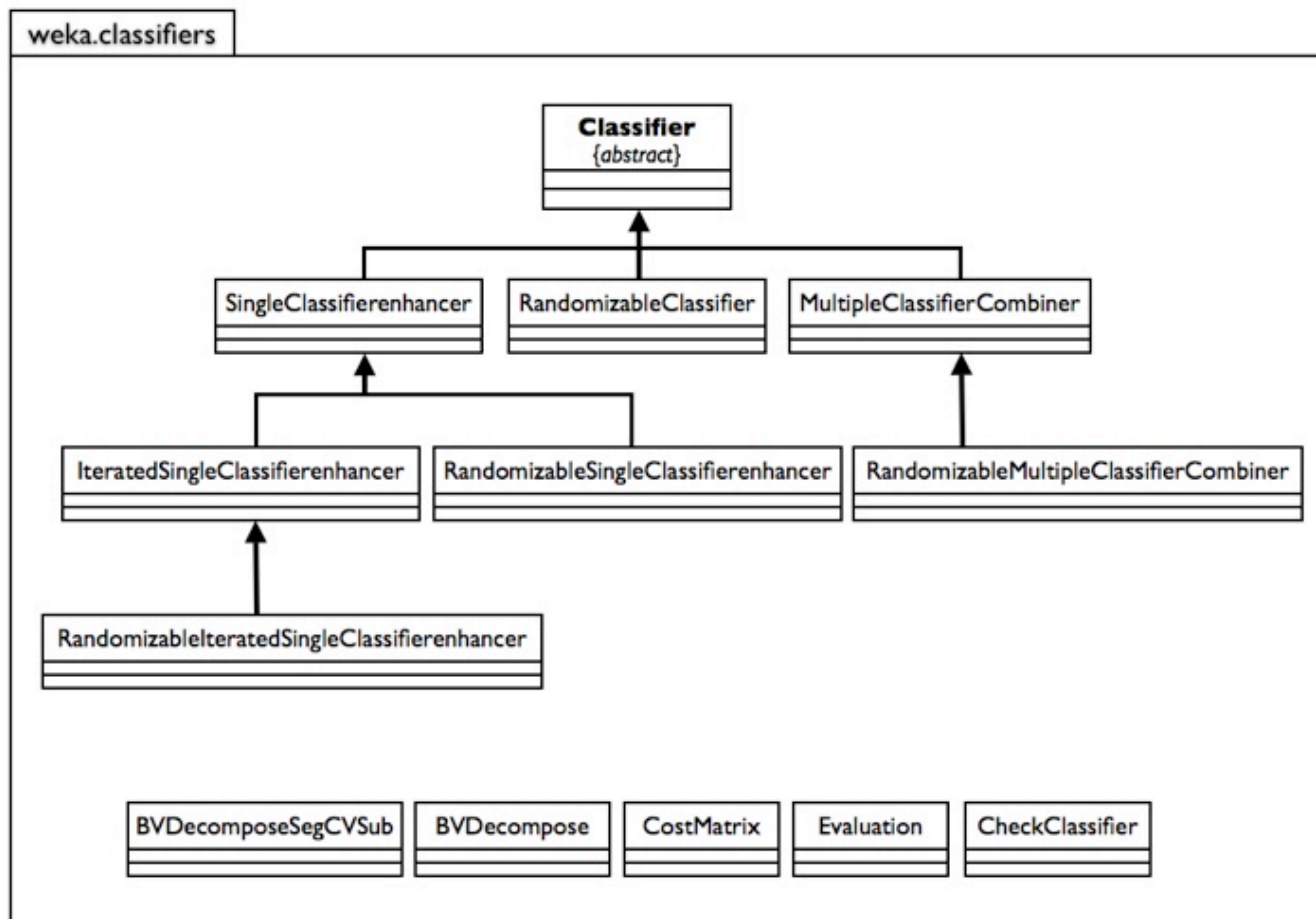
# Main Concept

¢ Class Diagram for weka.core package



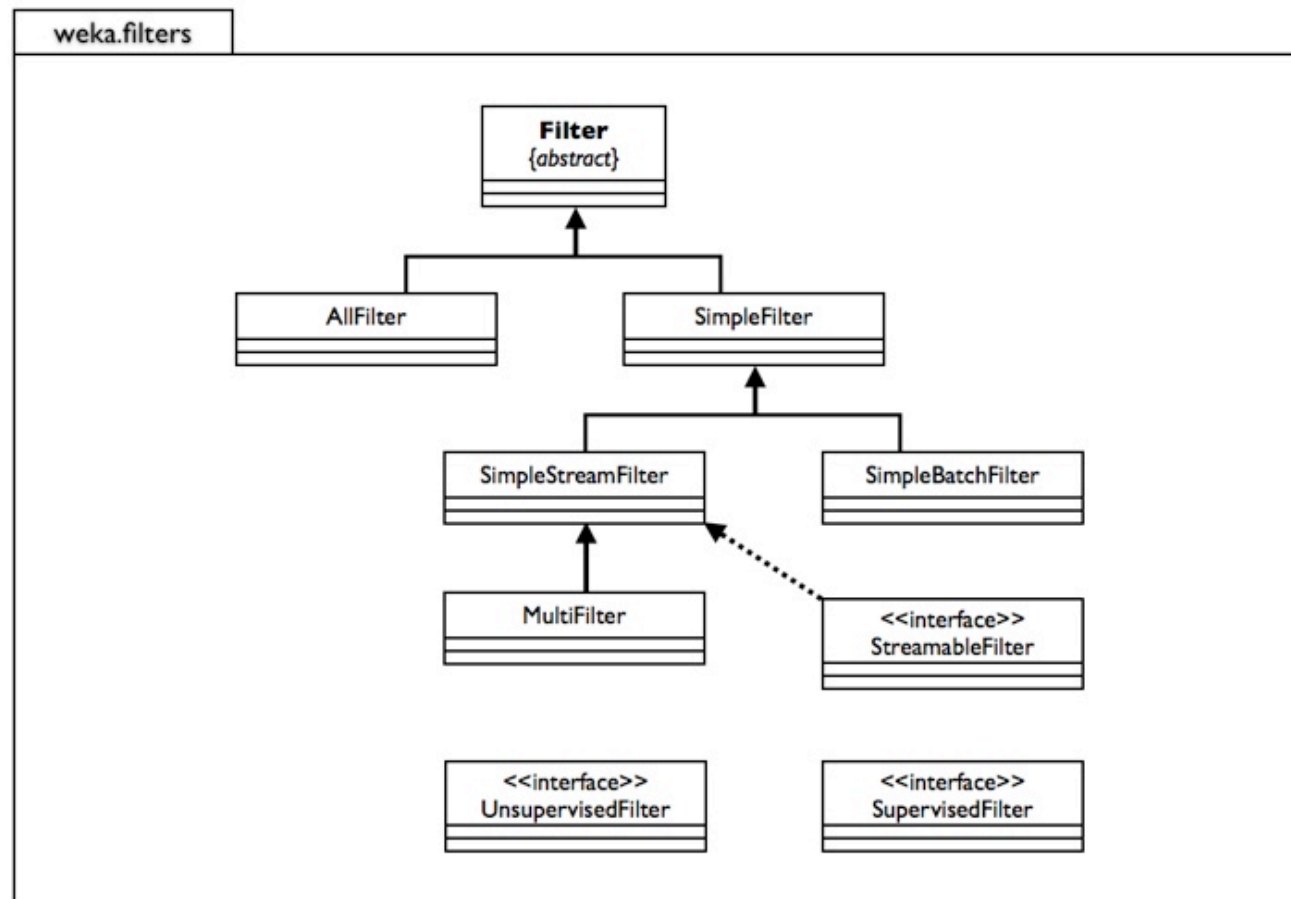
# Main Concept

¢ Class Diagram for weka.classifiers package



# Main Concept

⌘ Class Diagram for weka.filters package



# Getting Started: Data

- Start with a collection of data

- Weka specific ARFF files or other sources (DB, CSV)

```
@relation weather
```

```
@attribute outlook {sunny, overcast, rainy}
```

```
@attribute temperature real
```

```
@attribute humidity real
```

```
@attribute windy {TRUE, FALSE}
```

```
@attribute play {yes, no}
```

```
@data
```

```
sunny,85,85,FALSE,no
```

```
sunny,80,90,TRUE,no
```

```
overcast,83,86,FALSE,yes
```

```
rainy,70,96,FALSE,yes
```

# Getting Started: Code



- First load data into Instances variable

```
DataSource source = new DataSource("weather.arff");  
Instances data = source.getDataSet();
```

```
if (data.classIndex() == -1)  
    data.setClassIndex(data.numAttributes() - 1);
```

# Getting Started: Code



- Filter if necessary, then conduct experiment

```
NaiveBayes cModel = new NaiveBayes();  
...  
Evaluation eval = new Evaluation(data);  
eval.crossValidateModel(cModel, data, 10, new Random(1));  
System.out.println(eval.toSummaryString("\nResults\n=====\n",  
false));
```

# Getting Started: Code



- Printed results

Correctly Classified Instances	9	64.2857 %
Incorrectly Classified Instances	5	35.7143 %
Kappa statistic	0.1026	
Mean absolute error	0.4649	
Root mean squared error	0.543	
Relative absolute error	97.6254 %	
Root relative squared error	110.051 %	
Total Number of Instances	14	

# Getting Started: Code

- Make a prediction

```
cModel.buildClassifier(data);
double[] fDistribution = cModel.distributionForInstance(iClassify);
if(fDistribution[0] >= fDistribution[1]){
    System.out.println("Go out and play!\n");
} else {
    System.out.println("Read a book.\n");
}
```