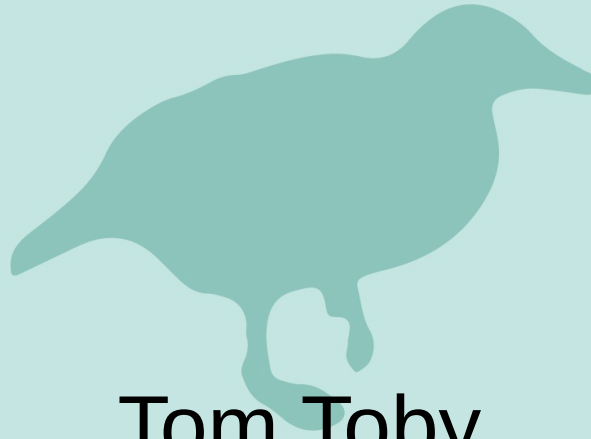


Machine Learning with WEKA

WEKA Explorer Tutorial



Tom Toby
Data Analyst
NIELIT
Calicut

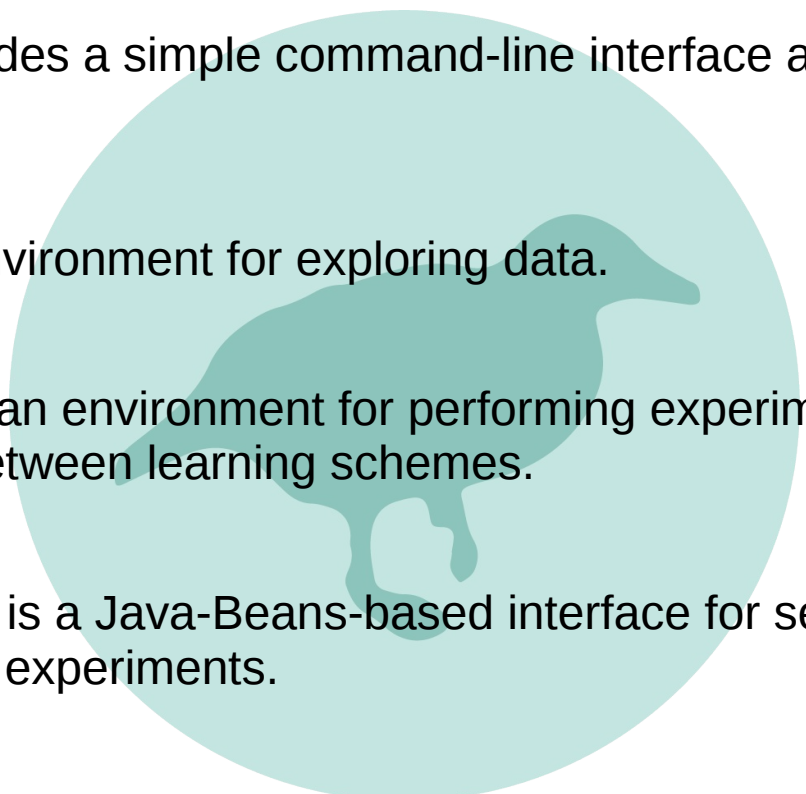
INTRODUCTION

- WEKA is a data mining system developed by the University of Waikato in New Zealand that implements data mining algorithms.
- Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.
- Weka is open source software issued under the GNU General Public License.
- Waikato Environment for Knowledge Analysis (Weka)

Launching WEKA Explorer

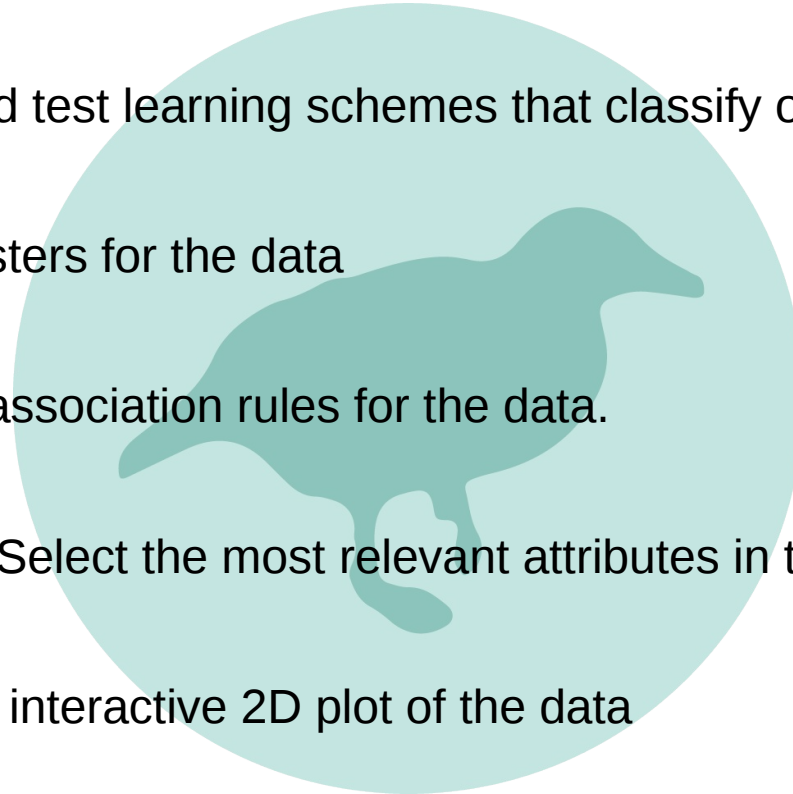
- Installation:
ubuntu : “**sudo apt install weka**”
- Basically a java program exported as runnable jar file
- For ubuntu use locate command to search for weka location
- Change directory to that destination and type
- “java -jar weka.jar
- A new GUI module will emerge

Launching WEKA Explorer

- **Simple CLI** provides a simple command-line interface and allows direct execution of Weka commands
 - **Explorer** is an environment for exploring data.
 - **Experimenter** is an environment for performing experiments and conducting statistical tests between learning schemes.
 - **KnowledgeFlow** is a Java-Beans-based interface for setting up and running machine learning experiments.
- 

Environment

- **Preprocess** Choose and modify the data being acted on
- **Classify** Train and test learning schemes that classify or perform regression.
- **Cluster** Learn clusters for the data
- **Associate** Learn association rules for the data.
- **Select attributes** Select the most relevant attributes in the data.
- **Visualize** View an interactive 2D plot of the data



Preprocessing Data

- The first four buttons at the top of the preprocess section enable you to load data into WEKA
- **Open file:** from local file system
- **Open URL** Uniform Resource Locator address
- **Open DB** Reads data from a database
- **Generate** Enables you to generate artificial data from a variety of DataGenerators

DataSets

- A dataset is roughly equivalent to a two-dimensional spreadsheet or database table
- In WEKA, it is implemented by the `weka.core.Instances` class.
- A dataset is a collection of examples, each one of class `weka.core.Instance`.
- Attribute types : nominal (= one of a predefined list of values),
numeric (= a real or integer number)
string (= an arbitrary long list of characters, enclosed in “double quotes”).
- Additional types are date and relational,

ARFF

- Attribute-Relation File Format (ARFF)
- An ARFF (= Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes.
- 2 Sections: Header information, Data information.
- Attribute values for each instance are delimited by commas
- Missing values are represented by a single question mark, as in:
- @data
- 4.4,?,1.5,?,Iris-setosa

% 1. Title: Iris Plants Database

% 2. Sources:

%(a) Creator: R.A. Fisher

%(b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)

%(c) Date: July, 1988

@RELATION iris

@ATTRIBUTE sepallength NUMERIC

@ATTRIBUTE sepalwidth NUMERIC

@ATTRIBUTE petallength NUMERIC

@ATTRIBUTE petalwidth NUMERIC

@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

5.4,3.9,1.7,0.4,Iris-setosa

More Examples

- @attribute LCC string
- @data
- AG5, 'Encyclopedias and dictionaries.;Twentieth century.'
-
- @RELATION Timestamps
- @ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"
- @DATA
- "2001-04-03 12:12:12"
- "2001-05-03 12:59:55"
- Relational data must be enclosed within double quotes "
- converters
- `java weka.core.converters.CSVLoader data.csv > data.arff`
- `java weka.core.converters.C45Loader c45_filestem > data.arff`

Preprocessing window

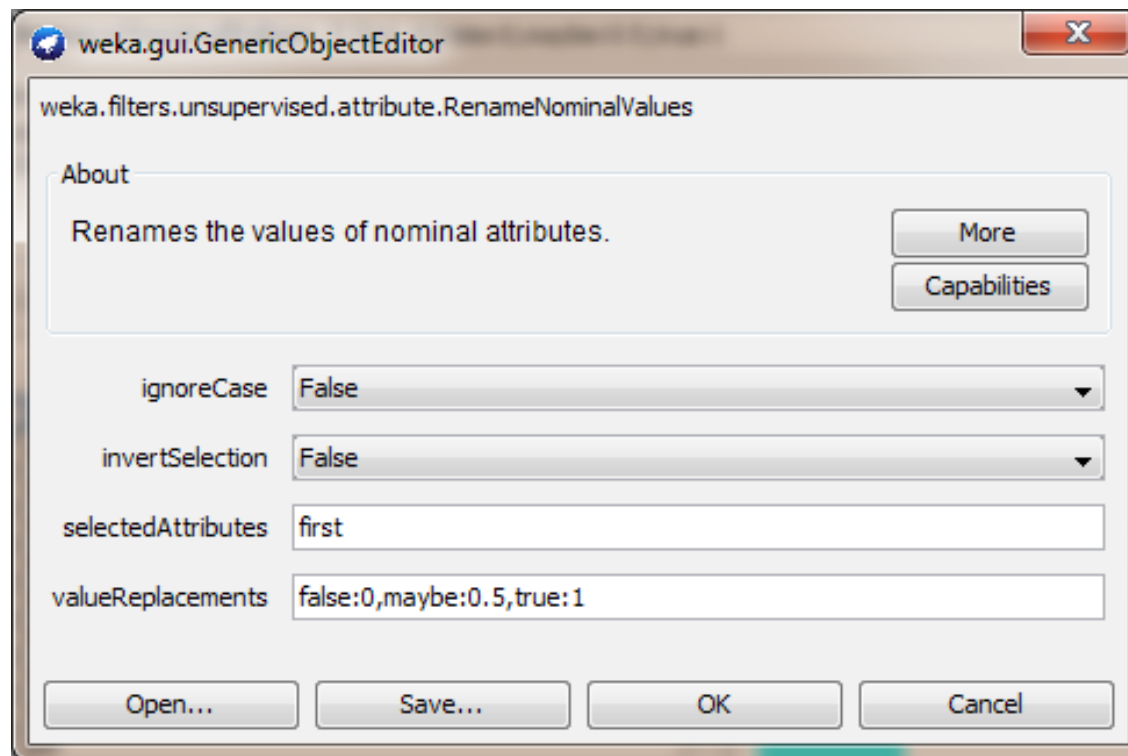
- At the bottom of the window there is 'Status' box
- When you are loading "train1.arff" file, the 'Status' box displays the message "Reading from file...".
- Once the file is loaded, the message in the 'Status' box changes to say "OK".
- Right-click anywhere in 'Status box', it brings up a menu with two options:
 - 1. Available Memory that displays in the log and in 'Status' box the amount of memory available to WEKA in bytes.
 - 2. Run garbage collector that forces Java garbage collector to search for memory that is no longer used, free this memory up and to allow this memory for new tasks.
- To the right of the 'Log' button there is an image of a bird. The bird is WEKA status icon.
- The number next to 'X' symbol indicates a number of concurrently running processes.

Loading data

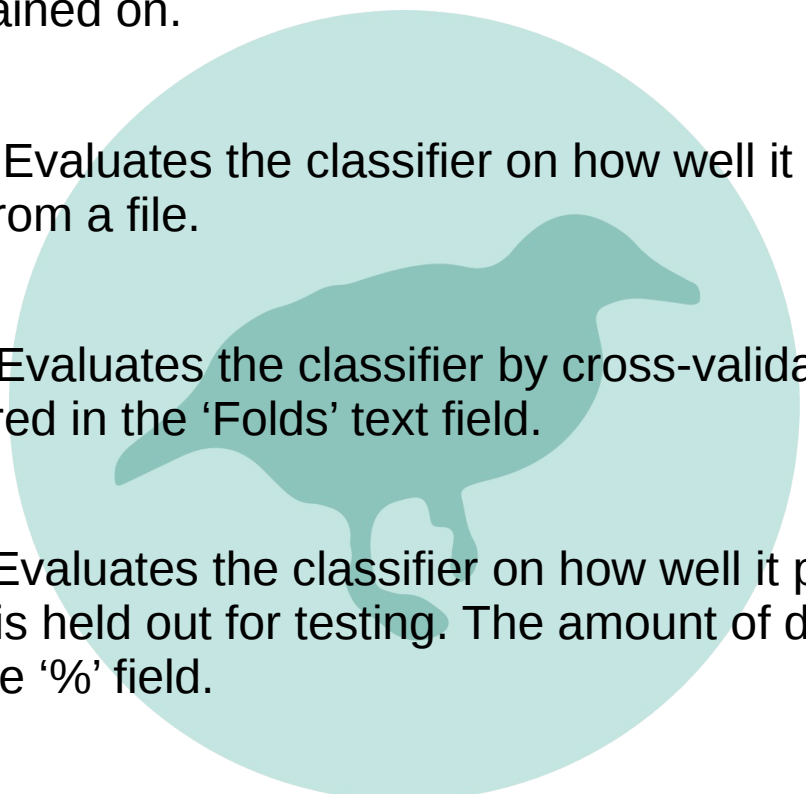
- The easiest way is ARFF file
- The data can be loaded from CSV file as well because some databases have the ability to convert data only into CSV format.
- **Attribute** window: Left panel of 'Preprocess' window shows the list of recognized attributes
- **Name** is the name of an attribute,
- **Type** is most commonly Nominal or Numeric
- **Missing** is the number (percentage) of instances in the data for which this attribute is unspecified,
- **Distinct** is the number of different values that the data contains for this attribute
- **Unique** is the number (percentage) of instances in the data having a value for this attribute no other instances have.

Setting Filters

WEKA contains filters for discretization, normalization, resampling, attribute selection, transformation and combination of attributes



Classifiers

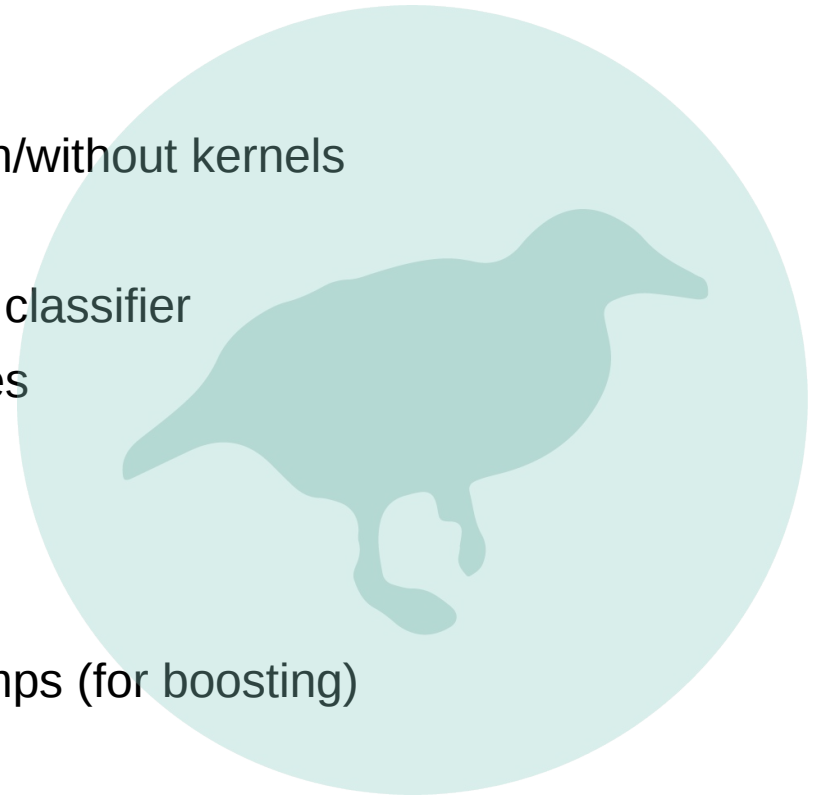
- **Use training set.** Evaluates the classifier on how well it predicts the class of the instances it was trained on.
 - **Supplied test set** Evaluates the classifier on how well it predicts the class of a set of instances loaded from a file.
 - **Cross-validation** Evaluates the classifier by cross-validation, using the number of folds that are entered in the 'Folds' text field.
 - **Percentage split** Evaluates the classifier on how well it predicts a certain percentage of the data, which is held out for testing. The amount of data held out depends on the value entered in the '%' field.
- 

Classifiers

- ZeroR's (`weka.classifiers.rules.ZeroR`) :
model just consists of a single value the most common class, or the median of all numeric values in case of predicting a numeric value (= regression learning).
- ZeroR is a trivial classifier, but it gives a lower bound on the performance
- **bayes**: Algorithms that use Bayes Theorem in some core way, like Naive Bayes.
- **function**: Algorithms that estimate a function, like Linear Regression.
- **lazy**: Algorithms that use lazy learning, like k-Nearest Neighbors.
- **meta**: Algorithms that use or combine multiple algorithms, like Ensembles.
- **misc**: Implementations that do not neatly fit into the other groups, like running a saved model.
- **rules**: Algorithms that use rules, like One Rule.
- **trees**: Algorithms that use decision trees, like Random Forest.

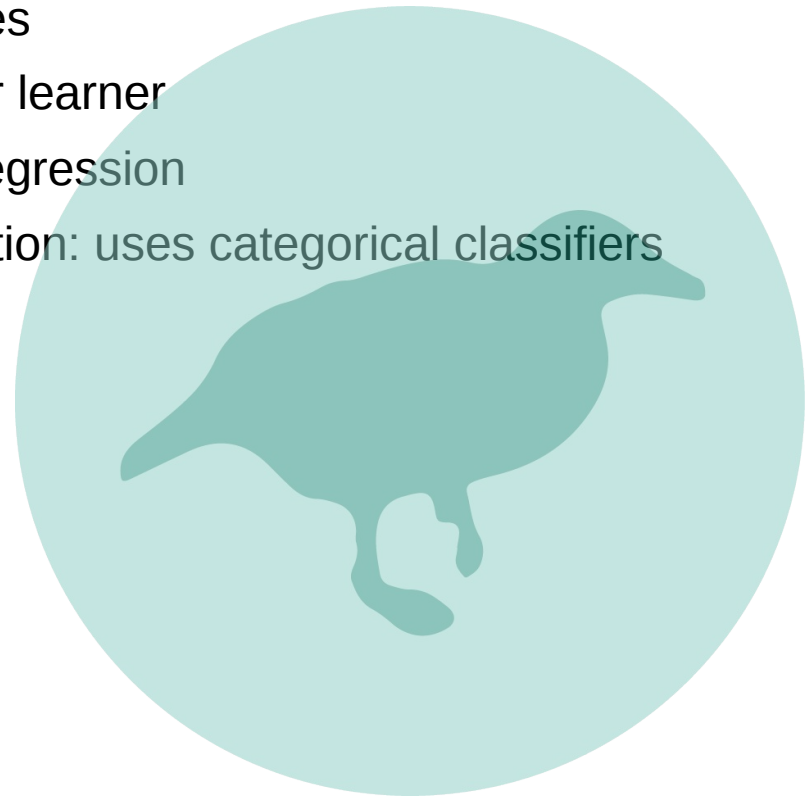
Classifiers for categorical prediction

- `weka.classifiers.IBk`: k-nearest neighbour learner
- `weka.classifiers.j48.J48`: C4.5 decision trees
- `weka.classifiers.j48.PART`: rule learner
- `weka.classifiers.NaiveBayes`: naive Bayes with/without kernels
- `weka.classifiers.OneR`: Holte's OneR
- `weka.classifiers.KernelDensity`: kernel density classifier
- `weka.classifiers.SMO`: support vector machines
- `weka.classifiers.Logistic`: logistic regression
- `weka.classifiers.AdaBoostM1`: AdaBoost
- `weka.classifiers.LogitBoost`: logit boost
- `weka.classifiers.DecisionStump`: decision stumps (for boosting)



Classifiers for numeric prediction

- `weka.classifiers.LinearRegression`: linear regression
- `weka.classifiers.m5.M5Prime`: model trees
- `weka.classifiers.IBk`: k-nearest neighbour learner
- `weka.classifiers.LWR`: locally weighted regression
- `weka.classifiers.RegressionByDiscretization`: uses categorical classifiers
- **Association rules:**
 - `java weka.associations.Apriori -t`



Linear Machine Learning Algorithms

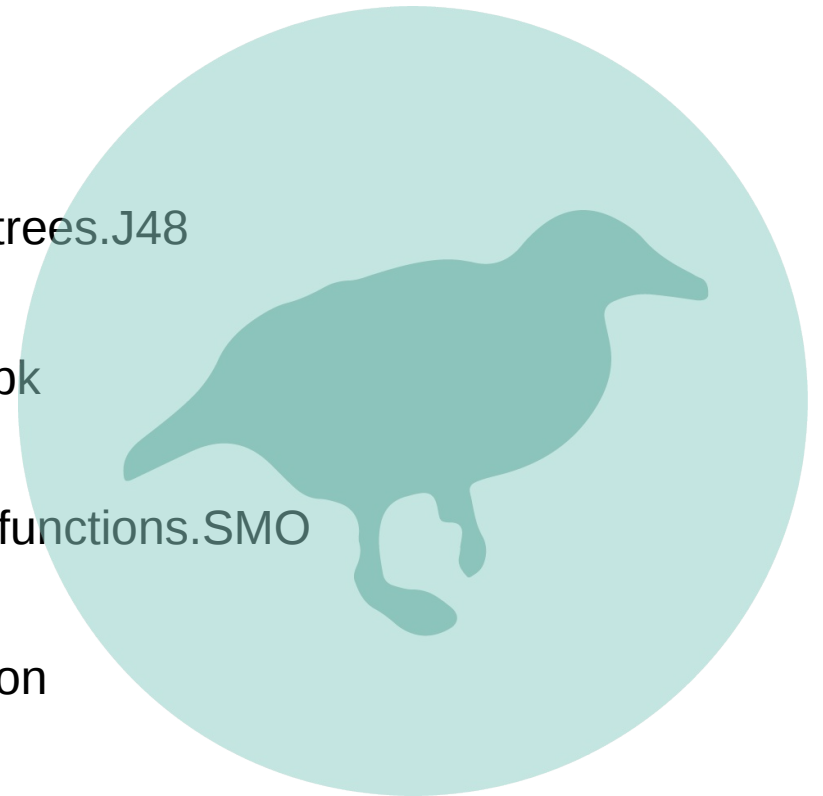
- Linear algorithms assume that the predicted attribute is a linear combination of the input attributes
- **Linear Regression:** `function.LinearRegression`
- **Logistic Regression:** `function.Logistic`



Nonlinear Machine Learning Algorithms

Nonlinear algorithms do not make strong assumptions about the relationship between the input attributes and the output attribute being predicted

- Naive Bayes: `bayes.NaiveBayes`
- Decision Tree (specifically the C4.5 variety): `trees.J48`
- k-Nearest Neighbors (also called KNN: `lazy.Ibk`)
- Support Vector Machines (also called SVM): `functions.SMO`
- Neural Network: `functions.MultilayerPerceptron`



Ensemble Machine Learning Algorithms

Ensemble methods combine the predictions from multiple models in order to make more robust predictions

- Random Forest: `trees.RandomForest`
- Bootstrap Aggregation (also called Bagging): `meta.Bagging`
- Stacked Generalization (also called Stacking or Blending): `meta.Stacking`

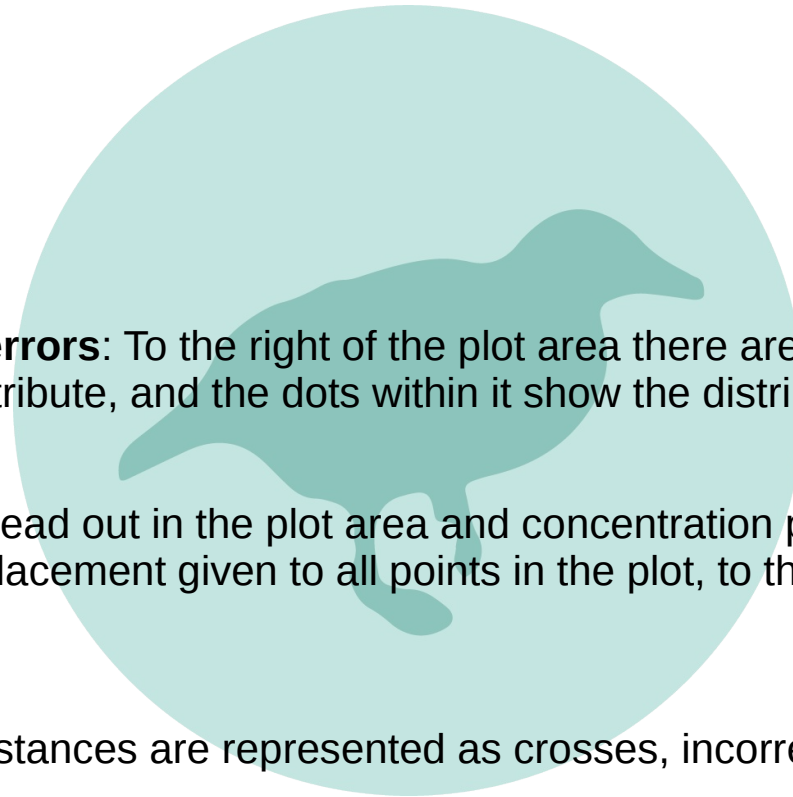


Classifier Evaluation Options

- **Output model** The output is the classification model on the full training set, so that it can be viewed, visualized, etc.
- **Output per-class stats** The precision/recall and true/false statistics for each class output.
- **Output confusion matrix** The confusion matrix of the classifier's predictions is included in the output.
- **Store predictions for visualization** The classifier's predictions are remembered so that they can be visualized.
- **'Random seed for Xval / % Split'** to 1. This specifies the random seed used when randomizing the data before it is divided up for evaluation purposes
- **Output entropy evaluation measures**
- **Output predictions**

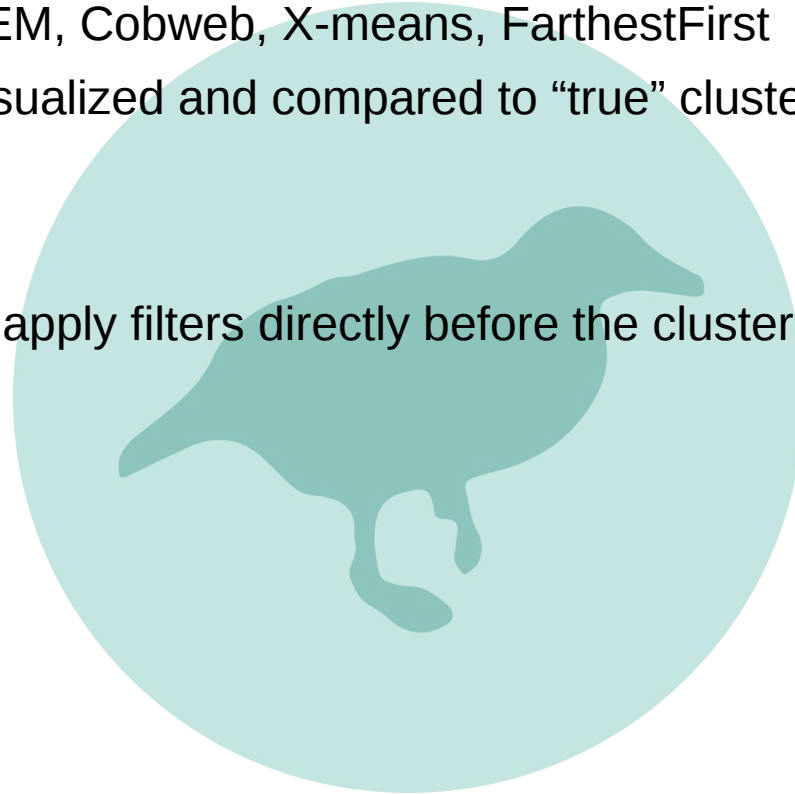
Results

- Analyzing Results
- Visualizing results :
- If tree: visualize tree
- **Visualize classifier errors:** To the right of the plot area there are series of horizontal strips. Each strip represents an attribute, and the dots within it show the distribution values of the attribute
- The instances are spread out in the plot area and concentration points are not visible. Keep sliding 'Jitter', a random displacement given to all points in the plot, to the right, until you can spot concentration points
- Correctly classified instances are represented as crosses, incorrectly classified once represented as squares



Clustering Data

- For finding groups of similar instances in a dataset
- Weka : k-Means, EM, Cobweb, X-means, FarthestFirst
- Clusters can be visualized and compared to “true” clusters (if given)
- Kmeans
- Ignore attributes
- FilteredClusterer : apply filters directly before the clusterer is learned.



Finding Associations

- An implementation of the Apriori algorithm for learning association rules
- It works only with discrete data
- Will identify statistical dependencies between groups of attributes
- The association rule scheme cannot handle numeric values
- use grocery store data from the “grocery.arff” file where all values are nominal.

$$\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

$$\text{Confidence (A} \rightarrow \text{B)} = \frac{\text{Support(A} \cup \text{B)}}{\text{Support(A)}}$$

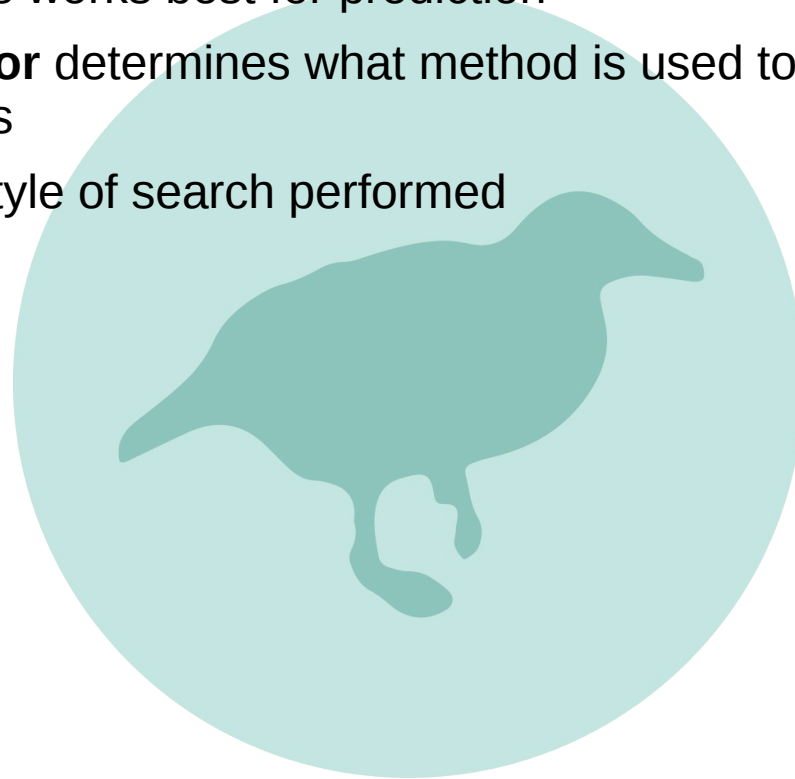
- Some fields where Apriori is used:
 - In Education Field: Extracting association rules in data mining of admitted students through characteristics and specialties.
 - In the Medical field: For example Analysis of the patient's database.
 - In Forestry: Analysis of probability and intensity of forest fire with the forest fire data.
 - Apriori is used by many companies like Amazon in the Recommender System and by Google for the auto-complete feature.

Finding Associations

- 'Associator' box :
- 'minMetric' to 0.4 for confidence = 40%
- Make sure that the default value of rules is set to 100.
- The upper bound for minimum support 'upperBoundMinSupport' should be set to 1.0 (100%) and 'lowerBoundMinSupport' to 0.1
- The algorithm halts when either the specified number of rules is generated, or the lower bound for minimum support is reached
- The number preceding = => symbol indicates the rule's support, that is, the number of items covered by its premise. Following the rule is the number of those items for which the rule's consequent holds as well

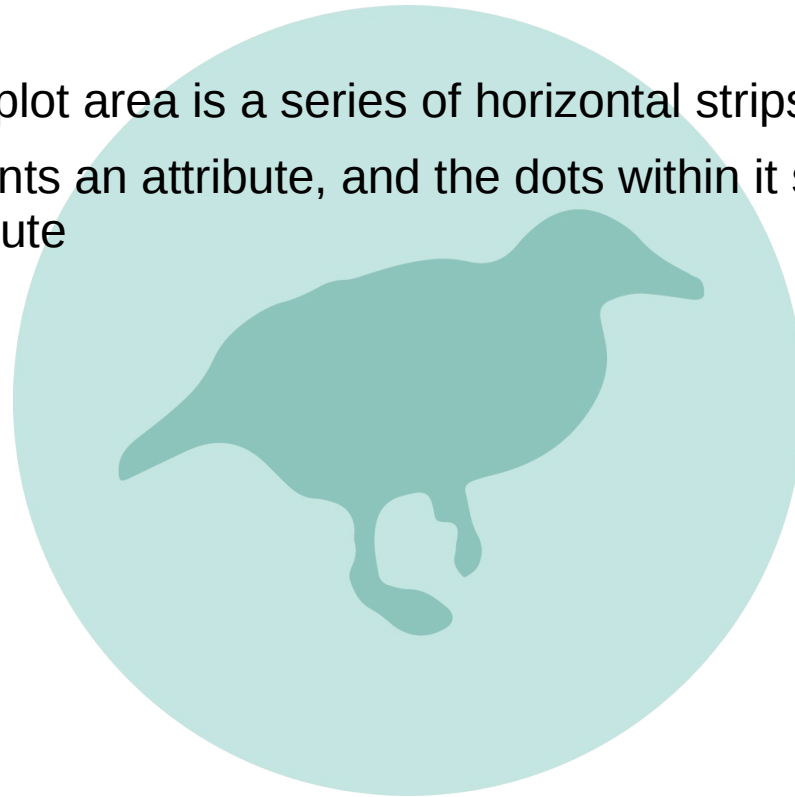
Selecting Attributes

- searching through all possible combinations of attributes in the data to find which subset of attributes works best for prediction
- **Attribute Evaluator** determines what method is used to assign a worth to each subset of attributes
- **Search Method** style of search performed



Visualizing

- Scatter plot matrix
- Select each plot
- To the right of the plot area is a series of horizontal strips.
- Each strip represents an attribute, and the dots within it show the distribution of values of the attribute



Selecting Instances

- Below the y-axis selector button is a drop-down list
- **Select Instance** Clicking on an individual data point brings up a window listing its attributes
- **Rectangle** selects the points inside it
- **Polygon**
- **Polyline** distinguishes the points on one side from those on the other
- **Submit** removes all instances from the plot except those within the grey selection area
- **Clear** erases the selected area without affecting the graph

