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 is still available, for those who prefer the old 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. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians. We have also set up a mirror site for the Repository.

Supported By: In Collaboration With: Rexa.info

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

Newest Data Sets:
- 2012-10-21: QtyT4010D100K
- 2012-10-19: Legal Case Reports
- 2012-09-29: seeds
- 2012-08-30: Individual household electric power consumption
- 2012-08-15: Nohthix
- 2012-08-06: PAMAP2 Physical Activity Monitoring
- 2012-08-04: Restaurant & consumer data
- 2012-08-03: CNAE-9

Most Popular Data Sets (hits since 2007):
1. Iris
2. Adult
3. Wine
4. Breast Cancer Wisconsin (Diagnostic)
5. Car Evaluation
6. Abalone
7. Poker Hand
8. Forest Fires
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>animal name</th>
<th>hair</th>
<th>feathers</th>
<th>eggs</th>
<th>milk</th>
<th>airborne</th>
<th>aquatic</th>
<th>predator</th>
<th>toothed</th>
<th>backbone</th>
<th>breathes</th>
<th>venomous</th>
<th>fins</th>
<th>legs</th>
<th>tail</th>
<th>domestic</th>
<th>catsize</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>aardvark</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>antelope</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>bass</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>fish</td>
<td></td>
</tr>
<tr>
<td>bear</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>boar</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>buffalo</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>calf</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>carp</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>catfish</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
<td></td>
</tr>
<tr>
<td>chicken</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>bird</td>
<td></td>
</tr>
<tr>
<td>chub</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>fish</td>
<td></td>
</tr>
<tr>
<td>clam</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>shellfish</td>
<td></td>
</tr>
<tr>
<td>crab</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>shellfish</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Zoo example

```python
aima-python> python
>>> from learning import *
>>> zoo
<DataSet(zoo): 101 examples, 18 attributes>
>>> dt = DecisionTreeLearner()
>>> dt.train(zoo)
>>> dt.predict(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0])
'fish'
>>> dt.predict(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0])
'mammal'
```
Evaluation methodology (1)

Standard methodology:

1. Collect large set of examples with correct classifications
2. Randomly divide collection into two disjoint sets: *training* and *test*
3. Apply learning algorithm to training set giving hypothesis H
4. Measure performance of H w.r.t. test set
Evaluation methodology (2)

• Important: keep the training and test sets disjoint!
• Study efficiency & robustness of algorithm: repeat steps 2-4 for different training sets & training set sizes
• On modifying algorithm, restart with step 1 to avoid evolving algorithm to work well on just this collection
Evaluation methodology (3)

Common variation on methodology:

1. Collect set of examples with correct classifications
2. Randomly divide it into two disjoint sets: development & test; further divide development into devtrain & devtest
3. Apply algorithm to devtrain giving hypothesis H
4. Measure performance of H wrt devtest set
5. Modify approach, repeat 3-4 as needed
6. Final test on test data
Zoo evaluation

train_and_test(learner, data, start, end) uses data[start:end] for test and the rest for train

```python
>>> dtl = DecisionTreeLearner
>>> train_and_test(dtl(), zoo, 0, 10)
1.0
>>> train_and_test(dtl(), zoo, 90, 100)
0.80000000000000004
>>> train_and_test(dtl(), zoo, 90, 101)
0.81818181818181823
>>> train_and_test(dtl(), zoo, 80, 90)
0.90000000000000002
```
K-fold Cross Validation

- Problem: getting *ground truth* data expensive
- Problem: Need different test data each time we test
- Problem: experiments needed to find right *feature space* & parameters for ML algorithm
- Goal: minimize training+test data needed
- Idea: split training data into K subsets, use K-1 for *training*, and one for *development testing*
- Repeat K times and average performance
- Common K values are 5 and 10
Zoo evaluation

cross_validation(learner, data, K, N) does N iterations, each time randomly selecting 1/K data points for test, rest for train

>>> cross_validation(dtl(), zoo, 10, 20)
0.95500000000000007

leave1out(learner, data) does len(data) trials, each using one element for test, rest for train

>>> leave1out(dtl(), zoo)
0.97029702970297027
Learning curve

- **Learning curve**: % correct on test set as function of training set size
- Steeper curves are better
Zoo

```python
>>> learningcurve(DecisionTreeLearner(), zoo)
[(2, 1.0), (4, 1.0), (6, 0.98333333333333339), (8,
  0.97499999999999998), (10, 0.94000000000000006), (12,
  0.908333333333333321), (14, 0.98571428571428577), (16,
  0.9375), (18, 0.94999999999999996), (20,
  0.94499999999999995), ... (86, 0.78255813953488373), (88,
  0.73636363636363644), (90, 0.707777777777777795)]
```
Iris Data Set

Download: Data Folder, Data Set Description

Abstract: Famous database; from Fisher, 1936

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

<table>
<thead>
<tr>
<th>Data Set Characteristics:</th>
<th>Multivariate</th>
<th>Number of Instances:</th>
<th>150</th>
<th>Area:</th>
<th>Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Characteristics:</td>
<td>Real</td>
<td>Number of Attributes:</td>
<td>4</td>
<td>Date Donated</td>
<td>1988-07-01</td>
</tr>
<tr>
<td>Associated Tasks:</td>
<td>Classification</td>
<td>Missing Values?:</td>
<td>No</td>
<td>Number of Web Hits:</td>
<td>386237</td>
</tr>
</tbody>
</table>

Source:
Iris Data

• Three classes: Iris Setosa, Iris Versicolour, Iris Virginica

• Four features: sepal length and width, petal length and width

• 150 data elements (50 of each)

AIMA-Python> more data/iris.csv
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5.0,3.6,1.4,0.2,setosa

http://code.google.com/p/aima-data/source/browse/trunk/iris.csv
Comparing ML Approaches

• The effectiveness of ML algorithms varies depending on the problem, data and features used
• You may have intuitions, but run experiments
• Average accuracy (% correct) is a standard metric

```python
>>> compare([DecisionTreeLearner, NaiveBayesLearner, NearestNeighborLearner], datasets=[iris, zoo], k=10, trials=5)

iris     zoo
DecisionTree 0.86   0.94
NaiveBayes   0.92   0.92
NearestNeighbor 0.85   0.96
```
A **confusion matrix** can be a better way to show results.

For binary classifiers it’s simple and is related to **type I and type II errors** (i.e., false positives and false negatives).

There may be different costs for each kind of error.

So we need to understand their frequencies.

<table>
<thead>
<tr>
<th></th>
<th>actual</th>
<th>predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C</strong></td>
<td>True positive</td>
<td>False positive</td>
</tr>
<tr>
<td><strong>~C</strong></td>
<td>False negative</td>
<td>True negative</td>
</tr>
</tbody>
</table>
Confusion Matrix (2)

- For multi-way classifiers, a confusion matrix is even more useful.
- It lets you focus in on where the errors are.

<table>
<thead>
<tr>
<th>predicted</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cat</td>
</tr>
<tr>
<td>Cat</td>
<td>5</td>
</tr>
<tr>
<td>Dog</td>
<td>2</td>
</tr>
<tr>
<td>Rabbit</td>
<td>0</td>
</tr>
</tbody>
</table>
Accuracy, Error Rate, Sensitivity, Specificity

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples that are correctly classified
  \[ \text{Accuracy} = \frac{TP + TN}{All} \]

- **Error rate**: \(1 - \text{accuracy}\), or
  \[ \text{Error rate} = \frac{FP + FN}{All} \]

**Class Imbalance Problem:**
- One class may be *rare*, e.g. fraud, HIV-positive, ebola
- Significant *majority in negative class* & rest in positive class

**Sensitivity**: True Positive recognition rate
  - \(\text{Sensitivity} = \frac{TP}{P} \)

**Specificity**: True Negative recognition rate
  - \(\text{Specificity} = \frac{TN}{N} \)

---

<table>
<thead>
<tr>
<th>P/A</th>
<th>C</th>
<th>¬C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>¬C</td>
<td>FN</td>
<td>TN</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>N</td>
</tr>
</tbody>
</table>
On Sensitivity and Specificity

• High sensitivity: few false negatives
• High specificity: few false positives
• TSA security scenario:
  metal scanners set for high sensitivity and low specificity (e.g., trigger on keys) to reduce risk of missing dangerous objects
Precision and Recall

Information retrieval uses similar measures, **precision & recall**, to characterize retrieval effectiveness.

- **Precision**: % of tuples classifier labels as positive that are actually positive
- **Recall**: % of positive tuples classifier labels as positive

\[
\text{precision} = \frac{TP}{TP + FP}
\]

\[
\text{recall} = \frac{TP}{TP + FN}
\]
Precision and Recall

- In general, increasing one causes the other to decrease
- Studying the precision recall curve is informative
If one system’s curve is always above the other, it’s better.
F1 measure

• We often want just one measure to compare two systems
• F1 measure combines both into a useful single metric
• It’s the harmonic mean of precision & recall

\[ F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]
Precision at N

• Ranking tasks return a set of results ordered from best to worst
  – E.g., documents about “barack obama”
  – Types for “Barack Obama”

• **Learning to rank** systems can do this using a variety of algorithms (including SVM)

• **Precision at K** is the fraction of top K answers that are correct