**ML methods for benchmarking comparison**

Values of threefold cross-validation were generated by different ML approaches for each dataset using the WEKA program. Diagram

Description automatically generated

The classification algorithms used were.

1. SVM

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A picture containing calendar

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

2. Decision trees (DT)

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing text

Description automatically generatedA picture containing graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A picture containing graphical user interface

Description automatically generatedA picture containing graphical user interface

Description automatically generated

3. RFGraphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

A picture containing text, screenshot, indoor

Description automatically generated

4.Naive Bayes (NB)

By default a Gaussian distribution is assumed for each numerical attributes.

You can change the algorithm to use a kernel estimator with the useKernelEstimator argument that may better match the actual distribution of the attributes in your dataset. Alternately, you can automatically convert numerical attributes to nominal attributes with the useSupervisedDiscretization parameter.

A picture containing text, screenshot, indoor

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generatedA picture containing graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

5. Multilayer perceptron (MLP) with a single hidden layer with 10 neurons

Table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

6. k-NN

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Clustering tested were.

1. k-means (K-M)

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

2. Hierarchical clustering (HC)

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Information gain**[[edit](https://en.wikipedia.org/w/index.php?title=Decision_tree_learning&action=edit&section=5)]

*Main article:*[*Information gain in decision trees*](https://en.wikipedia.org/wiki/Information_gain_in_decision_trees)

Used by the [ID3](https://en.wikipedia.org/wiki/ID3_algorithm), [C4.5](https://en.wikipedia.org/wiki/C4.5_algorithm) and C5.0 tree-generation algorithms. [Information gain](https://en.wikipedia.org/wiki/Information_gain) is based on the concept of [entropy](https://en.wikipedia.org/wiki/Information_entropy) and [information content](https://en.wikipedia.org/wiki/Information_content) from [information theory](https://en.wikipedia.org/wiki/Information_theory).

Entropy is defined as below

{\displaystyle \mathrm {H} (T)=\operatorname {I} \_{E}\left(p\_{1},p\_{2},...,p\_{J}\right)=-\sum \_{i=1}^{J}{p\_{i}\log \_{2}p\_{i}}}

where {\displaystyle p\_{1},p\_{2},...}are fractions that add up to 1 and represent the percentage of each class present in the child node that results from a split in the tree.[[19]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-Witten_2011_102%E2%80%93103-19)

{\displaystyle \overbrace {IG(T,a)} ^{\text{Information Gain}}=\overbrace {\mathrm {H} (T)} ^{\text{Entropy (parent)}}-\overbrace {\mathrm {H} (T|a)} ^{\text{Sum of Entropy (Children)}}}

{\displaystyle =-\sum \_{i=1}^{J}p\_{i}\log \_{2}{p\_{i}}-\sum \_{i=1}^{J}-\Pr(i|a)\log \_{2}{\Pr(i|a)}}

Averaging over the possible values of {\displaystyle A},

{\displaystyle \overbrace {E\_{A}\left(IG(T,a)\right)} ^{\text{Expected Information Gain}}=\overbrace {I(T;A)} ^{\text{Mutual Information between T and A}}=\overbrace {\mathrm {H} (T)} ^{\text{Entropy (parent)}}-\overbrace {\mathrm {H} (T|A)} ^{\text{Weighted Sum of Entropy (Children)}}}

{\displaystyle =-\sum \_{i=1}^{J}p\_{i}\log \_{2}{p\_{i}}-\sum \_{a}{p(a)\sum \_{i=1}^{J}-\Pr(i|a)\log \_{2}{\Pr(i|a)}}}

That is, the expected information gain is the mutual information, meaning that on average, the reduction in the entropy of T is the mutual information.

Information gain is used to decide which feature to split on at each step in building the tree. Simplicity is best, so we want to keep our tree small. To do so, at each step we should choose the split that results in the most consistent child nodes. A commonly used measure of consistency is called [information](https://en.wikipedia.org/wiki/Information_theory) which is measured in [bits](https://en.wikipedia.org/wiki/Bit). For each node of the tree, the information value "represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given that the example reached that node".[[19]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-Witten_2011_102%E2%80%93103-19)

Consider an example data set with four attributes: *outlook* (sunny, overcast, rainy), *temperature* (hot, mild, cool), *humidity* (high, normal), and *windy* (true, false), with a binary (yes or no) target variable, *play*, and 14 data points. To construct a decision tree on this data, we need to compare the information gain of each of four trees, each split on one of the four features. The split with the highest information gain will be taken as the first split and the process will continue until all children nodes each have consistent data, or until the information gain is 0.

To find the information gain of the split using *windy*, we must first calculate the information in the data before the split. The original data contained nine yes's and five no's.

{\displaystyle I\_{E}([9,5])=-{\frac {9}{14}}\log \_{2}^{}{\frac {9}{14}}-{\frac {5}{14}}\log \_{2}{\frac {5}{14}}=0.94}

The split using the feature *windy* results in two children nodes, one for a *windy* value of true and one for a *windy* value of false. In this data set, there are six data points with a true *windy* value, three of which have a *play* (where *play* is the target variable) value of yes and three with a *play* value of no. The eight remaining data points with a *windy* value of false contain two no's and six yes's. The information of the *windy*=true node is calculated using the entropy equation above. Since there is an equal number of yes's and no's in this node, we have

{\displaystyle I\_{E}([3,3])=-{\frac {3}{6}}\log \_{2}^{}{\frac {3}{6}}-{\frac {3}{6}}\log \_{2}^{}{\frac {3}{6}}=-{\frac {1}{2}}\log \_{2}^{}{\frac {1}{2}}-{\frac {1}{2}}\log \_{2}^{}{\frac {1}{2}}=1}

For the node where *windy*=false there were eight data points, six yes's and two no's. Thus we have

{\displaystyle I\_{E}([6,2])=-{\frac {6}{8}}\log \_{2}^{}{\frac {6}{8}}-{\frac {2}{8}}\log \_{2}^{}{\frac {2}{8}}=-{\frac {3}{4}}\log \_{2}^{}{\frac {3}{4}}-{\frac {1}{4}}\log \_{2}^{}{\frac {1}{4}}=0.81}

To find the information of the split, we take the weighted average of these two numbers based on how many observations fell into which node.

{\displaystyle I\_{E}([3,3],[6,2])=I\_{E}({\text{windy or not}})={\frac {6}{14}}\cdot 1+{\frac {8}{14}}\cdot 0.81=0.89}

Now we can calculate the information gain achieved by splitting on the *windy* feature.

{\displaystyle IG({\text{windy}})=I\_{E}([9,5])-I\_{E}([3,3],[6,2])=0.94-0.89=0.05}

To build the tree, the information gain of each possible first split would need to be calculated. The best first split is the one that provides the most information gain. This process is repeated for each impure node until the tree is complete. This example is adapted from the example appearing in Witten et al.[[19]](https://en.wikipedia.org/wiki/Decision_tree_learning#cite_note-Witten_2011_102%E2%80%93103-19)