Decision Trees in AIMA, WEKA, and SCIKIT-LEARN
Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 233 data sets as a service to the machine learning community. You may view all data sets through our searchable interface. Our old web site format. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy.

Supported By: 

In Collaboration With: 

Latest News:
- 2010-03-01: Note from donor regarding Netflix data
- 2009-10-16: Two new data sets have been added.
- 2009-09-14: Several data sets have been added.
- 2008-07-23: Repository mirror has been set up.
- 2008-03-24: New data sets have been added!
- 2007-06-25: Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope
- 2007-04-13: Research papers that cite the repository have been associated to specific data sets.

Featured Data Set: Yeast

Task: Classification
Data Type: Multivariate
# Attributes: 8
# Instances: 1484

Predicting the Cellular Localization Sites of Proteins
Zoo Data Set

Download: Data Folder, Data Set Description

Abstract: Artificial, 7 classes of animals

http://archive.ics.uci.edu/ml/datasets/Zoo

<table>
<thead>
<tr>
<th>Data Set Characteristics:</th>
<th>Multivariate</th>
<th>Number of Instances:</th>
<th>101</th>
<th>Area:</th>
<th>Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Characteristics:</td>
<td>Categorical, Integer</td>
<td>Number of Attributes:</td>
<td>17</td>
<td>Date Donated</td>
<td>1990-05-15</td>
</tr>
<tr>
<td>Associated Tasks:</td>
<td>Classification</td>
<td>Missing Values?:</td>
<td>No</td>
<td>Number of Web Hits:</td>
<td>18038</td>
</tr>
</tbody>
</table>
Zoo training data

101 Instances

animal name: string
1) hair: Boolean
2) feathers: Boolean
3) eggs: Boolean
4) milk: Boolean
5) airborne: Boolean
6) aquatic: Boolean
7) predator: Boolean
8) toothed: Boolean
9) backbone: Boolean
10) breathes: Boolean
11) venomous: Boolean
12) fins: Boolean
13) legs: {0,2,4,5,6,8}
14) tail: Boolean
15) domestic: Boolean
16) catsize: Boolean
17) type: {mammal, fish, bird, shellfish, insect, reptile, amphibian}

category label

aardvark, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 4, 0, 0, 1, **mammal**

antelope, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 0, 1, **mammal**

bass, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, **fish**

bear, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 0, 0, 1, **mammal**

boar, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 1, 0, 1, **mammal**

buffalo, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 0, 1, **mammal**

calf, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 1, 1, 1, **mammal**

carp, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, **fish**

catfish, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, **fish**

cavy, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 4, 0, 1, 0, **mammal**

cheetah, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 4, 1, 0, 1, **mammal**

chicken, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 2, 1, 1, 0, **bird**

chub, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, **fish**

clam, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, **shellfish**

crab, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, **shellfish**

...
Zoo example

>>> from learning4e import *

>>> zoo

<DataSet(zoo): 101 examples, 18 attributes>

>>> zdt = DecisionTreeLearner(zoo)

>>> zdt(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0])  # eggs=1

'fish'

>>> zdt(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0])  # eggs=0

'mammal'
Zoo example

>> zdt

DecisionTree(13, 'legs', {0: DecisionTree(12, 'fins', {0: DecisionTree(8, 'toothed', {0: 'shellfish', 1: 'reptile'}), 1: DecisionTree(3, 'eggs', {0: 'mammal', 1: 'fish'})}, 2: DecisionTree(1, 'hair', {0: 'bird', 1: 'mammal'}), 4: DecisionTree(1, 'hair', {0: DecisionTree(6, 'aquatic', {0: 'reptile', 1: DecisionTree(8, 'toothed', {0: 'shellfish', 1: 'amphibian'})}, 1: 'mammal'}), 5: 'shellfish', 6: DecisionTree(6, 'aquatic', {0: 'insect', 1: 'shellfish'}), 8: 'shellfish'})

AIMA’s decision tree representation difficult for people to interpret
Zoo example

>>> zt.display()
Test legs
    legs = 0 ==> Test fins
    fins = 0 ==> Test toothed
            toothed = 0 ==> RESULT =  shellfish
            toothed = 1 ==> RESULT =  reptile
    fins = 1 ==> Test eggs
            eggs = 0 ==> RESULT =  mammal
            eggs = 1 ==> RESULT =  fish
    legs = 2 ==> Test hair
            hair = 0 ==> RESULT =  bird
            hair = 1 ==> RESULT =  mammal
    legs = 4 ==> Test hair
            hair = 0 ==> Test aquatic
            aquatic = 0 ==> RESULT =  reptile
            aquatic = 1 ==> Test toothed
            toothed = 0 ==> RESULT =  shellfish
            toothed = 1 ==> RESULT =  amphibian
            hair = 1 ==> RESULT =  mammal
    legs = 5 ==> RESULT =  shellfish
    legs = 6 ==> Test aquatic
            aquatic = 0 ==> RESULT =  insect
            aquatic = 1 ==> RESULT =  shellfish
    legs = 8 ==> RESULT =  shellfish

Better, but still difficult for people to understand
Zoo example

legs = 0 ==> Test fins
    fins = 0 ==> Test toothed
        toothed = 0 ==> RESULT = shellfish
        toothed = 1 ==> RESULT = reptile
    fins = 1 ==> Test milk
        milk = 0 ==> RESULT = fish
        milk = 1 ==> RESULT = mammal

legs = 1 ==> Test hair
    hair = 0 ==> RESULT = bird
    hair = 1 ==> RESULT = mammal

legs = 2 ==> Test hair
    hair = 0 ==> Test aquatic
        aquatic = 0 ==> RESULT = reptile
        aquatic = 1 ==> Test toothed
            toothed = 0 ==> RESULT = shellfish
            toothed = 1 ==> RESULT = amphibian
    hair = 1 ==> RESULT = mammal

legs = 3 ==> RESULT = shellfish

legs = 4 ==> RESULT = shellfish

legs = 5 ==> RESULT = shellfish

legs = 6 ==> Test aquatic
    aquatic = 0 ==> RESULT = insect
    aquatic = 1 ==> RESULT = shellfish

legs = 8 ==> RESULT = shellfish

After adding the shark example to the training data & retraining
['shark',0,0,0,0,0,1,1,1,0,0,1,0,1,0,0, 'fish']
Weka

• Open-source Java machine learning tool
• Implements many classifiers & ML algorithms
• Uses common data representation format; easy to try different ML algorithms and compare results
• Comprehensive set of data pre-processing tools and evaluation methods
• Three modes of operation: GUI, command line, Java API
Classifier output

J48 pruned tree

HowCrowded = None: No (2.0)
HowCrowded = Some: Yes (4.0)
HowCrowded = Full
  | Hungry = Yes
  |   | IsFridayOrSaturday = Yes
  |   |   | Price = $ : Yes (2.0)
  |   |   | Price = $$ : Yes (0.0)
  |   | IsFridayOrSaturday = No: No (1.0)
  | Hungry = No: No (2.0)

Number of Leaves : 7
Size of the tree : 11

Time taken to build model: 0.11 seconds
Common .arff* data format

% Simplified data for predicting heart disease with just six variables
% Comments begin with a % allowed at the top
@relation heart-disease-simplified
@attribute age numeric
@attribute sex { female, male }
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina {no, yes}
@attribute class {present, not_present}
@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...

Training data

*ARFF = Attribute-Relation File Format
Weka demo

The workbench for machine learning

Weka is tried and tested open source machine learning software that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It is widely used for teaching, research, and industrial applications, contains a plethora of built-in tools for standard machine learning tasks, and additionally gives transparent access to well-known toolboxes such as scikit-learn, R, and Deeplearning4j.

https://cs.waikato.ac.nz/ml/weka/
Install Weka

• Download and install Weka
  – Requires Java
• cd to your weka directory
• Invoke the GUI interface or call components from the command line
  – You may want to set environment variables (e.g., CLASSPATH) or aliases (e.g., weka)
Getting your data ready

• Our class [code repo](#)’s [ML](#) directory has several data files for the restaurant example
  1. [restaurant.csv](#): original data in simple text format
  2. [restaurant.arff](#): data put in Weka’s [arff](#) format
  3. [restaurant_test.arff](#): more data for test/evaluation
  4. [restaurant_predict.arff](#): new data we want predictions for using a saved model

• #1 is the raw training data we’re given
• #2 is an arff version of #1
• We’ll train and save a model with #2
• Test it with #3
• Predict target on new data with #4
Open Weka app

- cd /Applications/weka
- java -jar weka.jar
- Apps optimized for different tasks
- Start with Explorer
Explorer Interface
Starts with Data Preprocessing; open file to load data
Load restaurant.arff training data
We can inspect/remove features
Select: classify > choose > trees > J48
Adjust parameters

Note command line like syntax

Change parameters here
Select the testing procedure.
See training results

Classifier output

- HowCrowded = None: No (2.0)
- HowCrowded = Some: Yes (4.0)
- HowCrowded = Full
  - Hungry = Yes
    - IsFridayOrSaturday = Yes
      - Price = $: Yes (2.0)
      - Price = $$: Yes (0.0)
      - Price = $$$: No (1.0)
    - IsFridayOrSaturday = No: No (1.0)
  - Hungry = No: No (2.0)

Number of Leaves: 1

Size of the tree: 11

Time taken to build model: 0.03 seconds

Evaluation on test set:

Correctly Classified Instances: 3
Correctly Classified Instances: 0
Incorrectly Classified Instances: 0
Incorrectly Classified Instances: 0
Kappa statistic: 1
Mean absolute error: 0
Root mean squared error: 0
Relative absolute error: 0
Root relative squared error: 0
Total Number of Instances: 3

Detailed Accuracy By Class:

- Correctly Classified Instances: 3 (100%)
- Incorrectly Classified Instances: 0 (0%)

Status:
OK
Compare results

HowCrowded = None: No (2.0)
HowCrowded = Some: Yes (4.0)
HowCrowded = Full
   |   Hungry = Yes
   |   |   IsFridayOrSaturday = Yes
   |   |   |   Price = $: Yes (2.0)
   |   |   |   Price = $$: Yes (0.0)
   |   |   |   Price = $$$: No (1.0)
   |   |   |   IsFridayOrSaturday = No: No (1.0)
   |   Hungry = No: No (2.0)

J48 pruned tree: nodes:11; leaves:7, max depth:4
ID3 tree: nodes:12; leaves:8, max depth:4

The two decision trees are equally good
scikit-learn

• Popular open source ML and data analysis tools for Python
• Built on NumPy, SciPy, and matplotlib for efficiency
• However, decision tree tools are a weak area
  – E.g., data features must be numeric, so working with restaurant example requires conversion
  – Perhaps because DTs not used for large problems
• We’ll look at using it to learn a DT for the classic iris flower dataset
50 samples from each of three species of Iris (setosa, virginica, versicolor) with four data features: length and width of the sepals and petals in centimeters
from sklearn import tree, datasets
import graphviz, pickle

iris = datasets.load_iris()

clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
	pickle.dump(clf, open('iris.pkl', 'wb'))
tree.export_graphviz(clf, out_file="iris.pdf")
Weka vs. scikit-learn vs. ...

• Weka: good for experimenting with many ML algorithms
  – Other tools are more efficient & scalable
• Scikit-learn: popular and efficient suite of open-source machine-learning tools in Python
  – Uses NumPy, SciPy, matplotlib for efficiency
  – Preloaded into Google’s Colaboratory
• Custom apps for a specific ML algorithm are often preferred for speed or features