CPSC 405 Data mining

Decision Tree Technical Manual

© Slippery Rock University

1 Morrow Way

Slippery Rock, PA 16057

Table of Contents

[Basics 1](#_Toc436828589)

[Algorithm 1](#_Toc436828590)

[Node 1](#_Toc436828591)

[Structure 1](#_Toc436828592)

[Tree 2](#_Toc436828593)

[Initializing 2](#_Toc436828594)

[Initializing the DataList 2](#_Toc436828595)

[Running the Algorithm 4](#_Toc436828596)

Section

1

Basics

This package is used to implement decision tree algorithms to

## Algorithm

* Algorithm –
* Entropy –
* Gini –

## Node

* Leaf –
* Node –
* NominalDecisionNode –
* OrdinalDecisionNode –

## Structure

* Attribute – Container for the attributes of a data column. Contains values for the name, type, min, max, mean, and stdDev. If the column is of type Categorial, it also contains an DMArrayList of CategorialPoints and the number of UniqueValues.
* CategorialPoint – A container to store UniqueValues and the number of instances within an attribute.
* DataList – Highest level container. This contains three DMArrayLists containing the headers, each row(Stored as a DataPoint), and attributes. Also contains the classification.
* DataPoint – A container to store each row of data. Contains a separate DMArrayList containing each of the values associated with that row, as well as a classification identifier.
* DMArrayList – A custom ArrayList used to hold all data. Contains custom functions.
* FrequencyTable –
* MathFunctions – Contains a number of functions to calculate the min, max, mean, and stdDev of a DMArrayList.

## Tree

* DecisionTree –

Initializing

To use the Algorithms, you must first load the data into a DataList. This is done by first instantiating a new DataList Object. By default, this new DataList contains no values, with no classifications, but instantiates the three DMArrayLists within. Data can then be added by using the readFile(String filename) method. The file to be read must be in csv format. The DataList will then automatically calculate the Headers, Attributes, and then fill the DataPoints DMArrayList with data.

## Initializing the DataList

After the DataList is initialized and has read the file, several methods can be called.

* trim(int) – Trims the data by the set number
* getLength() – returns the number of columns
* getNumRows() – returns the number of rows
* getHead(int) – returns the Header for the specified column
* getDataPoints() – Returns the DMArrayList of DataPoints
  + getDataVal(int) – Returns the DataValue of the specified column
* getHeaders() – Returns the DMArrayList of Headers
* getAttributes() – Returns the DMArrayList of Attributes
  + getName()
  + getType()
  + getMin()
  + getMax()
  + getMean()
  + getStdDev()
  + getUniqueVals()
  + getData() – Returns a DMArrayList containing Categorial Points. Each point contains a name, and the number of times that it occurs within that column. getName() and getCount() and be used to access these.
* removeRow(int) – Removes the specified row
* removeColumn(int) – Removes the specified column

After the data has been customized to your liking, you must set a main class using the setClass(int) method. This class will then be set to the main class for all the data.

The DataList class contains methods for exporting the new data set.

* writeToCSV(String Filename) – exports the headers and data rows as a csv file.
* attributeToCsv(String Filename) – exports the Attributes as a csv file. The headers for this file are the variables contained in Attributes(name, type, min…etc) This file lists all of the Attributes that are available.

The DataList class contains methods to create two new instances of a DataList containing a training set and a test set. These methods return an Object array, Object[0] is the training set, Object[1] is the test set.

* everyOther() – Shuffles all data points, then returns two sets of equal size.
* randomShuffle(float) – Shuffles all data points, then returns two sets. The parameter for this method is the percentage that the training set will consume. EX: If 20 is passed as the percent, the training set will contain 20% of the data and the test set will contain 80%.

These two sets will then be used as input for the DataMining algorithm.

Running the Algorithm