Riaan Van Onselen- 97124275

31005 Assignment 2

ID3 Implementation

Practical Data Mining Project

Table of Contents

[Introduction: 2](#_Toc495161213)

[Program Usage: 2](#_Toc495161214)

[Program design: (30%): 2](#_Toc495161215)

[Java Object Explanations: 2](#_Toc495161216)

[Main: 2](#_Toc495161217)

[DataParser 3](#_Toc495161218)

[DataDescriptor 3](#_Toc495161219)

[DataElement 3](#_Toc495161220)

[DataPreprocessor 3](#_Toc495161221)

[ID3 3](#_Toc495161222)

[Node 3](#_Toc495161223)

[Output of program (20%): 3](#_Toc495161224)

[Appendix: 3](#_Toc495161225)

[Clarity and readability (30%): 3](#_Toc495161226)

[Quality of recorded presentation (20%): 3](#_Toc495161227)

# Introduction:

This report will present an implementation of the ID3 classification algorithm developed by Ross Quinlan in Java to create a decision tree classifier for a mushroom data set. The objective of the classifier is to determine if a given mushroom is either “edible” or “poisonous” through the comparison of categorical data describing the mushrooms attributes. The implementation was completed using five Java classes and an external command line argument parsing library called Docopt. This report will cover the programs usage and design as well as the output of the final implementation.

https://www.cise.ufl.edu/~ddd/cap6635/Fall-97/Short-papers/2.htm

https://en.wikipedia.org/wiki/ID3\_algorithm

# Program Usage:

# Program design: (30%):

The following sections will describe the design of the implementation of the ID3 algorithm. The implementation was completed using five public class files and an inner class to provide the core functionality of the program. An additional library called Docopt was used to create the command line interaction allowing for easy use of the program.

## Java Object Explanations:

### Main Class:

This is the main class of the program and provides the starting method for execution. When the program starts, the main method is called and the arguments are passed in from the command line interface. This class makes use of the external library called Docopt to categorise the arguments that are passed in as well as to provide the user with usage instructions when an incorrect launch attempt is detected.

Once the arguments are passed in and stored in a Map data structure, the main method determines what functionality of the program is desired and store the conditions in Booleans. As the training data set is the base requirement of running the program, the main method extracts the data set from the file passed in using the data parser object. If the options for a test data set and prediction data set are active, the program will extract the data set from the corresponding csv files. All data sets are stored in Arraylists for quick access to each individual element. If the flag for binarising the data is set, the program then converts all the data sets into binarised data sets using the data preprocessor class. Once all the data has been converted, a decision tree is created using the ID3 class by passing in a data set list. This data set can be either the original data set extracted from the CSV files or the binarised data set. Using the ID3 object, the main class will pass in data to be tested or predicted. Finally, depending on the options provided files can be created for the decision tree structure, test analysis data or a list of predicted class values.

### DataParser Class:

This class provides the methods for extracting a data set from a CSV file. Once the object is created the public method “parseData” is used to create a list of data elements and a data descriptor from the given file. The training file is assumed to have every possible unique value for each attribute. To Increase efficacy the program, the data descriptor is built only once and used to encode all extra CSV files. There are getter methods for both the data descriptor and data set list to allow access outside of the class.

### DataDescriptor Class:

This class acts as a description for the stored data set. It contains two lists; One for all the attribute names and one for all the unique values for each attribute. These lists are used to encode the categorical data, from the CSV files, into integers instead of strings. This allows for faster comparisons between values as well as a faster means of counting each data element with the same value. This class provides also methods to convert an array of integer values to an array of string values for printing out information.

### DataElement Class:

This class acts as an individual element of data from the data set. Each row from the CSV files is converted into a data element which store an integer array of values that correspond to the categorical data. The data element class also provides methods to print out an individual element in either integer values or string categorical data like the CSV.

### DataPreprocessor Class:

This class provides the methods to convert

### ID3 Class:

### Node Inner Class:

Library

<https://github.com/docopt/docopt.java>

5 classes

<https://dreampuf.github.io/GraphvizOnline/>

1. a short description of the design of the program (3 pages maximum);
2. the source code for your program;
3. transcript of output from your program.

# Output of program (20%):

# Appendix:

# Clarity and readability (30%):

# Quality of recorded presentation (20%):