

Kutztown University

Research Commons at Kutztown University

Computer Science and Information Technology Faculty Computer Science and Information Technology Department

5-24-2021

Visualizing Decision Trees and Forests using Radial Trees

Angela Kozma

Kutztown University of Pennsylvania, akozm527@live.kutztown.edu

Follow this and additional works at: <https://research.library.kutztown.edu/cisfaculty>



Part of the [Data Science Commons](#), and the [Graphics and Human Computer Interfaces Commons](#)

Recommended Citation

Master's Thesis for completion of Master of Science in Computer Science at Kutztown University of Pennsylvania

This Research paper is brought to you for free and open access by the Computer Science and Information Technology Department at Research Commons at Kutztown University. It has been accepted for inclusion in Computer Science and Information Technology Faculty by an authorized administrator of Research Commons at Kutztown University. For more information, please contact czerny@kutztown.edu.

Visualizing Decision Trees and Forests using Radial Trees

A Thesis

Presented to the Faculty of

The Department of Computer Science and Information Technology

Kutztown University of Pennsylvania

Kutztown, Pennsylvania

In Partial Fulfillment

Of the Requirements of the Degree

Master of Science

By

Angela Kozma

May 2021

APPROVAL TO SUBMIT THESIS

This thesis presented by Angela Kozma entitled
Visualizing Decision Trees and Forests using Radial Trees

Is hereby approved:

Approved: May 24, 2021

Date

Dale E. Pearson

Adviser

May 24, 2021

Date

John M. Ward

Second Reader & Chair, Department of CS & IT

May 25, 2021 John Ward

Date

Dean, Graduate Studies

Abstract

Data visualization has become a big representation of many company's data and schedules. Now people are not using just simple bar graphs and pie charts in business meetings but utilizing other fields of study and even more complex graphs. By using multiple visualizations to display their results and projects, it is letting more outside people understand what they are working on and can lead to more viewpoints on the topic being displayed. Also, schedules for projects are now being displayed visually so the workers can see how much time each part of their project is going to take.

With this increase in visualization, decision trees are now starting to become visualized. Decision trees zero in on object classification and find a way to label or group those objects. In this paper, complex decision trees that can be hard to understand for everyone will be visualized using radial trees. The program will take the advantages that radial trees offer for data and create an interactive display for users of decision trees and forests.

Acknowledgments

I wish to express my appreciation to all those who have made this thesis possible. Especially my thesis advisor, Dr. Dale Parson, who has suggested the topic and has guided my work with patience, understanding, and skill.

Besides my thesis advisor, I would like to thank Dr. Lisa Frye for taking the time to remediate my thesis paper.

Also, my sincere thanks go to Dr. Daniel Spiegel, who has always been available for advisement and guidance in the Computer Science and Information Technology Master Program.

To my family, I would also like to thank you for all the support and motivation throughout the whole process.

Tables of Contents

Abstract.....iii

Acknowledgments.....iv

List of Tables and Figures.....vii

Chapter 1.....1

 1. Introduction.....1

Chapter 2.....3

 2. Background.....3

 2.1. What Are Radial Trees?.....3

 2.2. What Are Decision Trees and Forests.....9

 2.3. What Is Data Visualization?.....11

 2.4. What Is Weka?.....12

 2.5. What Is Processing?.....14

Chapter 3.....17

 3. Procedure17

 3.1. Visualization Sample Data Overview.....17

 3.2. Data from Weka to Processing.....17

 3.3. Data Transformation.....19

Chapter 4.....20

 4. Design and Implementation Results.....20

 4.1. TreeNode Setup and Results.....20

4.2. Keyboard Functions.....	21
v	
4.3. Mouse Functions.....	22
4.4. How Everything is Connected.....	22
Chapter 5.....	26
5. Conclusion.....	26
Chapter 6.....	27
6. Future Work.....	27
Bibliography.....	28
Appendix A, Code.....	30
WekaTreeVisualizer.pde.....	30
TreeNode.pde.....	34
CartesianPolar.pde.....	39
Appendix B, Data.....	43
numAndNomsToNominal.arff.....	43
Appendix C, Parsing Trees into Processing.....	46
HoeffdingTree.....	46
REPTree.....	48
J48.....	53
RandomTree.....	62

List of Tables and Figures

Figure 1 - Radial Tree.....4

Figure 2 - Old Figurative Tree.....5

Figure 3 - New Figurative Tree.....5

Figure 4 - Vertical Tree.....6

Figure 5 - Horizontal Tree.....7

Figure 6 - Multidirectional Tree.....8

Figure 7 - Hyperbolic Tree.....9

Figure 8 - Decision Tree Breakdown.....10

Figure 9 - Weka Start Screen.....13

Figure 10 - Weka’s Screen with a Dataset Showing a Histogram of pH Value Ranges.....14

Figure 11 - Processing IDE Screen with Code Loaded In.....15

Figure 12 - A Portion of RandomTree’s Tree for Predicting OxygenMgPerLiter.....18

Figure 13 - HoeffdingTree Displayed.....23

Figure 14 - HoeffdingTree Displayed with Depth of 2.....23

Figure 15 - REPTree Displayed.....24

Figure 16 - REPTree Displayed with Depth of 5.....24

Chapter 1

1. Introduction

Visualization has always been an important part of learning throughout history. Many people can picture what is being described in their heads but with some subjects and information, it can be a hassle or practically impossible. One form where it is difficult to visualize what is being conveyed is decision trees and forests. These expressions can get lengthy and complicated so expressing them differently can have a bigger impact on understanding.

Data science analytics is closely linked to data visualization in the process of inspecting, cleansing, transforming, and modeling data. There are two conflicting goals in data analysis: the accuracy of prediction by a model and the intelligibility of a model. For this research, the primary goal is to improve readability (intelligibility) without decreasing the accuracy of the decision trees being studied.

Since visualization can expand one's understanding, we will figure out a technique for improving the 10,000-foot view of the information obtained of the decision trees and forests, while providing a means for expanding details through interactive navigation.

Our research is tightly coupled with using data obtained from Weka [9] and Processing [4] for visualization, therefore, getting familiar with some of the terms, such as decision trees and forests, radial trees, and the two programs mentioned above would help understand the context of the research. Decision trees are a way to classify or label objects by zeroing on the classifications and forests are multiple randomized decision trees [8]. Combining multiple and random decision trees can reduce the effects of overfitting the data. Radial trees, or radial map, is displaying a

data tree that expands outward in the radial direction (forms into a circle) [10]. Weka is a program that houses a collection of machine learning algorithms and it is used for solving real-world data mining problems, whereas Processing is a software sketchbook that produces visual arts.

The United States Geological Survey (USGS) data on Pennsylvania's streams from 2012, which has been data-mined on Weka, will be used for this study. This data was used as a class project to see if there was any valuable information. It produces a lot of information when using the decision trees and forests; some were hard to read due to the length. Using the outcome of the algorithms from Weka, the information was then imported into Processing, which turns that data into a radial tree with interactive navigation. There are limitations because this study is only testing both Weka and Processing.

Chapter 2

2. Background

2.1. What Are Radial Trees?

Trees have been an important religious symbol for multiple types of cultures throughout history, but they now also have another significance which is describing and organizing human knowledge [6]. Even though tree diagrams have lost some of the “traditional” and “lifelike” features, trees are now used more widely with different types of uses with new features added to them. One of the trees that break from the lifelike features of trees is radial trees.

A radial tree, see Figure 1, also known as radial map, “is a method of displaying a tree structure in such a way that expands outward in the radial direction” [10]. This type of tree tends to be weighted if there is a vertex R , the root. The root is the main vertex of the tree where all vertices connect to in some way, and it is at depth zero. Measuring each vertex from R helps show what the depth number is and the larger number the farther it is from the root [1].

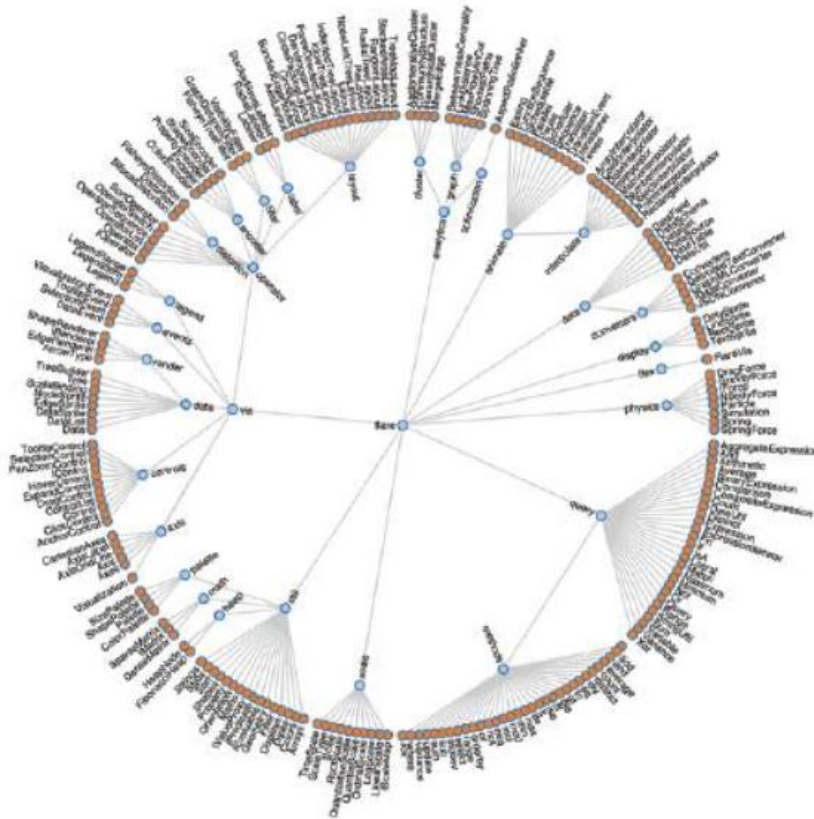


Figure 1 - Radial Tree [6]

There are multiple other forms of visual trees such as:

2.1.1. *Figurative Trees* [6]

- a. The most traditional and lifelike tree diagram was used a lot throughout history as a religious symbol for numerous cultures (see Figure 2). Even though now figurative tree diagrams lost some lifelikeness, there were features added such as roots, branches, and leaves (see Figure 3).

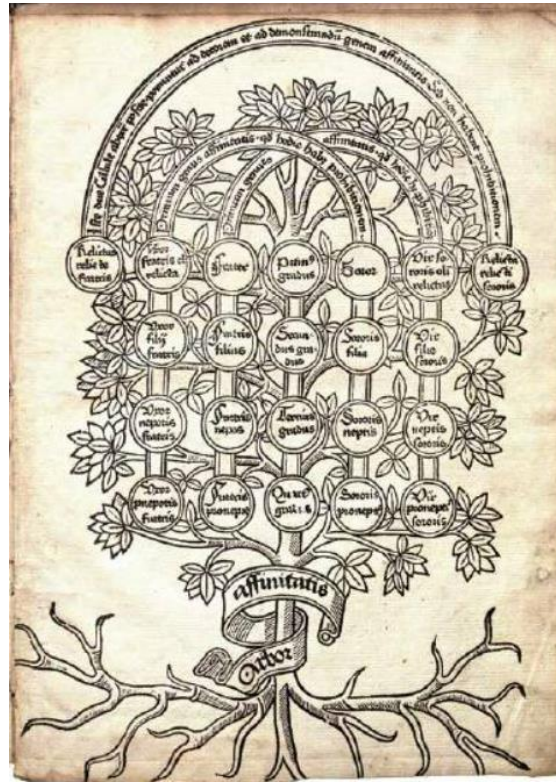


Figure 2 - Old Figurative Tree [6]

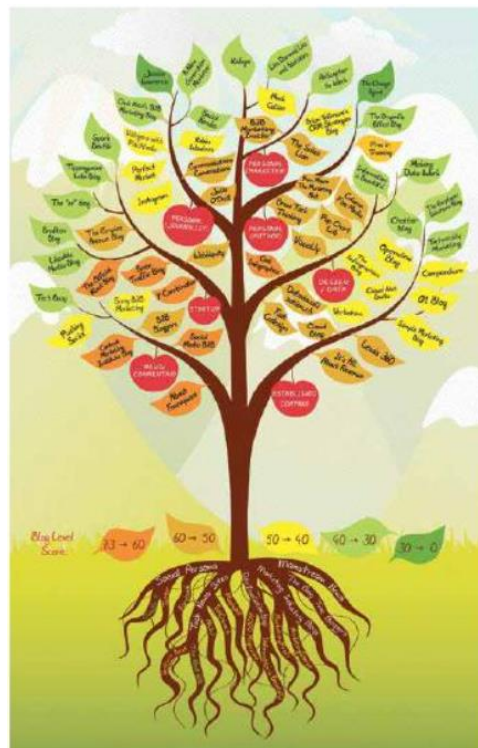


Figure 3 - New Figurative Tree [6]

2.1.2. Vertical Trees [6]

- a. A popular tree used today and is used a lot for family trees (see Figure 4). The family trees' branches are commonly represented by circles, squares, or other polygonal shapes. "Of all visualization models, vertical trees are the ones that retain the strongest resemblance to figurative trees, due to their vertical layout and forking arrangement from a central trunk. In most cases they are inverted trees, with the root at the top, emphasizing the notion of descent and representing a more natural writing pattern from top to bottom" [6].



Figure 4 - Vertical Tree [6]

2.1.3. *Horizontal Trees* [6]

- a. Horizontal trees, see Figure 5, resemble the grammatical construct of a sentence and tend to be highly efficient for archetypal models (flow charts, mind maps, etc.). The rank is most frequently from left to right and they most likely came about as an alternative to vertical trees to address spatial constraints and layout requirements.

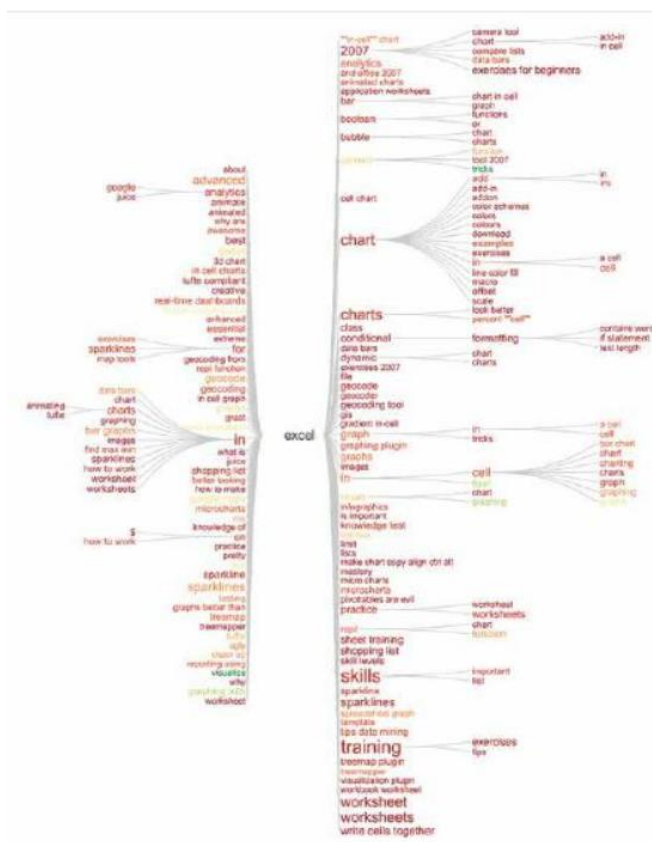


Figure 5 - Horizontal Tree [6]

2.1.4. *Multidirectional Trees* [6]

- a. Multidirectional trees' configurations are free flowing, unlike vertical or horizontal trees (see Figure 6). "From an initial root or source within the

plotted area, multidirectional trees expand toward the edges of the space, moving in distinct paths and periodically bifurcating. This leads to an organic, unconfined appearance...” [6]. Multidirectional trees are now mostly generated by computers and with this adaption it has made these trees a highly efficient node-link variant that can be used to map large hierarchies.

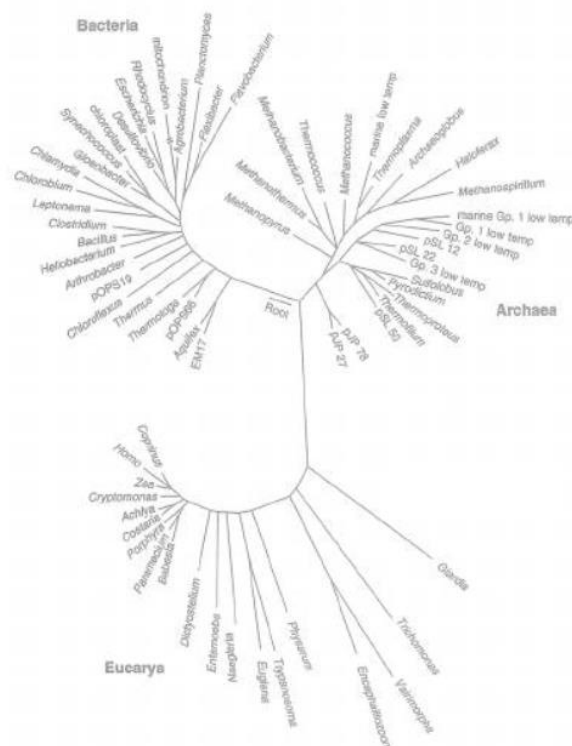


Figure 6 - Multidirectional Tree [6]

2.1.5. Hyperbolic Trees [6]

- a. The hyperbolic tree is like the radial tree, but it is a more recent, computer-aided visualization generated with advanced algorithms (see Figure 7). This type of tree treats nodes and their linkages using a “focus

and context' technique that emphasizes a given set of nodes by centering and enlarging them while giving less prominence to other dependencies, making them progressively smaller and closer to the periphery" [6].

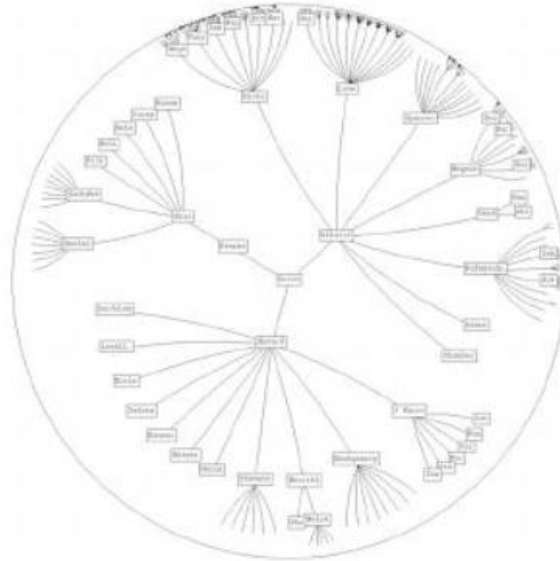


Figure 7 - Hyperbolic Tree [6]

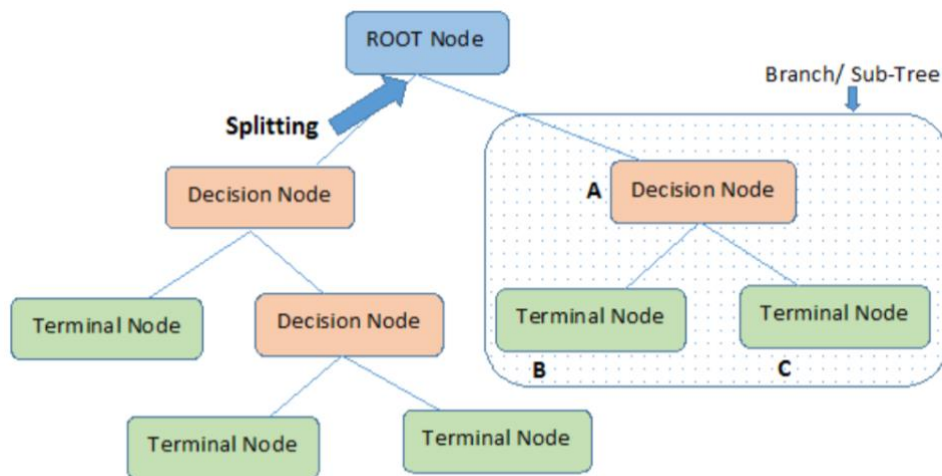
2.2. What Are Decision Trees and Forests?

Decision trees are used in machine learning algorithms and follow a set of if-else conditions to visualize the data and classify it according to the conditions. Forests for decision trees are multiple decision trees combined. Machine learning consists of self-learning programs that use pattern recognition and computational learning theory to find data patterns.

Important terminology used with decision trees are (see Figure 8) [7]:

- a. **Root node:** An attribute that is used when dividing the data into two or more sets. It is also at the beginning of a tree and it represents the entire population of nodes being analyzed.
- b. **Branch or Sub-Tree:** A portion/section of the whole decision tree.

- c. **Splitting:** Dividing a node into two or more sub-nodes based on the if-else conditions.
- d. **Decision Node:** When sub-nodes are split even further, then the node is called a decision node.
- e. **Leaf or Terminal Node:** The end of the decision tree where nodes cannot be split. For decision trees, its leaves are a class value where the tree is used for classification, whereas when the tree is used for regression, like M5P's linear regression, the leaves are numeric formulas. In this research, the work is focused on decision trees for classifications, so the leaves are class values.
- f. **Pruning:** Removing or combining the sub-nodes from the decision tree.



Note:- A is parent node of B and C.

Figure 8 - Decision Tree Breakdown [3]

2.3. What Is Data Visualization?

Data visualization is the representation of data or information through charts, graphs, maps, trees, and other visual formats. Having the data or information in a visual format provides a way to see and understand trends, outliers, and patterns in data [2]. This is not only important to data scientists and data analysts but other careers as well. If the job needs to process and understand data, then data visualization is important. Some other fields that work with the visualization of data are finance, marketing, tech, and design, to list a few.

People need data visualization because “a visual summary of information makes it easier to identify patterns and trends than looking through thousands of rows on a spreadsheet” [5]. Without the charts, graphs, etc. it would be difficult to communicate data findings and identify the patterns. If the user is looking at your data without visualizations and knowledge of how to read the data, it becomes extremely hard to interpret that data. By providing pictures it broadens the audience that can understand the data and point trying to be made.

The most popular ways to use data visualization are [5]:

a. Changes over time

- i. Displays how the data trends over some time.

b. Determining frequency

- i. Sees how often the events happen over time.

c. Determining relationships (correlations)

- i. Determine the relationship between two variables in the data that can be very difficult without visualization.

d. Examining a network

- i. Examining a network is when data visualization is used to identify the audience's target and show the connected entities corresponding to that target.

e. Scheduling

- i. Can display a visualization of a timeline or schedule for projects that can clearly illustrate each task within the project and the duration of how long it will take to complete.

f. Analyzing value and risk

- i. By color-coding, a formula of what is valuable and what is risky can greatly influence choices when seen in a visual chart or graph rather than just written down.

2.4. What Is Weka?

Weka is an open-source machine learning software that was developed at the University of Waikato, New Zealand. This software is used 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" [9].

When the Weka GUI window opens the user has five options: Explorer, Experimenter, KnowledgeFlow, Workbench, and Simple CLI (see Figure 9). Each one of these performs different tasks. The one used in this research and one to clean and test the dataset is the Explorer tab. After clicking that tab, the user can open their dataset file and Weka will organize the data

attributes and information for the user. On the opening tab, Preprocess, attributes can be added, deleted, and modified. This process helps clean the data so the user can get a more accurate analysis (see Figure 10).

In the Classify tab, the data is tested with the desired classifier. The decision tree classifiers that are used in the visual radial trees are HoeffdingTree, REPTree, J48, and RandomTree. These four classifiers were used for this research because the tree is used for classification and not regression. There are other choices that the user can pick in the ‘tree’ section of the classifier tab, but they are not needed for the decision tree that is being analyzed.

Another helpful tab in Weka for data analysis and simple visualization is the Visualize tab. This tab allows the user to see the relationship that each attribute has with the other attributes through graphs. This helps show patterns and can be read easily.

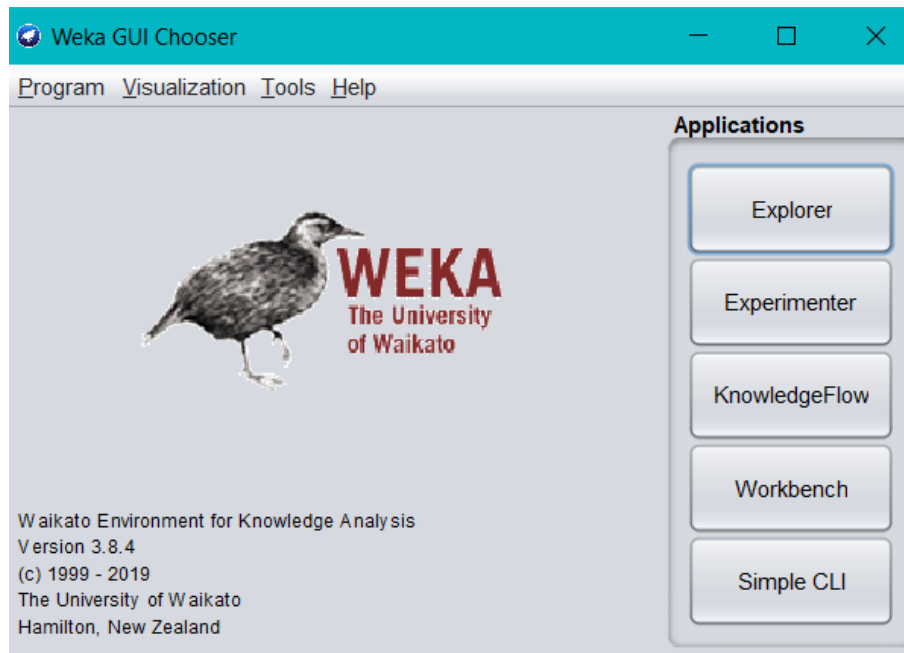


Figure 9- Weka Start Screen

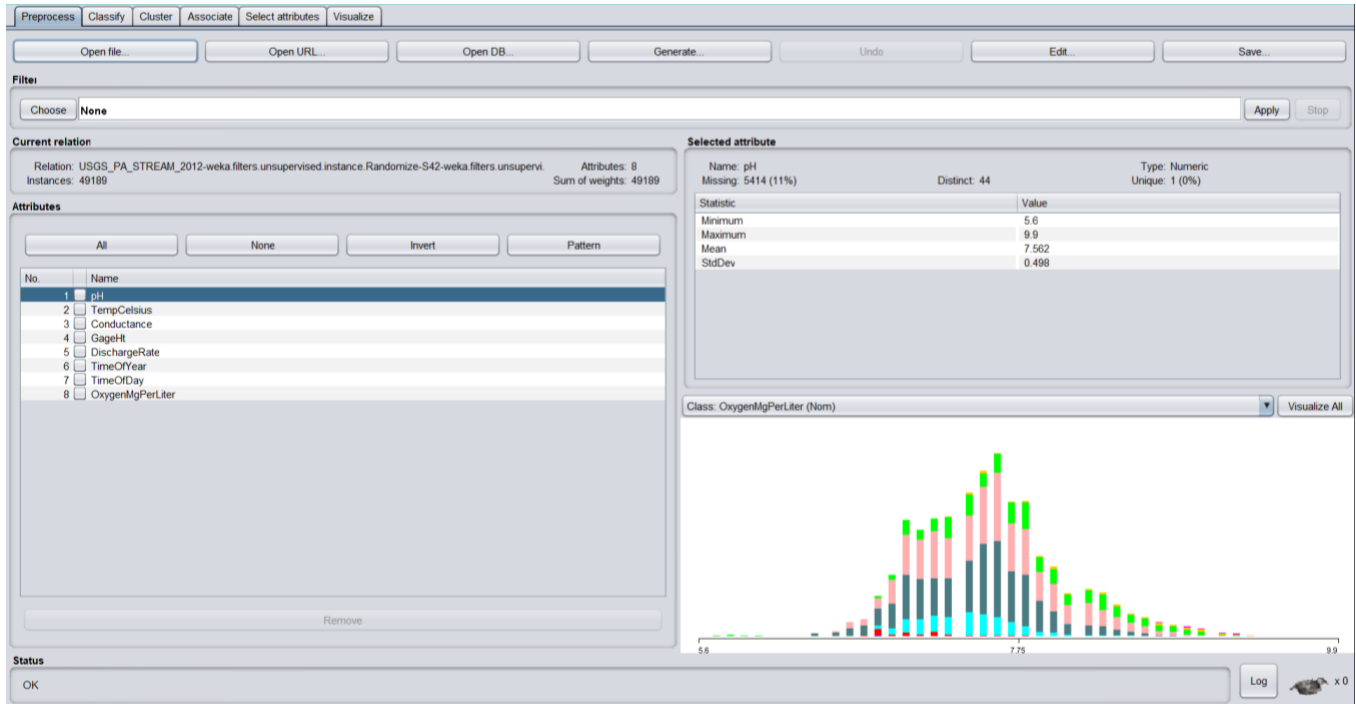
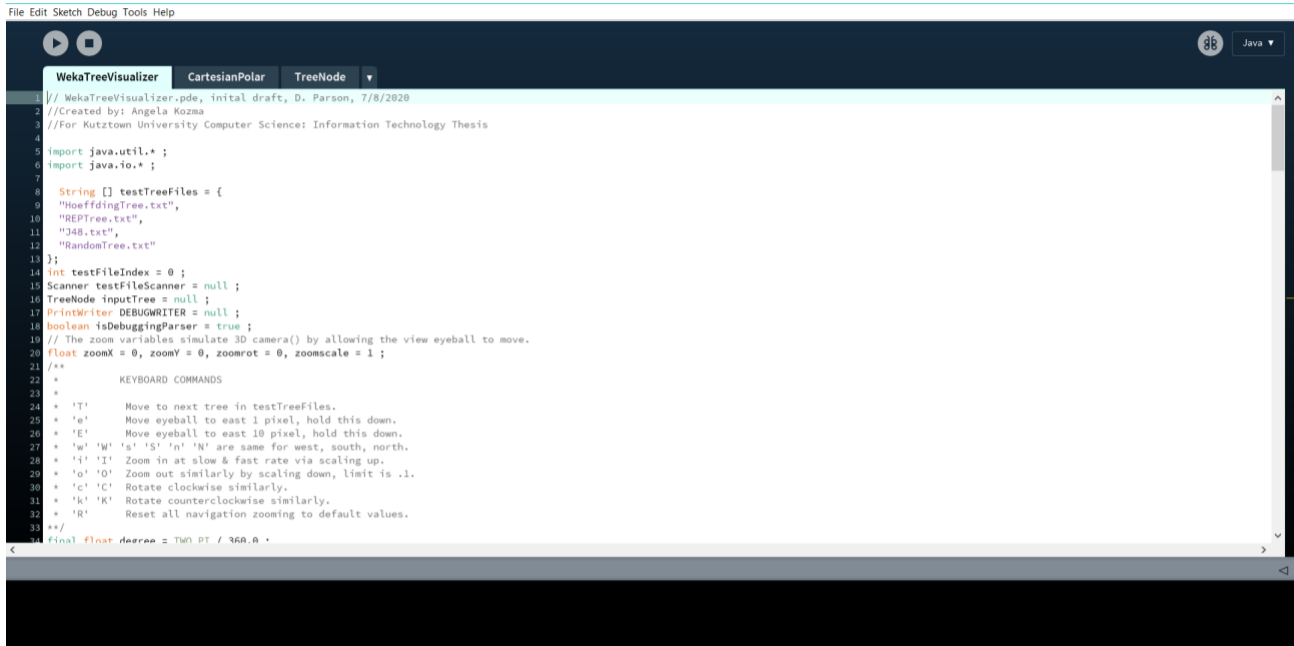


Figure 10- Weka's Screen with a Dataset Showing a Histogram of pH Value Ranges

2.5. What Is Processing?

Processing is a free graphical library and integrated development environment (IDE) built for electronic arts. This software sketchbook is open source that can run on the Mac, Windows, and GNU/Linux platforms [4]. Processing uses the language Java with additional simplifications and has a graphical user interface that also simplifies the compilation and execution stage. Due to the additional simplifications, Processing can be used to teach non-programmers the fundamentals of computer programming in a visual context.



```

1 // WekaTreeVisualizer.pde, initial draft, D. Parson, 7/8/2020
2 //Created by: Angela Kozma
3 //For Kutztown University Computer Science: Information Technology Thesis
4
5 import java.util.* ;
6 import java.io.* ;
7
8 String [] testTreeFiles = {
9   "HoeffdingTree.txt",
10  "REPTree.txt",
11  "J48.txt",
12  "RandomTree.txt"
13 };
14 int testFileIndex = 0 ;
15 Scanner testFileScanner = null ;
16 TreeNode inputTree = null ;
17 PrintWriter DEBUGWRITER = null ;
18 boolean isDebuggingParser = true ;
19 // The zoom variables simulate 3D camera() by allowing the view eyeball to move.
20 float zoomX = 0, zoomY = 0, zoomrot = 0, zoomscale = 1 ;
21 /**
22  * KEYBOARD COMMANDS
23  *
24  * 'T' Move to next tree in testTreeFiles.
25  * 'e' Move eyeball to east 1 pixel, hold this down.
26  * 'E' Move eyeball to east 10 pixel, hold this down.
27  * 'w' 'W' 's' 'S' 'n' 'N' are same for west, south, north.
28  * 'I' 'i' Zoom in at slow & fast rate via scaling up.
29  * 'O' 'o' Zoom out similarly by scaling down, limit is .1.
30  * 'c' 'C' Rotate clockwise similarly.
31  * 'k' 'K' Rotate counterclockwise similarly.
32  * 'R' Reset all navigation zooming to default values.
33 **/
34 final float DEGREE = TWO_PI / 360.0 ;

```

Figure 11- Processing IDE Screen with Code Loaded In

The figure above, Figure 11, is an example of the Processing IDE. Users can add as many or few tabs for their project and can choose a language of their choice in the drop-down on the top right of the screen. Processing also supplies the users with a debug function to help those whose codes are not working. Another useful tool is the color selector, which is located under the Tools tab. Since the user will have to insert the number form of the color they want to use into the code, this tool lets the user either pick the color from a wheel or by the number representation of that color. Also, at the bottom of the figure, there is a black screen that will display the print statements that are throughout the code when it runs. To run and stop the code there are the top two buttons that are displayed as the ‘play’ and ‘stop’ buttons.

Since this software can be used by non-programmers a lot of students can use this program in school; however, Processing is not just limited to students, visual designers, artists, and architects also create projects using this software. Although it is easy to use the platform it

can create simple to complex visuals, and some projects that have been created using Processing have been featured at the Museum of Modern Art in New York, the Victoria and Albert Museum in London, the Centre Georges Pompidou in Paris, and many other locations.

Using Processing is a good fit for this research because of the vastness that can be created using it. With having such a wide range of abilities to be able to create anything, it becomes easier to upload data from Weka and to create an interactive radial tree.

Chapter 3

3. Procedure

3.1. Visualization Sample Data Overview

The data used from the United States Geological Survey (USGS) on Pennsylvania's streams from 2012 was the main source of data used to receive the trees' algorithm outcome [See Appendix B]. This was a good dataset to use because it gives complex and long trees that would be hard to visualize with all the different conditions for each node. An example of this is when using the RandomTree classifier function in Weka which produces a RandomTree that would take up at least five pages to display it all. Also, this data was a good choice to use because there was little cleaning that needed to be done to the dataset. Before trying to convert the results of the trees into a visual radial tree, there can be duplicates in data that can lead to confusion for viewers and the decision trees can be even simpler to read if cleaned properly. Also, if there is a specific part in the dataset that the user wants to focus on more than the other parts, then by eliminating the parts not needed, the point of the visualization can be expressed.

3.2. Data from Weka to Processing

When the data is all cleaned and the decision trees and forests' data has been collected, the user will have to copy and paste the information in a simple file, such as a text file. Other information that the user wants in the file can be added but the main information that is needed is the tree information.

```

RandomTree
=====

TempCelsius < 14.55
|  pH < 7.75
|  |  Conductance < 88.5
|  |  |  TempCelsius < 7.35
|  |  |  |  TempCelsius < 4.45
|  |  |  |  |  GageHt < 3.45
|  |  |  |  |  |  DischargeRate < 73.5
|  |  |  |  |  |  |  TempCelsius < 1.75
|  |  |  |  |  |  |  |  TimeOfDay = morning
|  |  |  |  |  |  |  |  |  GageHt < 0.88 : '(13.54-15.78]' (5/0)
|  |  |  |  |  |  |  |  |  GageHt >= 0.88
|  |  |  |  |  |  |  |  |  |  pH < 7.05 : '(11.3-13.54]' (3/0)
|  |  |  |  |  |  |  |  |  |  pH >= 7.05
|  |  |  |  |  |  |  |  |  |  |  TempCelsius < 1.5 : '(13.54-15.78]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  TempCelsius >= 1.5 : '(11.3-13.54]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  TimeOfDay = afternoon
|  |  |  |  |  |  |  |  |  |  |  |  pH < 7
|  |  |  |  |  |  |  |  |  |  |  |  |  Conductance < 36.5 : '(13.54-15.78]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  Conductance >= 36.5 : '(11.3-13.54]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  pH >= 7 : '(11.3-13.54]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  TimeOfDay = evening
|  |  |  |  |  |  |  |  |  |  |  |  |  |  Conductance < 36.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  TempCelsius < 0.4 : '(11.3-13.54]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  TempCelsius >= 0.4 : '(13.54-15.78]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  Conductance >= 36.5 : '(11.3-13.54]' (3/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  TimeOfDay = night : '(11.3-13.54]' (9/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  TempCelsius >= 1.75
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  GageHt < 0.84 : '(11.3-13.54]' (241/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  GageHt >= 0.84
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  DischargeRate < 52.5 : '(11.3-13.54]' (115/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  DischargeRate >= 52.5
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  TimeOfDay = morning : '(11.3-13.54]' (1/0)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  TimeOfDay = afternoon
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  Conductance < 54.5 : '(11.3-13.54]' (2/0)

```

Figure 12 - A Portion of RandomTree's Tree for Predicting OxygenMgPerLiter

The figure above, Figure 12, shows an example of part of the data tree's information. The whole tree must be saved in a file to be used for the visualization [See Appendix C]. After the file is saved, it must be transferred to where the code for creating the radial tree in Processing is stored. Without doing this step the data cannot be read.

Once the file is with the code, in the code itself there is a declaration for a string array called testTreeFiles. The names of the files being used to create the radial tree are in quotes, within the curly brackets.

3.3. Data Transformation

The text files that contain the appropriate data needed is read through a try statement to sketch the path of the data and to make sure the files have all the needed information. This process is done in the setup function by setting up the size of the screen and the size of the nodes. Next, with the trees' data stored in a variable, the other code is used to set up the radial tree [See Appendix A].

Chapter 4

4. Design and Implementation Results

4.1 TreeNode Setup and Results

To set up the radial tree, there must be a place to store all the data for each node and where to place the nodes. This class, `TreeNode`, consists of the variables:

1. **String test:** holds the “labels” of what each node is
2. **TreeNode [] children:** an array that empties for the leaves of the tree (the last nodes that do not have any nodes continuing from it)
3. **String leaf:** classifies as null for non-leaves
4. **Final int depth:** how far away the node is from the root node
5. **Float radius:** set to = -1 to start and is computed by the `layoutTree` function per polar geometry. Polar geometry uses radius and angles as an alternative to (x, y) 2D coordinates.
6. **Float angleInRadians:** set to = -1 to start so the coordinates to start the circle is at (0, 0) at the center-right, which is 1.0, and two Pi is a full circle
7. **Float areaInRadians:** set to = -1 and is the size of the sub-arc
8. **Float X, Y:** displays the x and y coordinate location
9. **Int physx, physy:** set to = 0 and updates screen coordinates, the physical location, within the display function

The other function in the `TreeNode` class is the `layoutTree` function. This does a recursive walk down through the tree, with the initial root at a polar coordinate, radius = 0 (center of the

radial tree), $\text{angleInRadians} = 0$ (at the center of the screen), and $\text{areaInRadians} = \text{two Pi}$ (how much of the display is available). Then at each sub-root, it splits the remaining areaInRadians by the number of children. This all creates the foundation of the radial tree.

While creating the layout of the radial tree, the code goes and finds the tree height. Finding the height using the `findTreeHeight` function will return the max height. Important information to receive is: the depth of a node, which is the number of edges from the root to the node, the height of a node, which is the number of edges from the node to the deepest leaf, and the height of a tree, which is the height of the root.

To put all the functions together is the display function. This is where the text of the nodes, the circles around the text, and the lines connecting all of the nodes are formed. There are also a couple of other functions in `TreeNode`, which are used for debugging, scanning, and reading through the data to use for the layout.

4.2. Keyboard Functions

To provide adequate visualization for the radial tree, keyboard commands were added.

1. **'T'**: Move to the next tree in `testTreeFiles`
2. **'n', 'e', 's', 'w'**: Moves the view to the north, east, south, or west 1 pixel, and the key can be held down to move the view faster
3. **'N', 'E', 'S', 'W'**: Moves the view to the north, east, south, or west 10 pixels, and the key can be held down to move the view faster
4. **'i', 'I'**: Zoom in at a slow rate (lower case) and fast rate via scaling up
5. **'o', 'O'**: Zoom out similarly to above but it scales down instead, limit is 0.1
6. **'c', 'C'**: Rotate the view clockwise

7. 'k', 'K': rotate the view counterclockwise
8. 'R': Reset all navigation and zooming to the default values

4.3. Mouse Functions

To create an interactive visualization, mouse functions were added to the code. This allows the user to be able to click a node and the screen will show all the nodes starting from the root to the node that was click's depth. The mousePressed function is used to verify that when the mouse was clicked that it was over a node. If the mouse was not over a node when clicked, then nothing happens, but if the mouse was clicked on a node, then the user is brought back to the draw function.

4.4. How Everything is Connected

To connect all the code that was written together is the result of the draw function. It is here where the display is fully drawn. It also calls the TreeNode class so the created radial tree can be visible. Also, when the mouse pressed a node, it is in this function that deals with changing the view so the user can see the node clicked and all the other nodes of that depth (see Figures 13-16).

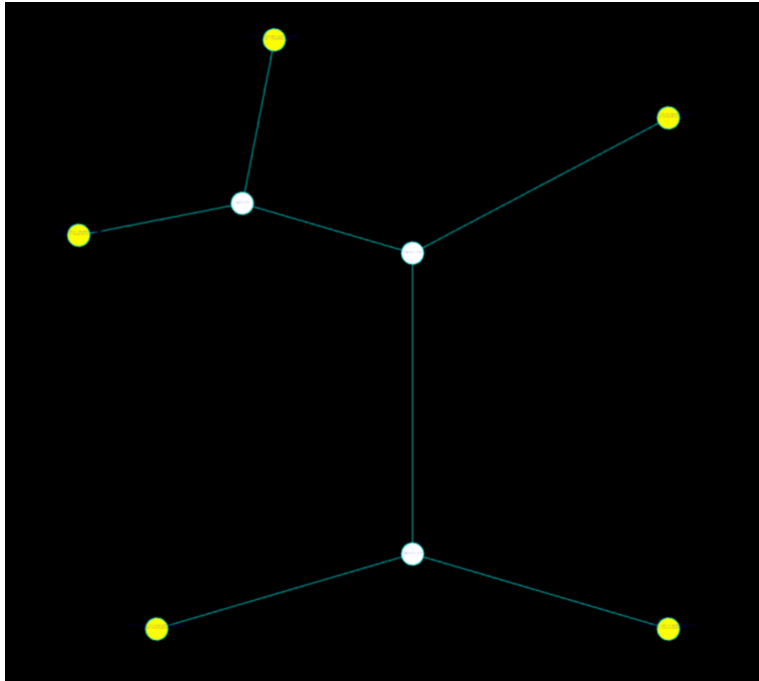


Figure 13 - HoeffdingTree Displayed

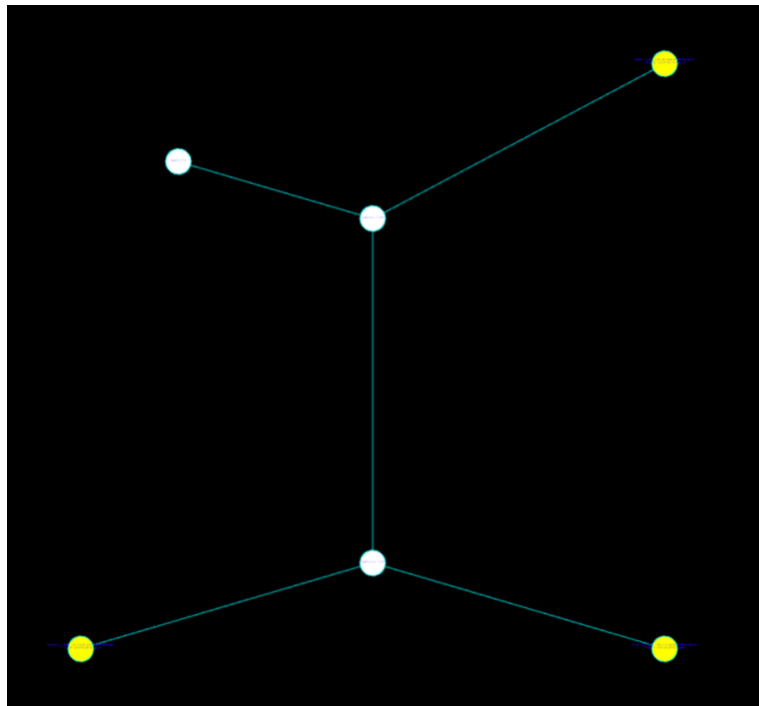


Figure 14 - HoeffdingTree Displayed with Depth of 2

By clicking on a node that was one depth away from the root node (represented as depth 1), the display shows all the nodes that were the same distance away.

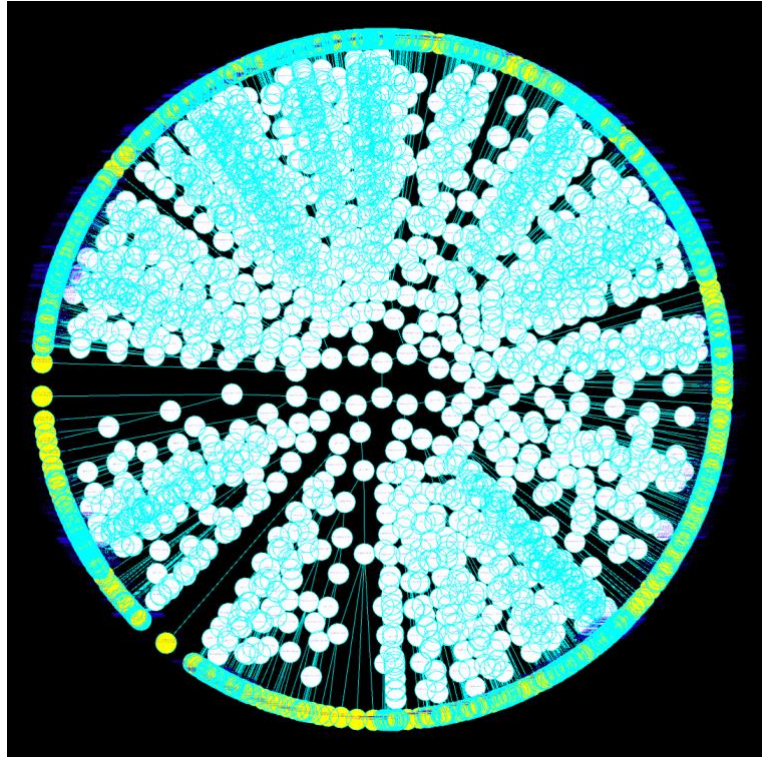


Figure 15 - REPTree Displayed

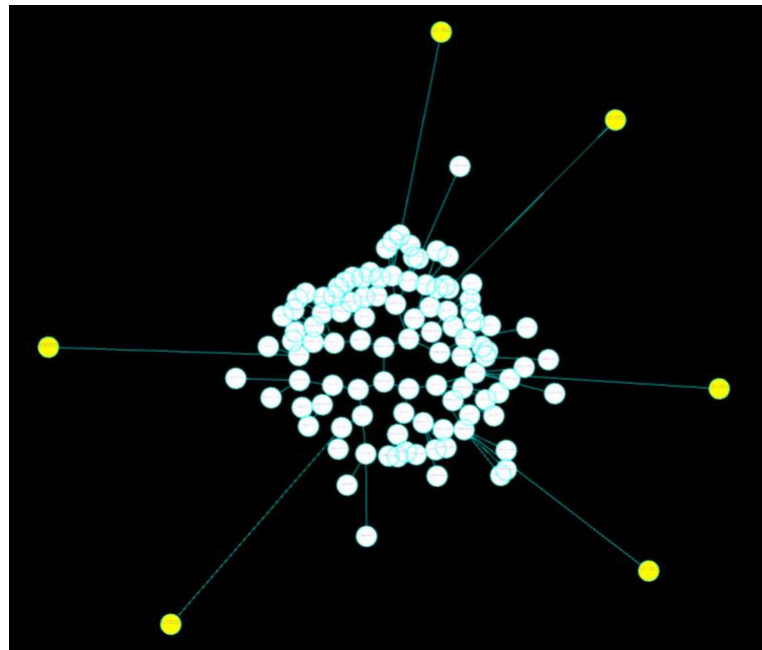


Figure 16 - REPTree Displayed with Depth of 5

By clicking on a node that was five depths away from the root node (represented as depth 5), the display shows all the nodes that were the same distance away. In this figure, there are even leaf nodes displayed, which are represented as the yellow circles.

Chapter 5

5. Conclusion

Decision trees and forests can be hard to picture and some of them can be so complex that it would take time to fully see how everything is broken down and connected. With the help of visualization, the trees can be seen easily and almost effortlessly.

Using a radial tree to show the tree structures is well suited for decision trees because it is a compressed view with a large amount of data. If a horizontal or vertical tree diagram was chosen, the visualization would not fit on a computer screen and the user would have to scroll to see the whole design. It is important to be able to see everything at once; it can lead to fewer problems and better understanding.

In the radial tree diagram created with Processing, there is node overlap, which does make the appearance look messy. However, this can be fixed at a later time. Another problem with the program is when the user is looking at all the nodes on the screen and picks a higher depth node, the processing time is long, which delays the visualization. The screen does eventually display all the nodes from the root to the one it picks, so the problem is just improving the processing rate. Though there are some minor problems with the radial tree, it still readily displays the data with the added interactive features.

Chapter 6

6. Future Work

The future for data visualization for decision trees and forests using radial trees is just the beginning. In this research, there is still a lot that can be improved on. As mentioned above, the nodes overlap when the decision tree is large and when a large tree is loaded into Processing, the run time is significantly longer when trying to move the screen around or when clicking a node. Another thing that can be modified is customizing the text size of the nodes for each tree so that when a small tree is displayed the text is bigger and the tree is zoomed in more, and when the tree is bigger the text is smaller and the tree is zoomed out to display the whole tree. Also, the research can be expanded to the third dimension instead of staying 2D. This can add a new depth and visual to radial trees and could also help declutter the bigger trees. In this research, the focus was mainly on decision trees, but more research could be made and tested on decision forests.

The drawback in continuing this research could be that a large amount of code could be rewritten regarding the improvements, depending on the approach taken for future work. Also, if the research is tested by making the radial tree 3D, the developer must make sure not to deviate too far from a radial tree.

There are some drawbacks in this work and there is still a lot that can improve visualizing decision trees and forests as radial trees; however, the work presented here is a great start to improving the visualization of decision trees using radial trees.

Bibliography

- [1] Carlson, R. (2000). *Nonclassical Sturm-Liouville problems and Schrodinger operators on radial trees* *. <https://digital.library.txstate.edu/bitstream/handle/10877/9056/2000-Carlson.pdf?sequence=1&isAllowed=y>.
- [2] *Data visualization beginner's guide: a definition, examples, and learning resources*. Tableau. (n.d.). <https://www.tableau.com/learn/articles/data-visualization>.
- [3] *Decision Tree*. Pathmind. (n.d.). <https://wiki.pathmind.com/decision-tree>.
- [4] Foundation, P. (n.d.). *Overview. A short introduction to the Processing software and projects from the community*. Back to the Processing cover. <https://processing.org/overview/>.
- [5] Import.io. (2019, October 28). *What is Data Visualization and Why Is It Important?* Import.io. <https://www.import.io/post/what-is-data-visualization/>.
- [6] Lima, M. (n.d.). *The Book of Trees: Visualizing Branches of Knowledge*. <https://ebookcentral.proquest.com/lib/kutztown-ebooks/reader.action?docID=3387599>.
- [7] Team, T. A. I. (2021, February 23). *Decision Trees Explained With a Practical Example*. Towards AI - The Best of Tech, Science, and Engineering. <https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53>.
- [8] VanderPlas, J. (n.d.). *In-Depth: Decision Trees and Random Forests*. In-Depth: Decision Trees and Random Forests | Python Data Science Handbook. <https://jakevdp.github.io/PythonDataScienceHandbook/05.08-random-forests.html>.
- [9] WEKA. Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (n.d.). <https://www.cs.waikato.ac.nz/ml/weka/>.

[10] ★ *Radial tree - graph drawing .. Info: About: What's This.* en.google. (n.d.).

<https://amp.en.google-info.org/28502338/1/radial-tree.html>.

Appendix A

Code

WekaTreeVisualizer.pde

The main class in the code, where all the data is uploaded, sets up the visuals and draws the radial trees by passing through the functions given.

```
// WekaTreeVisualizer.pde, initial draft, D. Parson, 7/8/2020
//Created by: Angela Kozma
//For Kutztown University Computer Science: Information Technology Thesis

import java.util.* ;
import java.io.* ;

String [] testTreeFiles = {
    "HoeffdingTree.txt",
    "REPTree.txt",
    "J48.txt",
    "RandomTree.txt"
};
int testFileIndex = 0 ;
Scanner testFileScanner = null ;
TreeNode inputTree = null ;
PrintWriter DEBUGWRITER = null ;
boolean isDebuggingParser = true ;
// The zoom variables simulate 3D camera() by allowing the view eyeball to move.
float zoomX = 0, zoomY = 0, zoomrot = 0, zoomscale = 1 ;
/**
 *      KEYBOARD COMMANDS
 *
 * 'T'   Move to next tree in testTreeFiles.
 * 'e'   Move eyeball to east 1 pixel, hold this down.
 * 'E'   Move eyeball to east 10 pixel, hold this down.
 * 'w' 'W' 's' 'S' 'n' 'N' are same for west, south, north.
 * 'i' 'I' Zoom in at slow & fast rate via scaling up.
 * 'o' 'O' Zoom out similarly by scaling down, limit is .1.
 * 'c' 'C' Rotate clockwise similarly.
 * 'k' 'K' Rotate counterclockwise similarly.
 * 'R'   Reset all navigation zooming to default values.
 **/
final float degree = TWO_PI / 360.0 ;
```



```

TreeNode child[];
TreeNode [] allnodes = new TreeNode [0];

int circleX, circleY;
int circleSize = width/height * 30;
boolean circleOver = false;

int posX = 0;
int posY = 0;
int cDepth;
int nheight;
int nodeMax;
String clickTest;

void setup() {
  fullScreen(P3D);
  //size(800,600,P3D);
  //size(1000,800,P3D);

  circleX = width/100 * 1/20 ; // plot everything relative to center of dome
  circleY = height/100 * 1/20 ;

  ellipseMode(CENTER);
  rectMode(CENTER);

  try {
    testFileScanner = new Scanner(new File(
      sketchPath()+"/"+testTreeFiles[testFileIndex]));
    inputTree = parseTree(testFileScanner);
    testFileScanner.close();
    if (isDebuggingParser) {
      DEBUGWRITER = new PrintWriter(new FileOutputStream(
        new File(sketchPath()+"/WekaTreeVisualizerDEBUG.txt"), false));
      DEBUGWRITER.println("\n" + testTreeFiles[testFileIndex] + ", height = "
        + inputTree.findTreeHeight(0));
      DEBUGWRITER.println(inputTree.toString() + "\n");
      DEBUGWRITER.flush();
    }
  } catch (FileNotFoundException fxx) {
    println("Cannot open: " + testTreeFiles[testFileIndex] + ": "
      + fxx.getMessage());
    exit();
  }
}

void draw() {
  textAlign(CENTER, BOTTOM);
  textSize(9);
  textMode(MODEL) ; // MODEL is default. Do NOT use SHAPE. It slows everything
  down.

```

```

push();
translate(width/2, height/2); // Display coords rotate and scale around center.
translate(zoomX, zoomY);
rotate(zoomrot);
scale(zoomscale);
background(0);
fill(255);
stroke(255);
inputTree.display();

if(circleOver == true) {
  for(TreeNode o : allnodes) {
    //println("IN if 1");
    println("nodeMax: " + nodeMax);
    println("cDepth: " + cDepth);
    //println("clickTest: " + clickTest);

    if(nodeMax != cDepth){
      //println("IN if 2");
      inputTree.display();
    }
  }
}
pollKey();
pop();
}

void mousePressed() {
  for(TreeNode n : allNodes) {
    if (n != null && dist(mouseX, mouseY, n.physx, n.physy) < circleSize/2) {
      cDepth = n.depth;
      //println("cDepth: " + n.depth);
      nheight = inputTree.findTreeHeight(inputTree.depth);
      //println("nheight: " + nheight);

      nodeMax = n.findTreeHeight(n.depth);
      clickTest = n.test;

      //fill(255, 0, 0); //red
      //rect(mouseX, mouseY, 100, 100);
      circleOver =! circleOver;
      //println("circleOver: " + circleOver);
      break ;
    }
  }
}

void pollKey() {
  if (keyPressed) {
    switch(key) {
      case 'n': // for eyeball to move up, graphics move down, which has 0 at top

```

```
    zoomY += 1 ;
    break ;
case 'N':
    zoomY += 10 ;
    break ;
case 's':
    zoomY -= 1 ;
    break ;
case 'S':
    zoomY -= 10 ;
    break ;
case 'e':
    zoomX -= 1 ;
    break ;
case 'E':
    zoomX -= 10 ;
    break ;
case 'w':
    zoomX += 1 ;
    break ;
case 'W':
    zoomX += 10 ;
    break ;
case 'i':
    zoomscale = min(zoomscale+.01, 10.0) ;
    break ;
case 'I':
    zoomscale = min(zoomscale+.05, 10.0);
    break ;
case 'o':
    zoomscale = max(zoomscale-.01, 0.1) ;
    break ;
case 'O':
    zoomscale = max(zoomscale-.05, 0.1);
    break ;
case 'c':
    zoomrot = zoomrot + degree * .1 ;
    break ;
case 'C':
    zoomrot = zoomrot + degree ;
    break ;
case 'k':
    zoomrot = zoomrot - degree * .1 ;
    break ;
case 'K':
    zoomrot = zoomrot - degree ;
    break ;
}
}
}
```

```

void keyPressed() {
  if (key == 'T') {
    // Next tree
    try {
      testFileIndex = (testFileIndex+1) % testTreeFiles.length ;
      testFileScanner = new Scanner(new File(
        sketchPath()+"/"+testTreeFiles[testFileIndex]));
      inputTree = parseTree(testFileScanner);
      testFileScanner.close();
      zoomX = 0; zoomY = 0; zoomrot = 0; zoomscale = 1 ;
      if (isDebuggingParser) {
        DEBUGWRITER.println("\n" + testTreeFiles[testFileIndex] + ", height = "
          + inputTree.findTreeHeight(0));
        DEBUGWRITER.println(inputTree.toString() + "\n");
        DEBUGWRITER.flush();
      }
    } catch (FileNotFoundException fxx) {
      println("Cannot open: " + testTreeFiles[testFileIndex] + ": "
        + fxx.getMessage());
      exit();
    }
  } else if (key == 'R') {
    zoomX = 0; zoomY = 0; zoomrot = 0; zoomscale = 1 ;
  }
}

```

TreeNode.pde

The class `TreeNode` takes the data structure for the classification tree and its recursive descent parser from a Weka output to create the decision tree into radial form.

```

// Data structure for the classification tree and its
// recursive descent parser from a Weka output.
public class TreeNode {
  String test ; // <,<=,>,>=,or = test
  TreeNode [] children ; // is empty for leaves
  String leaf ; // classification, null for non-leaves
  final int depth ; // depth of this node from top;
  float radius = -1 ; // computed by layoutTree() per polar geomtry,
  float angleInRadians = -1 ; // 0,0 at center right, 1.0,TWO_PI is full circle
  float arealInRadians = -1 ; // size of my sub-arc
  float X, Y ; // Display X,Y location, angleInRadians is rotation for
  display()
}

```

```

int physx = 0, physy = 0 ;    // Update screen coordinates (physical location) within
display()
TreeNode(String test, TreeNode [] children, String leaf, int depth) {
    this.test = test ;
    this.children = (children == null) ? (new TreeNode [0]) : children ;
    this.leaf = leaf ;
    this.depth = depth ;
}
void addChild(TreeNode child) {
    children = (TreeNode []) append(children, child);
    allnodes = (TreeNode []) append(allnodes, child);
}

/**
 * Do a recursive walk down through the tree, with the initial root at polar coordinate
 * radius = 0 (center of circle), angleInRadians = 0 (at center this does not matter),
 * arealInRadians = TWO_PI (entire display available). At each subroot it splits the
 * remaining arealInRadians by the number of children. Hopefully, the depth won't
blow out
 * the Java run-time stack.
**/
void layoutTree(float radius, float angleInRadians, float arealInRadians) {
    this.radius = radius ;
    this.angleInRadians = angleInRadians ;
    this.arealInRadians = arealInRadians ;
    float [] cartesian = polarToCartesian(radius, angleInRadians);
    int [] xy = cartesianToPhysical(cartesian[0], cartesian[1]);
    this.X = xy[0]-width/2;    // the subtraction is because we are putting 0,0
    this.Y = xy[1]-height/2;  // at center of display for rotation & scaling.
    // device rotation is in this.angleInRadians
    if (children.length > 0) {
        float areaPerChild = arealInRadians / children.length ;
        // I am in the center of my area. Make that the same for each child.
        float childAngleInRadians = angleInRadians - 0.5 * arealInRadians
            + 0.5 * areaPerChild ;
        float radiusBelowMe = 0.9 - radius ; // Do not go out all the way to the edge = 1.0
        for (TreeNode c : children) {
            int cheight = c.findTreeHeight(0);
            //float childradius = (cheight > 0) ? (radiusBelowMe / cheight) : (radiusBelowMe
/2);
            float childradius = radiusBelowMe / (cheight+1.0);
            c.layoutTree(radius+childradius, childAngleInRadians, areaPerChild);
            childAngleInRadians += areaPerChild ;
        }
    }
}

int findTreeHeight(int incomingDepth) {
    // https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html
    // The depth of a node is the number of edges from the root to the node.
    // The height of a node is the number of edges from the node to the deepest leaf.
    // The height of a tree is a height of the root.

```

```

// A full binary tree is a binary tree in which each node has exactly zero or two
children.
if (children.length == 0) {
    return incomingDepth ;
} else {
    int maxheight = incomingDepth+1 ;
    for (TreeNode c : children) {
        maxheight = max(maxheight, c.findTreeHeight(incomingDepth+1));
    }
    return maxheight ;
}
}
void display() {
    push();
    translate(X, Y);
    //rotate(angleInRadians);          //GOT RID OF WORDS BEING ROTATED
    String dtext = "" ;
    if (test != null) {
        dtext = test ;
        //println("dtext: " + dtext); //prints out the name of attributes
        for(TreeNode c : children) {
            ellipse(circleX, circleY, circleSize, circleSize);    //Makes the circles that connect
to other circles
            fill(255, 255, 255);    //White
        }
        if(leaf != null) {
            fill(255, 255, 0);        //Yellow
            ellipse(circleX, circleY, circleSize, circleSize);    //Makes all the circles
        }
        physx = round(modelX(circleX, circleY,0));
        physy = round(modelY(circleX, circleY,0));
    }
    if (leaf != null) {
        dtext += " :: " + leaf ;
        dtext += "\n" + leaf; //new experiment july 10
    }
    //NO!!! This REALLY SLOWS NAVIGATION textMode(SHAPE); // maintain vector
graphics resolution
    stroke(37,0,255);    // dark blue (using RGB)
    fill(37,0,255);
    scale(0.25); //new experiment july 10
    text(dtext, 0, 0);
    pop();
    stroke(0,255,255);    // cyan for now
/*
    stroke(0,255,255); //stroke(37,0,255);    // dark blue (using RGB)
    fill(0,255,255); // fill(37,0,255);
    scale(0.25); //new experiment july 10
    textSize(40); // Parson experiment 5/4/2021
    text(dtext, 0, -80) ; // move above circle 0);
    pop();

```

```

stroke(0,255,255); // cyan for now
*/
for (TreeNode c : children) {
    if(circleOver == true) {
        //println("tn circleOver");
        //println("tn c.depth: " + c.depth);
        //println("tn cDepth: " + cDepth);
        if(c.depth < cDepth) {
            //println("tn if");
            line(X, Y, -1, c.X, c.Y, -1);
            c.display();
        }
        if(c.depth == cDepth) {
            line(X, Y, -1, c.X, c.Y, -1);
            c.display();
        }
    }
    else {
        line(X, Y, -1, c.X, c.Y, -1);
        c.display();
    }
}

} //for loop TreeNode end bracket

}

String toString() {
    // Used for debugging printouts.
    String result = "";
    for (int i = 0 ; i < depth ; i++) {
        result += "|";
    }
    String leafpart = (leaf != null) ? (" :: " + leaf) : " ";
    result = result + test + leafpart + " (R=" + radius + " A=" + degrees(angleInRadians)
        + " areaR=" + areaInRadians + ")\n";
    for (TreeNode t : children) {
        result = result + t.toString();
    }
    return result ;
}

}

// Use a parseStack instead of recursion to avoid blowing out the
// Java run-time stack.
LinkedList<TreeNode> parseStack = new LinkedList<TreeNode>() ;

TreeNode parseTree(Scanner input) throws RuntimeException {
    parseStack.clear();
    // In a classification tree, the root has at least 2 children and no tests

```

```

// of its own. It just contains children.
TreeNode root = new TreeNode(null, null, null, 0);
parseStack.add(root);
// Now look through input until we find tree syntax.
// It requires 1 line lookahead to find the first child of root & its child.
String thisline = "", lastline = "";
while (input.hasNextLine()) {
    String tmpLine = input.nextLine().trim();
    if (tmpLine.length() < 1) {
        continue ;
    }
    lastline = thisline ;
    thisline = tmpLine ;
    if (thisline.charAt(thisline.length()-1) == ':') {
        // HoeffdingTree puts an extraneous ':' on the non-leaf nodes,
        // (most?) others only on the leaves, so get rid of it.
        thisline = thisline.substring(0,thisline.length()-1).trim();
    }
    if (thisline.charAt(0) == '|') {
        // We have scanned to the first tree data starting with |.
        // A tree with only a top level and no '|'s will not work
        // with this parser.
        TreeNode subroot = new TreeNode(lastline, null, null, 1);
        root.addChild(subroot);
        parseStack.add(subroot);
        recursiveDescent(input, thisline);
        break ; // recursiveDescent consumes remaining tree structure
    }
}
parseStack.clear(); // Recover memory.
root.layoutTree(0.0, 0.0, TWO_PI);
return root ;
}

void recursiveDescent(Scanner input, String newnodeline) {
    newnodeline = newnodeline.trim();
    while (newnodeline.length() > 0) {
        if (newnodeline.charAt(newnodeline.length()-1) == ':') {
            // HoeffdingTree puts an extraneous ':' on the non-leaf nodes,
            // (most?) others only on the leaves, so get rid of it.
            newnodeline = newnodeline.substring(0,newnodeline.length()-1).trim();
        }
        // The first blank line after a tree breaks this loop.
        int newnodeDepth = 1 ; // Start at 1 because outermost test is 1 below root.
        while (newnodeline.charAt(0) == '|') {
            newnodeDepth += 1 ;
            newnodeline = newnodeline.substring(1).trim();
        }
        String newnodeTest = null, newnodeLeaf = null ;
        int colonIndex = newnodeline.indexOf(':');
        if (colonIndex < 0) {

```



```

    newnodeTest = newnodeline ;
  } else {
    newnodeTest = newnodeline.substring(0,colonIndex).trim();
    newnodeLeaf = newnodeline.substring(colonIndex+1).trim();
  }
  TreeNode newNode = new TreeNode(newnodeTest, null, newnodeLeaf,
newnodeDepth);
  while (parseStack.size() > newnodeDepth) {
    // While this condition holds, pop nodes off of the stack.
    parseStack.removeLast();
  }
  TreeNode parent = parseStack.get(parseStack.size()-1);
  parent.addChild(newNode);
  parseStack.add(newNode);
  if (input.hasNextLine()) {
    newnodeline = input.nextLine().trim();
  } else {
    newnodeline = "";
  }
}
}
}

```

CartesianPolar.pde

Maps locations throughout the Cartesian coordinate space (-1.0, -1.0) through (1.0, 1.0) to the physical space (0, 0) through (width-1, height-1). This is to create a circle boundary for the visual so that no visual can go outside that space.

```

/**
  PhotoArchitecture2018Parson.CartesianPolar tab, D. Parson,
  adapted from SoundToWavePixels on 5/18/2018
  Copied into CSC220F19MIDlassn3 10/16/2019.
  Fixed some bugs on 11/10/2019.
  **/

/**
 * cartesianToPhysical maps a location in the Cartesian coordinate
 * space -1.0, -1.0 through 1.0, 1.0 to the physical space
 * 0, 0 through width-1, height-1. If either input coordinate
 * lies outside of the -1.0..1.0 range, cartesianToPhysical returns
 * a pair of coordinates that lie outside the central clipping
 * circle. This latter condition is not an error.
  **/

```

```

/* DEBUG VARIABLES
double debugsumx = 0.0 ;
double debugsumy = 0.0 ;
double debugcountsum = 0.0 ;
double debugminx = 0.0, debugmaxx = 0.0 ;
*/
int [] cartesianToPhysical(float cartesianX, float cartesianY) {
    int [] result = new int[2];
    int mywidth = width ; // locate centered rectangle for clipping
    int myheight = height ;
    int xoffset = 0, yoffset = 0 ;
    /* CHANGED 5/2018
    if (cartesianX < -1.0 || cartesianX > 1.0 || cartesianY < -1.0
        || cartesianY > 1.0) {
        result[0] = result[1] = -1 ;
        return result ;
    }
    */
    // Update summer 2015 cartesian is in the centered square in a rectangular display.
    if (mywidth > myheight) {
        // normally this is the case
        xoffset = (mywidth - myheight) / 2 ;
        mywidth = myheight ;
    } else {
        yoffset = (myheight - mywidth) / 2 ;
        myheight = mywidth ;
    }
    result[0] = (int)(Math.round(((cartesianX + 1.0) / 2.0) * mywidth + xoffset));
    if (result[0] == (mywidth + xoffset)) {
        result[0] -= 1 ;
    }
    result[1] = /* my */ height - (int)(Math.round(((cartesianY + 1.0) / 2.0) * myheight +
yoffset));
    // Physical requires Y == 0 to be at the top.
    if (result[1] == myheight + yoffset) {
        result[1] -= 1 ;
    }
    //debugsumx = debugsumx + result[0]; debugsumy = debugsumy + result[1] ;
    debugcountsum = debugcountsum + 1.0 ;println("DEBUG cartesianToPhysical 1, " +
width + "," + height + "    " + debugminx + "," + debugmaxx + "    " + result[0] + "," +
result[1]);
    return result ;
}

/**
 * physicalToCartesian maps a location in the physical display coordinate
 * space 0, 0 through width-1, height-1 to the Cartesian space
 * -1.0, -1.0 through 1.0, 1.0. If either input coordinate
 * lies outside of the 0, 0 through width-1, height-1 range, this function
 * returns coordinates outside the -1.0,-1.0 THRU 1.0,1.0 range.
 * The latter condition for out of range data is not an error.

```

```

**/
float [] physicalToCartesian(int physX, int physY) {
    // Update summer 2015 cartesian is in the centered square in a rectangular display.
    float [] result = new float[2];
    int mywidth = width ; // locate centered rectangle for clipping
    int myheight = height ;
    int xoffset = 0, yoffset = 0 ;
    if (mywidth > myheight) {
        // normally this is the case
        xoffset = (mywidth - myheight) / 2 ;
        mywidth = myheight ;
    } else {
        yoffset = (myheight - mywidth) / 2 ;
        myheight = mywidth ;
    }
    physX -= xoffset ; // summer 2015
    physY -= yoffset ; // summer 2015
    if (physX == mywidth) {
        physX = mywidth - 1 ; // same as cartesian 1.0
    }
    if (physY == myheight) {
        physY = myheight - 1 ; // same as cartesian 1.0
    }
    /* DROPPED 5/2018
    if (physX < 0 || physX >= mywidth || physY < 0
        || physY >= myheight) {
        result[0] = result[1] = -2.0 ;
        return result ;
    }
    */
    result[0] = ((float)(physX) / (float) mywidth * 2.0) - 1.0 ;
    result[1] = -(((float)(physY) / (float) myheight * 2.0) - 1.0) ;
    return result ;
}

/**
 * cartesianToPolar maps a location in the Cartesian coordinate
 * space -1.0, -1.0 through 1.0, 1.0 to the unit circle centered
 * at 0,0 with a radius of 1.0. The return value stores the
 * polar radius in [0] and the angle in radians in [1].
 * If either input coordinate
 * lies outside of the -1.0..1.0 range, cartesianToPolar returns
 * the corresponding results, and the calling code must check to
 * determine whether the returned radius exceeds 1.0, requiring
 * clipping.
 */
float [] cartesianToPolar(float cartesianX, float cartesianY) {
    float [] result = new float[2];
    float radius = (float) Math.sqrt(cartesianX * cartesianX + cartesianY * cartesianY);
    float angleInRadians = (float) Math.atan2(cartesianY, cartesianX);
    result[0] = radius ;

```

```

    result[1] = angleInRadians ;
    return result ;
}

/**
 * cartesianToPolar maps a location in the polar coordinate unit
 * circle 0.0, radius=0.0 upto 1.0, angle=2 * PI to the Cartesian
 * coordinates centered at 0.0,0.0. The returned result, with
 * Cartesian X in [0] and Y in [1], may lie outside the range
 * of -1.0, -1.0 through 1.0, 1.0. The caller must verify that
 * the return values are not outside the clipping boundary.
 */
float [] polarToCartesian(float radius, float angleInRadians) {
    float [] result = new float[2];
    result[0] = (float)(radius * Math.cos(angleInRadians));
    result[1] = (float)(radius * Math.sin(angleInRadians));
    return result ;
}

```

Appendix B

Data

numsAndNomsToNominal.arff

One arff file was used for this research. There are a total of 8 attributes: pH, TempCelsius, Conductance, GageHt, DischargeRate, TimeOfYear, TimeOfDay, and OxygenMgPerLiter.

These attributes all deal with what the water was like during the time of year and what day that it was tested.

https://docs.google.com/spreadsheets/d/1qINf5vMRV2hzhXNqf5ME1eR3LrBqxEZ8u8GiYVfK_gfo/edit?usp=sharing

Part of the data from numsAndNomsToNominal.arff

No	pH	TempCelsius	Conductance	GageHeight	DischargeRate	TimeOfYear	TimeOfDay	OxygenMgPerLiter
1	7.3	13.8	50		61	spring	afternoon	(9.06-11.3]'
2	7.3	8.1	73		203	spring	evening	(11.3-13.54]'
3	8.1	18.6	311		238	spring	evening	(9.06-11.3]'
4	8.2	26.1	345		162	summer	afternoon	(9.06-11.3]'
5		7.6				autumn	morning	(9.06-11.3]'
6	7.4	15.5	117		703	summer	night	(6.82-9.06]'
7	8.1	17.2	205		4290	spring	afternoon	(9.06-11.3]'
8	7.9	23.5	449		21	summer	night	(6.82-9.06]'
9	7.4	22.9	471		1090	summer	morning	(4.58-6.82]'
10	7.8	13.4	368	1.9	99	autumn	morning	(9.06-11.3]'
11	7.4	18	259		286	summer	morning	(6.82-9.06]'
12	7.1	8.4	43	0.93	14	autumn	evening	(9.06-11.3]'
13	7.6	23.5	307		376	summer	afternoon	(6.82-9.06]'
14	6.8	11.9	109		799	summer	evening	(6.82-9.06]'
15		17.3				summer	morning	(6.82-9.06]'
16	7.7	4.7	224	4.81	7860	autumn	night	(11.3-13.54]'
17		24.9		10.46		summer	evening	(6.82-9.06]'
18	9.1	31.1	976		28	summer	evening	(15.78-18.02]'
19	7.8	29.1	124		151	summer	night	(6.82-9.06]'
20	8.3	4.5	542	3.56	21	autumn	night	(11.3-13.54]'

•
•
•

49169		4.7		9.97		autumn	morning	(11.3-13.54]'
49170	7.8	13.8	469	3.59	25	autumn	night	(9.06-11.3]'
49171	7	26.5	302			summer	morning	(2.34-4.58]'
49172	7.6	17.3	340		46	spring	morning	(6.82-9.06]'
49173	7.4	13.7	288	2.65	86	autumn	evening	(9.06-11.3]'
49174	7.3	25.1	283		4410	spring	night	(4.58-6.82]'
49175	6	6	182	6.21	5310	autumn	afternoon	(11.3-13.54]'
49176	7.1	26.8	836			summer	night	(4.58-6.82]'
49177	7.7	9.7	680	3.1	20	autumn	afternoon	(11.3-13.54]'
49178	7.4	29.7	460		809	summer	evening	(4.58-6.82]'
49179	8.3	4.4	555	3.54	20	autumn	evening	(11.3-13.54]'
49180		28.1				summer	afternoon	(6.82-9.06]'
49181	7.6	13.8	359		160	autumn	afternoon	(11.3-13.54]'
49182		28		10.51		summer	afternoon	(4.58-6.82]'
49183	8.2	21.6	625		53	summer	afternoon	(11.3-13.54]'
49184	7.7	15.9	439		214	spring	night	(6.82-9.06]'
49185	8.3	21.9	788		32	spring	afternoon	(9.06-11.3]'
49186	7.6	25.9	376		110	summer	night	(4.58-6.82]'
49187	7.9	21.4	782		30	spring	night	(4.58-6.82]'
49188	7.8	17.7	432		149	spring	afternoon	(9.06-11.3]'
49189	6.9	21.4	323		2150	spring	night	(6.82-9.06]'

9								
---	--	--	--	--	--	--	--	--

Download the arff file

<https://drive.google.com/file/d/1eTtrhF0ENCNN8oWhiqKdHQc3T1HyTKVK/view?usp=sharing>

g

Appendix C

Parsing Trees into Processing

HoeffdingTree

“A Hoeffding tree (VFDT) is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time” [9]. This link below contains the information used to create the HoeffdingTree, the HoeffdingTree itself, the stratified cross-validation summary, detailed accuracy by class, and the confusion matrix.

<https://drive.google.com/file/d/1jqeMcHD5fc50IzSjwzOHYG1HJy9h6KXR/view?usp=sharing>

=== Run information ===

Scheme: weka.classifiers.trees.HoeffdingTree -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0

Relation: USGS_PA_STREAM_2012-weka.filters.unsupervised.instance.Randomize-S42-weka.filters.unsupervised.instance.RemovePercentage-P90.0-weka.filters.unsupervised.attribute.Remove-R1-4-weka.filters.unsupervised.attribute.Remove-R10-weka.filters.unsupervised.attribute.Reorder-R2-11,1-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rlast-unset-class-temporarily-weka.filters.unsupervised.attribute.Remove-R7,9-10

Instances: 49189

Attributes: 8

pH
TempCelsius
Conductance
GageHt
DischargeRate
TimeOfYear
TimeOfDay
OxygenMgPerLiter

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

TempCelsius <= 15.682:

| TempCelsius <= 9.464: '(11.3-13.54]' (3236.580) NB1 NB adaptive1

| TempCelsius > 9.464: '(9.06-11.3]' (4392.583) NB2 NB adaptive2

TempCelsius > 15.682:

| GageHt <= 9.727: '(6.82-9.06]' (686.169) NB3 NB adaptive3

| GageHt > 9.727:

| | pH <= 7.291: '(6.82-9.06]' (77.270) NB4 NB adaptive4

| | pH > 7.291: '(9.06-11.3]' (108.000) NB5 NB adaptive5

Time taken to build model: 0.76 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	29576	60.1273 %
Incorrectly Classified Instances	19613	39.8727 %
Kappa statistic	0.419	
Mean absolute error	0.1073	
Root mean squared error	0.2337	
Relative absolute error	74.5816 %	
Root relative squared error	87.1194 %	
Total Number of Instances	49189	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.800	0.001	0.386	0.800	0.520	0.555	0.895	0.613	'(-inf-2.34]'
0.043	0.002	0.263	0.043	0.074	0.102	0.761	0.055	'(2.34-4.58]'
0.011	0.009	0.120	0.011	0.021	0.007	0.742	0.177	'(4.58-6.82]'
0.856	0.394	0.548	0.856	0.668	0.446	0.772	0.593	'(6.82-9.06]'
0.525	0.122	0.686	0.525	0.595	0.436	0.760	0.626	'(9.06-11.3]'
0.648	0.049	0.731	0.648	0.687	0.629	0.899	0.732	'(11.3-13.54]'
0.252	0.008	0.395	0.252	0.307	0.305	0.856	0.277	'(13.54-15.78]'
0.214	0.006	0.088	0.214	0.124	0.133	0.777	0.086	'(15.78-18.02]'
0.000	0.000	?	0.000	?	?	0.740	0.001	'(18.02-20.26]'
0.000	0.000	?	0.000	?	?	0.715	0.000	'(20.26-inf)'
Weighted Avg.	0.601	0.191	?	0.601	?	?	0.788	0.572

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	<-- classified as
32	1	0	3	4	0	0	0	0	0	a = '(-inf-2.34]'
14	26	5	552	5	5	0	0	0	0	b = '(2.34-4.58]'
13	26	56	4659	94	34	2	0	0	0	c = '(4.58-6.82]'
24	16	230	15060	2127	137	2	0	0	0	d = '(6.82-9.06]'
0	22	145	6175	8697	1425	53	37	0	0	e = '(9.06-11.3]'
0	8	32	754	1704	5437	292	159	0	0	f = '(11.3-13.54]'
0	0	0	181	41	398	240	94	0	0	g = '(13.54-15.78]'
0	0	0	77	4	4	18	28	0	0	h = '(15.78-18.02]'
0	0	0	27	0	0	0	1	0	0	i = '(18.02-20.26]'
0	0	0	9	0	0	0	0	0	0	j = '(20.26-inf)'

REPTree

REPTree is a “fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting). Only sorts values for numeric attributes once. Missing values are dealt with by splitting the corresponding instances into pieces” [9]. The link below contains the information used to create the REPTree, the REPTree itself, the stratified cross-validation summary, detailed accuracy by class, and the confusion matrix.

https://drive.google.com/file/d/1KpR0eUeb2OTtwSzrfdxcuRmitv_71RR/view?usp=sharing

=== Run information ===

```

Scheme:      weka.classifiers.trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
Relation:    USGS_PA_STREAM_2012-weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.RemovePercentage-P90.0-
weka.filters.unsupervised.attribute.Remove-R1-4-weka.filters.unsupervised.attribute.Remove-
R10-weka.filters.unsupervised.attribute.Reorder-R2-11,1-
weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rlast-unset-class-temporarily-
weka.filters.unsupervised.attribute.Remove-R7,9-10
Instances:   49189
Attributes:  8
              pH
              TempCelsius
              Conductance
              GageHt
              DischargeRate
              TimeOfYear
              TimeOfDay
              OxygenMgPerLiter
Test mode:   10-fold cross-validation

```

=== Classifier model (full training set) ===

REPTree

=====

```

TempCelsius < 14.55
| TempCelsius < 8.45
| | pH < 7.75
| | | TempCelsius < 4.45
| | | | Conductance < 139.5
| | | | | TempCelsius < 1.75 : '(11.3-13.54]' (22/6) [8/2]

```



```

| | | | | | | | | pH < 6.95 : '(11.3-13.54]' (62.19/8.79) [22.71/1.89]
| | | | | | | | | pH >= 6.95
| | | | | | | | | | DischargeRate < 2430
| | | | | | | | | | | Conductance < 93 : '(11.3-13.54]' (19.13/0.47) [13.07/0.95]
| | | | | | | | | | | Conductance >= 93
| | | | | | | | | | | | DischargeRate < 1160
| | | | | | | | | | | | | GageHt < 2.53
| | | | | | | | | | | | | GageHt < 2.5
| | | | | | | | | | | | | | Conductance < 107.5 : '(6.82-9.06]' (3/0) [4.15/1.15]
| | | | | | | | | | | | | | Conductance >= 107.5 : '(9.06-11.3]' (11.13/3) [2.08/0]
| | | | | | | | | | | | | | GageHt >= 2.5 : '(4.58-6.82]' (2.02/0.02) [1.03/1.03]
| | | | | | | | | | | | | | GageHt >= 2.53 : '(9.06-11.3]' (27.29/1) [15.46/0.3]
| | | | | | | | | | | | | | DischargeRate >= 1160 : '(11.3-13.54]' (31.16/15.22) [7.6/5.6]
| | | | | | | | | | | | | | DischargeRate >= 2430 : '(9.06-11.3]' (106.21/13.28) [49.62/4]
| | | | | | | | | | | | | | pH >= 7.25
| | | | | | | | | | | | | | pH < 7.45
| | | | | | | | | | | | | | | GageHt < 6.77
| | | | | | | | | | | | | | | | TempCelsius < 6.15 : '(11.3-13.54]' (139.21/4.06) [49.01/0.01]
| | | | | | | | | | | | | | | | TempCelsius >= 6.15
| | | | | | | | | | | | | | | | | TimeOfDay = morning : '(11.3-13.54]' (5.01/1.01) [4/2]
| | | | | | | | | | | | | | | | | TimeOfDay = afternoon : '(11.3-13.54]' (7.15/0) [4/0]
| | | | | | | | | | | | | | | | | TimeOfDay = evening
| | | | | | | | | | | | | | | | | | GageHt < 2.55 : '(11.3-13.54]' (3/1) [1/0]
| | | | | | | | | | | | | | | | | | GageHt >= 2.55 : '(9.06-11.3]' (2/0) [5/0]
| | | | | | | | | | | | | | | | | | TimeOfDay = night : '(11.3-13.54]' (5/1) [1/0]
| | | | | | | | | | | | | | | | | | GageHt >= 6.77 : '(9.06-11.3]' (2/0) [4/1]
| | | | | | | | | | | | | | | | | | pH >= 7.45
| | | | | | | | | | | | | | | | | | TempCelsius < 4.65 : '(11.3-13.54]' (12.01/2.01) [3/0]
| | | | | | | | | | | | | | | | | | TempCelsius >= 4.65
| | | | | | | | | | | | | | | | | | | TimeOfYear = winter : '(11.3-13.54]' (21.74/0) [11.6/0.72]
| | | | | | | | | | | | | | | | | | | TimeOfYear = spring : '(9.06-11.3]' (0.13/0.01) [0.04/0.03]
| | | | | | | | | | | | | | | | | | | TimeOfYear = summer : '(11.3-13.54]' (0/0) [0/0]
| | | | | | | | | | | | | | | | | | | TimeOfYear = autumn : '(11.3-13.54]' (202.91/0.03) [103.47/0]
| | | | | | | | | | | | | | | | | | | Conductance >= 387.5
| | | | | | | | | | | | | | | | | | | GageHt < 1.89
| | | | | | | | | | | | | | | | | | | | DischargeRate < 10.35
| | | | | | | | | | | | | | | | | | | | | TimeOfDay = morning : '(9.06-11.3]' (6/0) [3/1]
| | | | | | | | | | | | | | | | | | | | | TimeOfDay = afternoon : '(6.82-9.06]' (4/2) [2/1]
| | | | | | | | | | | | | | | | | | | | | TimeOfDay = evening : '(6.82-9.06]' (3/0) [4/3]
| | | | | | | | | | | | | | | | | | | | | TimeOfDay = night : '(6.82-9.06]' (6/3) [5/1]
| | | | | | | | | | | | | | | | | | | | DischargeRate >= 10.35
| | | | | | | | | | | | | | | | | | | | | pH < 6.85 : '(-inf-2.34]' (4/1) [2/0]
| | | | | | | | | | | | | | | | | | | | | pH >= 6.85 : '(4.58-6.82]' (6/1) [0.01/0.01]
| | | | | | | | | | | | | | | | | | | | | GageHt >= 1.89
| | | | | | | | | | | | | | | | | | | | | TimeOfDay = morning
| | | | | | | | | | | | | | | | | | | | | DischargeRate < 25.5
| | | | | | | | | | | | | | | | | | | | | GageHt < 2.25 : '(9.06-11.3]' (15/2) [2/1]
| | | | | | | | | | | | | | | | | | | | | GageHt >= 2.25
| | | | | | | | | | | | | | | | | | | | | pH < 7.15 : '(9.06-11.3]' (4/0) [1/0]

```

```

| | | | | | | | | | | | | | pH >= 7.15
| | | | | | | | | | | | | | GageHt < 3.7
| | | | | | | | | | | | | | pH < 7.45
| | | | | | | | | | | | | | TempCelsius < 6.05
| | | | | | | | | | | | | | Conductance < 649.5 : '(11.3-13.54)' (11/2) [2/0]
| | | | | | | | | | | | | | Conductance >= 649.5 : '(9.06-11.3)' (6/2) [2/0]
| | | | | | | | | | | | | | TempCelsius >= 6.05 : '(9.06-11.3)' (5/1) [4/1]
| | | | | | | | | | | | | | pH >= 7.45
| | | | | | | | | | | | | | Conductance < 662 : '(11.3-13.54)' (8/0) [8/2]
| | | | | | | | | | | | | | Conductance >= 662
| | | | | | | | | | | | | | TempCelsius < 5.2 : '(11.3-13.54)' (5/0) [1/0]
| | | | | | | | | | | | | | TempCelsius >= 5.2 : '(9.06-11.3)' (2/0) [3/1]
| | | | | | | | | | | | | | GageHt >= 3.7 : '(9.06-11.3)' (6/1) [3/0]
| | | | | | | | | | | | | | DischargeRate >= 25.5 : '(9.06-11.3)' (13.83/0.19) [6.71/2.04]
| | | | | | | | | | | | | | TimeOfDay = afternoon
| | | | | | | | | | | | | | DischargeRate < 6.25 : '(13.54-15.78)' (3/1) [1/0]
| | | | | | | | | | | | | | DischargeRate >= 6.25
| | | | | | | | | | | | | | GageHt < 3.63 : '(11.3-13.54)' (12.38/2) [6/1]
| | | | | | | | | | | | | | GageHt >= 3.63 : '(9.06-11.3)' (4.49/0) [4.86/2]
| | | | | | | | | | | | | | TimeOfDay = evening
| | | | | | | | | | | | | | GageHt < 3.25 : '(11.3-13.54)' (11/3) [7/4]
| | | | | | | | | | | | | | GageHt >= 3.25 : '(9.06-11.3)' (5.79/0) [3.49/1]
| | | | | | | | | | | | | | TimeOfDay = night : '(9.06-11.3)' (41.9/8) [18.86/3]
.
.
.
| | | | | DischargeRate >= 1980
| | | | | DischargeRate < 3555
| | | | | TimeOfYear = winter : '(4.58-6.82)' (0/0) [0/0]
| | | | | TimeOfYear = spring : '(4.58-6.82)' (33.62/4.7) [12.7/4.41]
| | | | | TimeOfYear = summer
| | | | | TempCelsius < 21.9 : '(6.82-9.06)' (3.48/1.01) [0.14/0]
| | | | | TempCelsius >= 21.9
| | | | | TempCelsius < 25.45
| | | | | pH < 8.15 : '(4.58-6.82)' (13.19/4.16) [5.74/2.37]
| | | | | pH >= 8.15 : '(6.82-9.06)' (6.6/0.52) [5.7/1.19]
| | | | | TempCelsius >= 25.45
| | | | | Conductance < 241 : '(6.82-9.06)' (2.95/0.73) [1.33/0.37]
| | | | | Conductance >= 241 : '(4.58-6.82)' (28.19/6.17) [14.65/4.08]
| | | | | TimeOfYear = autumn : '(4.58-6.82)' (0/0) [0/0]
| | | | | DischargeRate >= 3555
| | | | | TempCelsius < 24.95 : '(6.82-9.06)' (16.1/1.19) [6.44/2.61]
| | | | | TempCelsius >= 24.95
| | | | | TimeOfYear = winter : '(6.82-9.06)' (0/0) [0/0]
| | | | | TimeOfYear = spring : '(6.82-9.06)' (3.94/0.69) [5.65/0.34]
| | | | | TimeOfYear = summer
| | | | | TempCelsius < 29
| | | | | TempCelsius < 26.45 : '(6.82-9.06)' (9.93/1.25) [4.34/0.56]
| | | | | TempCelsius >= 26.45 : '(6.82-9.06)' (5.99/2.56) [6.44/1.25]
| | | | | TempCelsius >= 29 : '(6.82-9.06)' (3/0) [0/0]

```

```
| | | | | | | | TimeOfYear = autumn : '(6.82-9.06]' (0/0) [0/0]
| | | | pH >= 8.55 : '(6.82-9.06]' (29.48/14.85) [11.66/3.86]
```

Size of the tree : 4237

Time taken to build model: 0.8 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	38674	78.6233 %
Incorrectly Classified Instances	10515	21.3767 %
Kappa statistic	0.6996	
Mean absolute error	0.0598	
Root mean squared error	0.1788	
Relative absolute error	41.5489 %	
Root relative squared error	66.6549 %	
Total Number of Instances	49189	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.475	0.000	0.731	0.475	0.576	0.589	0.897	0.570	'(-inf-2.34]'
	0.557	0.003	0.687	0.557	0.615	0.614	0.944	0.631	'(2.34-4.58]'
	0.600	0.026	0.717	0.600	0.653	0.621	0.946	0.700	'(4.58-6.82]'
	0.831	0.133	0.777	0.831	0.803	0.688	0.920	0.843	'(6.82-9.06]'
	0.805	0.106	0.795	0.805	0.800	0.697	0.919	0.841	'(9.06-11.3]'
	0.816	0.031	0.844	0.816	0.830	0.796	0.966	0.881	'(11.3-13.54]'
	0.560	0.005	0.702	0.560	0.623	0.620	0.949	0.615	'(13.54-15.78]'
	0.397	0.001	0.559	0.397	0.464	0.470	0.925	0.397	'(15.78-18.02]'
	0.393	0.000	0.500	0.393	0.440	0.443	0.908	0.402	'(18.02-20.26]'
	0.444	0.000	0.800	0.444	0.571	0.596	1.000	0.700	'(20.26-inf)'
Weighted Avg.		0.786	0.091	0.785	0.786	0.784	0.700	0.931	0.826

=== Confusion Matrix ===

```
a  b  c  d  e  f  g  h  i  j  <-- classified as
19  5  2  10  4  0  0  0  0  0 | a = '(-inf-2.34]'
```

```
3 338 205  50  10  1  0  0  0  0 | b = '(2.34-4.58]'
```

```
0 134 2928 1682 122  17  1  0  0  0 | c = '(4.58-6.82]'
```

```
2  13  891 14620 2032  38  0  0  0  0 | d = '(6.82-9.06]'
```

```
2  2  49 2300 13327  857  9  8  0  0 | e = '(9.06-11.3]'
```

```
0  0  8  141 1224 6841 166  6  0  0 | f = '(11.3-13.54]'
```

```
0  0  2  20  48 332 534  16  2  0 | g = '(13.54-15.78]'
```

```
0  0  0  4  6  17  47  52  5  0 | h = '(15.78-18.02]'
```

```
0  0  0  1  0  1  4  10  11  1 | i = '(18.02-20.26]'
```

```
0  0  0  0  0  0  0  1  4  4 | j = '(20.26-inf)'
```

J48

J48 is a “class for generating a pruned or unpruned” [9]. The link below contains the information used to create J48, the J48 tree itself, the stratified cross-validation summary, detailed accuracy by class, and the confusion matrix.

<https://drive.google.com/file/d/1le7Ei89ElcUNZYySh1dyu6Flva5pjaFI/view?usp=sharing>

=== Run information ===

```

Scheme:   weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: USGS_PA_STREAM_2012-weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.RemovePercentage-P90.0-
weka.filters.unsupervised.attribute.Remove-R1-4-weka.filters.unsupervised.attribute.Remove-
R10-weka.filters.unsupervised.attribute.Reorder-R2-11,1-
weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rlast-unset-class-temporarily-
weka.filters.unsupervised.attribute.Remove-R7,9-10
Instances: 49189
Attributes: 8
    pH
    TempCelsius
    Conductance
    GageHt
    DischargeRate
    TimeOfYear
    TimeOfDay
    OxygenMgPerLiter
Test mode: 10-fold cross-validation

```

=== Classifier model (full training set) ===

J48 pruned tree

```

TempCelsius <= 14.5
| TempCelsius <= 8.3
| | TempCelsius <= 5.3
| | | TempCelsius <= 2.4
| | | | Conductance <= 171: '(11.3-13.54]' (104.02/8.02)
| | | | Conductance > 171
| | | | | pH <= 7.3: '(11.3-13.54]' (6.01/1.01)
| | | | | pH > 7.3
| | | | | | pH <= 7.7: '(13.54-15.78]' (89.04/1.04)

```

```

| | | | | | pH > 7.7
| | | | | | | TempCelsius <= 2.2
| | | | | | | | DischargeRate <= 5480: '(13.54-15.78]' (38.03/0.03)
| | | | | | | | DischargeRate > 5480: '(11.3-13.54]' (2.0/0.0)
| | | | | | | | TempCelsius > 2.2
| | | | | | | | pH <= 8.1: '(11.3-13.54]' (8.01/1.01)
| | | | | | | | pH > 8.1
| | | | | | | | TempCelsius <= 2.3
| | | | | | | | | Conductance <= 745: '(13.54-15.78]' (9.01/1.01)
| | | | | | | | | Conductance > 745: '(11.3-13.54]' (2.0/0.0)
| | | | | | | | | TempCelsius > 2.3: '(11.3-13.54]' (6.0/2.0)
| | | | | | | | TempCelsius > 2.4
| | | | | | | | GageHt <= 1.41: '(11.3-13.54]' (525.55/2.06)
| | | | | | | | GageHt > 1.41
| | | | | | | | | pH <= 7.2
| | | | | | | | | GageHt <= 5.17
| | | | | | | | | | DischargeRate <= 2450
| | | | | | | | | | DischargeRate <= 47
| | | | | | | | | | pH <= 6.9
| | | | | | | | | | TempCelsius <= 4.1: '(9.06-11.3]' (4.0)
| | | | | | | | | | TempCelsius > 4.1
| | | | | | | | | | | pH <= 6.7: '(4.58-6.82]' (2.0)
| | | | | | | | | | | pH > 6.7: '(6.82-9.06]' (11.0/1.0)
| | | | | | | | | | | pH > 6.9: '(9.06-11.3]' (28.0/7.0)
| | | | | | | | | | | DischargeRate > 47
| | | | | | | | | | | pH <= 6.4: '(11.3-13.54]' (42.08/2.01)
| | | | | | | | | | | pH > 6.4
| | | | | | | | | | | TempCelsius <= 3.9
| | | | | | | | | | | pH <= 6.9: '(6.82-9.06]' (2.0/0.0)
| | | | | | | | | | | pH > 6.9
| | | | | | | | | | | | Conductance <= 159
| | | | | | | | | | | | TempCelsius <= 3.8: '(6.82-9.06]' (3.0/1.0)
| | | | | | | | | | | | TempCelsius > 3.8: '(9.06-11.3]' (2.0/1.0)
| | | | | | | | | | | | Conductance > 159: '(9.06-11.3]' (2.0)
| | | | | | | | | | | | TempCelsius > 3.9
| | | | | | | | | | | | pH <= 6.8: '(9.06-11.3]' (3.01/2.01)
| | | | | | | | | | | | pH > 6.8: '(11.3-13.54]' (49.52/7.01)
| | | | | | | | | | | | DischargeRate > 2450
| | | | | | | | | | | | | pH <= 6.4: '(11.3-13.54]' (3.0/0.0)
| | | | | | | | | | | | | pH > 6.4: '(9.06-11.3]' (58.0/0.0)
| | | | | | | | | | | | GageHt > 5.17: '(11.3-13.54]' (100.83/0.05)
| | | | | | | | | | | | | pH > 7.2
| | | | | | | | | | | | | GageHt <= 11.62
| | | | | | | | | | | | | pH <= 8.4
| | | | | | | | | | | | | Conductance <= 592
| | | | | | | | | | | | | TempCelsius <= 4.6
| | | | | | | | | | | | | pH <= 7.4
| | | | | | | | | | | | | GageHt <= 7.67
| | | | | | | | | | | | | | GageHt <= 2.28: '(9.06-11.3]' (3.0/0.0)

```



```

| | | | | | | | | | pH > 7.7
| | | | | | | | | | | DischargeRate <= 79
| | | | | | | | | | | TempCelsius <= 3.9
| | | | | | | | | | | pH <= 8.3
| | | | | | | | | | | | GageHt <= 2.92: '(13.54-15.78]' (3.0/1.0)
| | | | | | | | | | | | GageHt > 2.92: '(11.3-13.54]' (4.12)
| | | | | | | | | | | | pH > 8.3: '(13.54-15.78]' (4.07/0.07)
| | | | | | | | | | | | TempCelsius > 3.9: '(11.3-13.54]' (10.52)
| | | | | | | | | | | | DischargeRate > 79: '(13.54-15.78]' (3.43/0.43)
| | | | | | | | | | | TimeOfDay = afternoon
| | | | | | | | | | | DischargeRate <= 55
| | | | | | | | | | | | GageHt <= 3.65: '(13.54-15.78]' (13.05/2.0)
| | | | | | | | | | | | GageHt > 3.65: '(11.3-13.54]' (5.37/0.0)
| | | | | | | | | | | | DischargeRate > 55: '(11.3-13.54]' (5.26/0.01)
| | | | | | | | | | | TimeOfDay = evening
| | | | | | | | | | | | TempCelsius <= 5.2: '(11.3-13.54]' (32.39/2.0)
| | | | | | | | | | | | TempCelsius > 5.2: '(13.54-15.78]' (3.0)
| | | | | | | | | | | TimeOfDay = night
| | | | | | | | | | | | TempCelsius <= 4.7: '(11.3-13.54]' (26.39)
| | | | | | | | | | | | TempCelsius > 4.7
| | | | | | | | | | | | DischargeRate <= 33
| | | | | | | | | | | | | GageHt <= 6.81: '(9.06-11.3]' (5.0)
| | | | | | | | | | | | | GageHt > 6.81: '(11.3-13.54]' (2.3)
| | | | | | | | | | | | | DischargeRate > 33: '(11.3-13.54]' (4.38)
| | | | | | | | | | | pH > 8.4
| | | | | | | | | | | | GageHt <= 7.15
| | | | | | | | | | | | | TimeOfDay = morning: '(11.3-13.54]' (1.05)
| | | | | | | | | | | | | TimeOfDay = afternoon: '(13.54-15.78]' (27.0/2.0)
| | | | | | | | | | | | | TimeOfDay = evening
| | | | | | | | | | | | | DischargeRate <= 1360
| | | | | | | | | | | | | GageHt <= 2.27
| | | | | | | | | | | | | | GageHt <= 2.19: '(11.3-13.54]' (3.0/1.0)
| | | | | | | | | | | | | | GageHt > 2.19: '(13.54-15.78]' (8.0/1.0)
| | | | | | | | | | | | | | GageHt > 2.27: '(11.3-13.54]' (7.0/1.0)
| | | | | | | | | | | | | | DischargeRate > 1360: '(13.54-15.78]' (9.0/1.0)
| | | | | | | | | | | | | | TimeOfDay = night: '(11.3-13.54]' (7.0/2.0)
| | | | | | | | | | | | | | GageHt > 7.15: '(11.3-13.54]' (17.09/0.0)
| | | | | | | | | | | | | | GageHt > 11.62
| | | | | | | | | | | | | | TempCelsius <= 3.6
| | | | | | | | | | | | | | | Conductance <= 216: '(13.54-15.78]' (22.17/1.17)
| | | | | | | | | | | | | | | Conductance > 216: '(4.58-6.82]' (9.01/3.01)
| | | | | | | | | | | | | | | TempCelsius > 3.6
| | | | | | | | | | | | | | | TempCelsius <= 4.8
| | | | | | | | | | | | | | | | Conductance <= 179: '(11.3-13.54]' (15.89/1.0)
| | | | | | | | | | | | | | | | Conductance > 179
| | | | | | | | | | | | | | | | Conductance <= 224
| | | | | | | | | | | | | | | | TempCelsius <= 4.7
| | | | | | | | | | | | | | | | | DischargeRate <= 16200: '(9.06-11.3]' (2.0/0.0)
| | | | | | | | | | | | | | | | | DischargeRate > 16200: '(6.82-9.06]' (2.0)

```

```

| | | | | | | | | | | TempCelsius > 4.7: '(6.82-9.06]' (2.0/0.0)
| | | | | | | | | | | Conductance > 224
| | | | | | | | | | | TimeOfDay = morning
| | | | | | | | | | | Conductance <= 246: '(11.3-13.54]' (5.0/1.0)
| | | | | | | | | | | Conductance > 246: '(2.34-4.58]' (2.22/1.22)
| | | | | | | | | | | TimeOfDay = afternoon
| | | | | | | | | | | TempCelsius <= 4.5: '(11.3-13.54]' (6.0/0.0)
| | | | | | | | | | | TempCelsius > 4.5: '(9.06-11.3]' (4.0/0.0)
| | | | | | | | | | | TimeOfDay = evening: '(11.3-13.54]' (6.0/2.0)
| | | | | | | | | | | TimeOfDay = night: '(11.3-13.54]' (6.0/1.0)
| | | | | | | | | | | TempCelsius > 4.8: '(11.3-13.54]' (24.5/0.06)
| | | | | | | | | | | TempCelsius > 5.3
.
.
.
| | | | | | | | | | | pH > 9
| | | | | | | | | | | Conductance <= 278
| | | | | | | | | | | TempCelsius <= 30.4
| | | | | | | | | | | TimeOfYear = winter: '(13.54-15.78]' (0.0)
| | | | | | | | | | | TimeOfYear = spring
| | | | | | | | | | | DischargeRate <= 2650
| | | | | | | | | | | DischargeRate <= 2440: '(13.54-15.78]' (2.05/0.05)
| | | | | | | | | | | DischargeRate > 2440: '(11.3-13.54]' (4.1/0.08)
| | | | | | | | | | | DischargeRate > 2650: '(13.54-15.78]' (6.15/0.15)
| | | | | | | | | | | TimeOfYear = summer
| | | | | | | | | | | pH <= 9.1
| | | | | | | | | | | Conductance <= 258: '(11.3-13.54]' (3.42/0.41)
| | | | | | | | | | | Conductance > 258: '(13.54-15.78]' (3.41/0.41)
| | | | | | | | | | | pH > 9.1: '(15.78-18.02]' (4.55/1.55)
| | | | | | | | | | | TimeOfYear = autumn: '(9.06-11.3]' (0.0)
| | | | | | | | | | | TempCelsius > 30.4: '(9.06-11.3]' (4.0/1.0)
| | | | | | | | | | | Conductance > 278
| | | | | | | | | | | TimeOfYear = winter: '(15.78-18.02]' (0.0)
| | | | | | | | | | | TimeOfYear = spring
| | | | | | | | | | | DischargeRate <= 604
| | | | | | | | | | | pH <= 9.1: '(13.54-15.78]' (2.24/1.23)
| | | | | | | | | | | pH > 9.1: '(15.78-18.02]' (2.24/0.23)
| | | | | | | | | | | DischargeRate > 604: '(9.06-11.3]' (2.24/0.19)
| | | | | | | | | | | TimeOfYear = summer
| | | | | | | | | | | pH <= 9.6
| | | | | | | | | | | pH <= 9.1: '(15.78-18.02]' (16.94/3.93)
| | | | | | | | | | | pH > 9.1
| | | | | | | | | | | DischargeRate <= 1890
| | | | | | | | | | | Conductance <= 328: '(18.02-20.26]' (6.35/1.35)
| | | | | | | | | | | Conductance > 328: '(20.26-inf)' (3.18/1.18)
| | | | | | | | | | | DischargeRate > 1890
| | | | | | | | | | | Conductance <= 308: '(15.78-18.02]' (11.64/2.64)
| | | | | | | | | | | Conductance > 308: '(18.02-20.26]' (13.76/3.76)
| | | | | | | | | | | pH > 9.6: '(20.26-inf)' (6.35/0.35)

```

```

| | | | | | | TimeOfYear = autumn: '(15.78-18.02]' (0.0)
| | | | | | | TimeOfDay = evening
| | | | | | | pH <= 8.9
| | | | | | | TempCelsius <= 28.7
| | | | | | | Conductance <= 256
| | | | | | | pH <= 8.8
| | | | | | | pH <= 8.7
| | | | | | | | TempCelsius <= 27.2: '(9.06-11.3]' (16.77/1.66)
| | | | | | | | TempCelsius > 27.2: '(6.82-9.06]' (3.48/0.29)
| | | | | | | | pH > 8.7
| | | | | | | | Conductance <= 249: '(9.06-11.3]' (14.62/4.51)
| | | | | | | | Conductance > 249: '(6.82-9.06]' (3.37/0.17)
| | | | | | | | pH > 8.8
| | | | | | | | DischargeRate <= 2950
| | | | | | | | | TimeOfYear = winter: '(9.06-11.3]' (0.0)
| | | | | | | | | TimeOfYear = spring: '(6.82-9.06]' (3.19/1.09)
| | | | | | | | | TimeOfYear = summer
| | | | | | | | | DischargeRate <= 2240: '(11.3-13.54]' (3.49/1.48)
| | | | | | | | | DischargeRate > 2240: '(9.06-11.3]' (2.33/0.31)
| | | | | | | | | TimeOfYear = autumn: '(9.06-11.3]' (0.0)
| | | | | | | | | DischargeRate > 2950: '(9.06-11.3]' (5.62/0.58)
| | | | | | | | Conductance > 256
| | | | | | | | TempCelsius <= 27.5
| | | | | | | | TempCelsius <= 25.2
| | | | | | | | | pH <= 8.8
| | | | | | | | | Conductance <= 606
| | | | | | | | | | Conductance <= 262: '(11.3-13.54]' (5.31/1.3)
| | | | | | | | | | Conductance > 262: '(9.06-11.3]' (28.71/5.49)
| | | | | | | | | | Conductance > 606: '(11.3-13.54]' (11.68/4.66)
| | | | | | | | | | pH > 8.8
| | | | | | | | | | Conductance <= 337
| | | | | | | | | | | DischargeRate <= 2890: '(11.3-13.54]' (7.44/0.43)
| | | | | | | | | | | DischargeRate > 2890: '(9.06-11.3]' (4.25/1.22)
| | | | | | | | | | | Conductance > 337: '(9.06-11.3]' (4.26/0.23)
| | | | | | | | | | TempCelsius > 25.2
| | | | | | | | | | | DischargeRate <= 2440
| | | | | | | | | | | TempCelsius <= 27.3
| | | | | | | | | | | | TimeOfYear = winter: '(9.06-11.3]' (0.0)
| | | | | | | | | | | | TimeOfYear = spring
| | | | | | | | | | | | TempCelsius <= 25.6
| | | | | | | | | | | | | Conductance <= 1310: '(9.06-11.3]' (3.05/1.05)
| | | | | | | | | | | | | Conductance > 1310: '(6.82-9.06]' (4.07/0.03)
| | | | | | | | | | | | | TempCelsius > 25.6: '(9.06-11.3]' (5.72/0.7)
| | | | | | | | | | | | | TimeOfYear = summer
| | | | | | | | | | | | | | Conductance <= 266: '(6.82-9.06]' (2.83/0.35)
| | | | | | | | | | | | | | Conductance > 266: '(9.06-11.3]' (11.32/4.23)
| | | | | | | | | | | | | | TimeOfYear = autumn: '(9.06-11.3]' (0.01)
| | | | | | | | | | | | | | TempCelsius > 27.3: '(11.3-13.54]' (2.42/0.4)
| | | | | | | | | | | | | | DischargeRate > 2440

```

```

| | | | | | | | | | | TempCelsius <= 25.6: '(13.54-15.78]' (3.35/1.35)
| | | | | | | | | | | TempCelsius > 25.6: '(11.3-13.54]' (10.13/3.03)
| | | | | | | | | | | TempCelsius > 27.5: '(9.06-11.3]' (15.88/1.63)
| | | | | | | | | | | TempCelsius > 28.7
| | | | | | | | | | | Conductance <= 327
| | | | | | | | | | | DischargeRate <= 4390
| | | | | | | | | | | pH <= 8.8
| | | | | | | | | | | Conductance <= 280: '(9.06-11.3]' (6.35/0.09)
| | | | | | | | | | | Conductance > 280
| | | | | | | | | | | DischargeRate <= 2330: '(9.06-11.3]' (3.04/1.04)
| | | | | | | | | | | DischargeRate > 2330: '(11.3-13.54]' (2.03/0.03)
| | | | | | | | | | | pH > 8.8: '(11.3-13.54]' (7.1/1.1)
| | | | | | | | | | | DischargeRate > 4390: '(9.06-11.3]' (10.9/0.15)
| | | | | | | | | | | Conductance > 327
| | | | | | | | | | | TimeOfYear = winter: '(11.3-13.54]' (0.0)
| | | | | | | | | | | TimeOfYear = spring: '(11.3-13.54]' (4.0/1.0)
| | | | | | | | | | | TimeOfYear = summer
| | | | | | | | | | | Conductance <= 427
| | | | | | | | | | | DischargeRate <= 2170: '(9.06-11.3]' (3.05/1.05)
| | | | | | | | | | | DischargeRate > 2170: '(11.3-13.54]' (5.09/1.09)
| | | | | | | | | | | Conductance > 427
| | | | | | | | | | | Conductance <= 731
| | | | | | | | | | | pH <= 8.7: '(13.54-15.78]' (2.04/0.04)
| | | | | | | | | | | pH > 8.7
| | | | | | | | | | | GageHt <= 5.85: '(15.78-18.02]' (2.02/0.02)
| | | | | | | | | | | GageHt > 5.85: '(13.54-15.78]' (3.07/1.07)
| | | | | | | | | | | Conductance > 731: '(9.06-11.3]' (2.04/0.03)
| | | | | | | | | | | TimeOfYear = autumn: '(11.3-13.54]' (0.0)
| | | | | | | | | | | pH > 8.9
| | | | | | | | | | | pH <= 9.2
| | | | | | | | | | | Conductance <= 262
| | | | | | | | | | | pH <= 9
| | | | | | | | | | | Conductance <= 243: '(11.3-13.54]' (9.88/3.85)
| | | | | | | | | | | Conductance > 243: '(9.06-11.3]' (5.49/1.46)
| | | | | | | | | | | pH > 9
| | | | | | | | | | | DischargeRate <= 2560: '(11.3-13.54]' (4.39/0.38)
| | | | | | | | | | | DischargeRate > 2560
| | | | | | | | | | | TimeOfYear = winter: '(9.06-11.3]' (0.0)
| | | | | | | | | | | TimeOfYear = spring
| | | | | | | | | | | DischargeRate <= 2660: '(9.06-11.3]' (2.08/0.07)
| | | | | | | | | | | DischargeRate > 2660: '(11.3-13.54]' (2.08/0.07)
| | | | | | | | | | | TimeOfYear = summer
| | | | | | | | | | | Conductance <= 252: '(11.3-13.54]' (3.43/1.43)
| | | | | | | | | | | Conductance > 252: '(9.06-11.3]' (2.29/0.28)
| | | | | | | | | | | TimeOfYear = autumn: '(9.06-11.3]' (0.0)
| | | | | | | | | | | Conductance > 262
| | | | | | | | | | | DischargeRate <= 4600
| | | | | | | | | | | pH <= 9
| | | | | | | | | | | TempCelsius <= 30.4: '(11.3-13.54]' (19.36/4.28)

```

```

| | | | | | | | | | TempCelsius > 30.4: '(13.54-15.78]' (7.0/1.0)
| | | | | | | | | | pH > 9
| | | | | | | | | | Conductance <= 325
| | | | | | | | | | pH <= 9.1: '(13.54-15.78]' (4.39/2.39)
| | | | | | | | | | pH > 9.1: '(15.78-18.02]' (3.29/1.29)
| | | | | | | | | | Conductance > 325
| | | | | | | | | | DischargeRate <= 2400: '(13.54-15.78]' (7.69/1.68)
| | | | | | | | | | DischargeRate > 2400: '(11.3-13.54]' (3.29/1.28)
| | | | | | | | | | DischargeRate > 4600: '(11.3-13.54]' (3.29/1.28)
| | | | | | | | | | pH > 9.2
| | | | | | | | | | TempCelsius <= 27.8: '(13.54-15.78]' (4.38/1.36)
| | | | | | | | | | TempCelsius > 27.8
| | | | | | | | | | pH <= 9.3
| | | | | | | | | | Conductance <= 314: '(15.78-18.02]' (2.03/0.03)
| | | | | | | | | | Conductance > 314: '(13.54-15.78]' (3.04/1.04)
| | | | | | | | | | pH > 9.3
| | | | | | | | | | Conductance <= 317: '(15.78-18.02]' (4.06/1.06)
| | | | | | | | | | Conductance > 317: '(18.02-20.26]' (4.06/0.06)
| | | | | TimeOfDay = night
| | | | | TimeOfYear = winter: '(6.82-9.06]' (0.0)
| | | | | TimeOfYear = spring
| | | | | Conductance <= 1410: '(9.06-11.3]' (7.22/1.91)
| | | | | Conductance > 1410: '(6.82-9.06]' (5.78/2.02)
| | | | | TimeOfYear = summer: '(6.82-9.06]' (24.58/11.79)
| | | | | TimeOfYear = autumn: '(6.82-9.06]' (0.0)

```

Number of Leaves : 2584

Size of the tree : 4817

Time taken to build model: 3.11 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	39584	80.4733 %
Incorrectly Classified Instances	9605	19.5267 %
Kappa statistic	0.7264	
Mean absolute error	0.0556	
Root mean squared error	0.1745	
Relative absolute error	38.6557 %	
Root relative squared error	65.0762 %	
Total Number of Instances	49189	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.700	0.000	0.718	0.700	0.709	0.709	0.911	0.691	'(-inf-2.34]'

0.643	0.003	0.730	0.643	0.684	0.681	0.944	0.660	'(2.34-4.58]'
0.644	0.025	0.738	0.644	0.688	0.658	0.947	0.703	'(4.58-6.82]'
0.839	0.121	0.795	0.839	0.816	0.710	0.923	0.839	'(6.82-9.06]'
0.819	0.094	0.815	0.819	0.817	0.723	0.920	0.841	'(9.06-11.3]'
0.839	0.028	0.860	0.839	0.849	0.819	0.962	0.874	'(11.3-13.54]'
0.619	0.005	0.723	0.619	0.667	0.663	0.942	0.629	'(13.54-15.78]'
0.450	0.001	0.546	0.450	0.494	0.495	0.893	0.400	'(15.78-18.02]'
0.250	0.000	0.368	0.250	0.298	0.303	0.852	0.302	'(18.02-20.26]'
0.556	0.000	0.714	0.556	0.625	0.630	0.831	0.439	'(20.26-inf)'
Weighted Avg.	0.805	0.082	0.804	0.805	0.804	0.726	0.931	0.824

=== Confusion Matrix ===

```

a  b  c  d  e  f  g  h  i  j  <-- classified as
28  7  1  4  0  0  0  0  0  0 | a = '(-inf-2.34]'
9 390 164 34  9  1  0  0  0  0 | b = '(2.34-4.58]'
2 121 3145 1517 87 12  0  0  0  0 | c = '(4.58-6.82]'
0 16 886 14770 1895 28  1  0  0  0 | d = '(6.82-9.06]'
0  0  61 2112 13553 814 12  2  0  0 | e = '(9.06-11.3]'
0  0  5 126 1039 7036 177  3  0  0 | f = '(11.3-13.54]'
0  0  0 19 41 277 591 25  1  0 | g = '(13.54-15.78]'
0  0  0  3 10 15 35 59  9  0 | h = '(15.78-18.02]'
0  0  0  1  0  0  1 17  7  2 | i = '(18.02-20.26]'
0  0  0  0  0  0  0  2  2  5 | j = '(20.26-inf)'

```

RandomTree

RandomTree is a “class for constructing a tree that considers K randomly chosen attributes at each node. Performs no pruning. Also has an option to allow estimation of class probabilities (or target mean in the regression case) based on a hold-out set (backfitting)” [9]. The link below contains the information used to create the RandomTree, the RandomTree itself, the stratified cross-validation summary, detailed accuracy by class, and the confusion matrix.

https://drive.google.com/file/d/1IHZR4_WzGtZPsWji3qm-aLi_0d59uUka/view?usp=sharing

=== Run information ===

```

Scheme:   weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1
Relation: USGS_PA_STREAM_2012-weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.RemovePercentage-P90.0-
weka.filters.unsupervised.attribute.Remove-R1-4-weka.filters.unsupervised.attribute.Remove-

```

R10-weka.filters.unsupervised.attribute.Reorder-R2-11,1-
 weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rlast-unset-class-temporarily-
 weka.filters.unsupervised.attribute.Remove-R7,9-10

Instances: 49189

Attributes: 8

pH
 TempCelsius
 Conductance
 GageHt
 DischargeRate
 TimeOfYear
 TimeOfDay
 OxygenMgPerLiter

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

RandomTree

=====

TempCelsius < 14.55

| pH < 7.75

| | Conductance < 88.5

| | | TempCelsius < 7.35

| | | | TempCelsius < 4.45

| | | | | GageHt < 3.45

| | | | | | DischargeRate < 73.5

| | | | | | | TempCelsius < 1.75

| | | | | | | | TimeOfDay = morning

| | | | | | | | | GageHt < 0.88 : '(13.54-15.78]' (5/0)

| | | | | | | | | GageHt >= 0.88

| | | | | | | | | | pH < 7.05 : '(11.3-13.54]' (3/0)

| | | | | | | | | | pH >= 7.05

| | | | | | | | | | | TempCelsius < 1.5 : '(13.54-15.78]' (1/0)

| | | | | | | | | | | TempCelsius >= 1.5 : '(11.3-13.54]' (1/0)

| | | | | | | | | | | TimeOfDay = afternoon

| | | | | | | | | | | | pH < 7

| | | | | | | | | | | | | Conductance < 36.5 : '(13.54-15.78]' (1/0)

| | | | | | | | | | | | | Conductance >= 36.5 : '(11.3-13.54]' (1/0)

| | | | | | | | | | | | | pH >= 7 : '(11.3-13.54]' (1/0)

| | | | | | | | | | | | | TimeOfDay = evening

| | | | | | | | | | | | | | Conductance < 36.5

| | | | | | | | | | | | | | | TempCelsius < 0.4 : '(11.3-13.54]' (1/0)

| | | | | | | | | | | | | | | TempCelsius >= 0.4 : '(13.54-15.78]' (1/0)

| | | | | | | | | | | | | | | Conductance >= 36.5 : '(11.3-13.54]' (3/0)

| | | | | | | | | | | | | | | TimeOfDay = night : '(11.3-13.54]' (9/0)

| | | | | | | | | | | | | | | | TempCelsius >= 1.75

| | | | | | | | | | | | | | | | | GageHt < 0.84 : '(11.3-13.54]' (241/0)


```

| | | | | | | | | | GageHt >= 0.84
| | | | | | | | | | DischargeRate < 52.5 : '(11.3-13.54]' (115/0)
| | | | | | | | | | DischargeRate >= 52.5
| | | | | | | | | | TimeOfDay = morning : '(11.3-13.54]' (1/0)
| | | | | | | | | | TimeOfDay = afternoon
| | | | | | | | | | Conductance < 54.5 : '(11.3-13.54]' (2/0)
| | | | | | | | | | Conductance >= 54.5 : '(13.54-15.78]' (1/0)
| | | | | | | | | | TimeOfDay = evening : '(11.3-13.54]' (4/0)
| | | | | | | | | | TimeOfDay = night : '(11.3-13.54]' (6/0)
| | | | | | DischargeRate >= 73.5
| | | | | | TimeOfDay = morning : '(11.3-13.54]' (34.78/0)
| | | | | | TimeOfDay = afternoon
| | | | | | DischargeRate < 505
| | | | | | TimeOfYear = winter : '(-inf-2.34]' (0/0)
| | | | | | TimeOfYear = spring : '(6.82-9.06]' (0.01/0)
| | | | | | TimeOfYear = summer : '(-inf-2.34]' (0/0)
| | | | | | TimeOfYear = autumn : '(11.3-13.54]' (19/0)
| | | | | | DischargeRate >= 505 : '(11.3-13.54]' (1.05/0.05)
| | | | | | TimeOfDay = evening : '(11.3-13.54]' (28/0)
| | | | | | TimeOfDay = night : '(11.3-13.54]' (30/0)
| | | | | | GageHt >= 3.45
| | | | | | TempCelsius < 4.15
| | | | | | TimeOfYear = winter : '(-inf-2.34]' (0/0)
| | | | | | TimeOfYear = spring : '(9.06-11.3]' (0/0)
| | | | | | TimeOfYear = summer : '(-inf-2.34]' (0/0)
| | | | | | TimeOfYear = autumn : '(11.3-13.54]' (10.39/0)
| | | | | | TempCelsius >= 4.15
| | | | | | GageHt < 6.88
| | | | | | DischargeRate < 2450 : '(9.06-11.3]' (1.83/0.56)
| | | | | | DischargeRate >= 2450 : '(9.06-11.3]' (0.77/0)
| | | | | | GageHt >= 6.88
| | | | | | TimeOfYear = winter : '(11.3-13.54]' (0.1/0)
| | | | | | TimeOfYear = spring : '(11.3-13.54]' (0.07/0)
| | | | | | TimeOfYear = summer : '(-inf-2.34]' (0/0)
| | | | | | TimeOfYear = autumn : '(11.3-13.54]' (10.03/0)
| | | | | | TempCelsius >= 4.45
| | | | | | pH < 7.05
| | | | | | DischargeRate < 344
| | | | | | Conductance < 64.5
| | | | | | DischargeRate < 249.5
| | | | | | TempCelsius < 6.55 : '(11.3-13.54]' (127.83/0)
| | | | | | TempCelsius >= 6.55
| | | | | | Conductance < 49.5
| | | | | | TimeOfYear = winter : '(11.3-13.54]' (4/0)
| | | | | | TimeOfYear = spring : '(11.3-13.54]' (16/0)
| | | | | | TimeOfYear = summer : '(-inf-2.34]' (0/0)
| | | | | | TimeOfYear = autumn
| | | | | | TempCelsius < 7.15
| | | | | | TempCelsius < 6.75 : '(11.3-13.54]' (1.02/0.48)

```

```

| | | | | | | | | | | | | TempCelsius >= 6.75
| | | | | | | | | | | | | | TimeOfDay = morning : '(11.3-13.54]' (0.39/0.13)
| | | | | | | | | | | | | | TimeOfDay = afternoon : '(11.3-13.54]' (0.61/0.26)
| | | | | | | | | | | | | | TimeOfDay = evening : '(11.3-13.54]' (0.59/0.13)
| | | | | | | | | | | | | | TimeOfDay = night : '(11.3-13.54]' (0.54/0.13)
| | | | | | | | | | | | | | TempCelsius >= 7.15 : '(11.3-13.54]' (0.91/0.02)
| | | | | | | | | | | | | | Conductance >= 49.5
| | | | | | | | | | | | | | TempCelsius < 7.15
| | | | | | | | | | | | | | Conductance < 55
| | | | | | | | | | | | | | TimeOfYear = winter : '(-inf-2.34]' (0/0)
| | | | | | | | | | | | | | TimeOfYear = spring : '(11.3-13.54]' (10/0)
| | | | | | | | | | | | | | TimeOfYear = summer : '(-inf-2.34]' (0/0)
| | | | | | | | | | | | | | TimeOfYear = autumn
| | | | | | | | | | | | | | TempCelsius < 6.95
| | | | | | | | | | | | | | TempCelsius < 6.65
| | | | | | | | | | | | | | | TimeOfDay = morning : '(11.3-13.54]' (1.25/0.09)
| | | | | | | | | | | | | | | TimeOfDay = afternoon : '(11.3-13.54]' (0.13/0)
| | | | | | | | | | | | | | | TimeOfDay = evening : '(11.3-13.54]' (0.07/0.02)
| | | | | | | | | | | | | | | TimeOfDay = night : '(11.3-13.54]' (2.04/0)
| | | | | | | | | | | | | | TempCelsius >= 6.65
| | | | | | | | | | | | | | | TimeOfDay = morning
| | | | | | | | | | | | | | | TempCelsius < 6.85
| | | | | | | | | | | | | | | | TempCelsius < 6.75 : '(11.3-13.54]' (2.09/0.04)
| | | | | | | | | | | | | | | | TempCelsius >= 6.75 : '(11.3-13.54]' (1.07/0.02)
| | | | | | | | | | | | | | | | TempCelsius >= 6.85 : '(9.06-11.3]' (0.11/0.04)
| | | | | | | | | | | | | | | | TimeOfDay = afternoon : '(11.3-13.54]' (1.36/0.22)
| | | | | | | | | | | | | | | | TimeOfDay = evening : '(11.3-13.54]' (1.36/0.16)
| | | | | | | | | | | | | | | | TimeOfDay = night : '(11.3-13.54]' (0.51/0.16)
| | | | | | | | | | | | | | | | TempCelsius >= 6.95 : '(11.3-13.54]' (1.25/0.38)
| | | | | | | | | | | | | | Conductance >= 55
| | | | | | | | | | | | | | pH < 6.95 : '(9.06-11.3]' (1.18/0.12)
| | | | | | | | | | | | | | pH >= 6.95
| | | | | | | | | | | | | | TempCelsius < 6.85
| | | | | | | | | | | | | | | TempCelsius < 6.65 : '(11.3-13.54]' (0.05/0.01)
| | | | | | | | | | | | | | | TempCelsius >= 6.65
| | | | | | | | | | | | | | | | TempCelsius < 6.75 : '(11.3-13.54]' (1.06/0.04)
| | | | | | | | | | | | | | | | TempCelsius >= 6.75 : '(11.3-13.54]' (1.07/0.02)
| | | | | | | | | | | | | | | | TempCelsius >= 6.85 : '(11.3-13.54]' (0.18/0.05)
| | | | | | | | | | | | | | TempCelsius >= 7.15
| | | | | | | | | | | | | | TempCelsius < 7.25
| | | | | | | | | | | | | | | TimeOfYear = winter : '(-inf-2.34]' (0/0)
| | | | | | | | | | | | | | | TimeOfYear = spring : '(9.06-11.3]' (2/0)
| | | | | | | | | | | | | | | TimeOfYear = summer : '(-inf-2.34]' (0/0)
| | | | | | | | | | | | | | | TimeOfYear = autumn : '(11.3-13.54]' (0.7/0)
| | | | | | | | | | | | | | | TempCelsius >= 7.25 : '(9.06-11.3]' (1.39/0.37)
| | | | | | | | | | | | | | | DischargeRate >= 249.5 : '(9.06-11.3]' (1.21/0.2)
| | | | | | | | | | | | | | Conductance >= 64.5
| | | | | | | | | | | | | | TempCelsius < 5.95 : '(11.3-13.54]' (13.67/0)
| | | | | | | | | | | | | | TempCelsius >= 5.95

```



```

.
.
| | | | | pH >= 7.95
| | | | | pH < 8.55
| | | | | DischargeRate < 499.5
| | | | | Conductance < 960.5
| | | | | GageHt < 10.32
| | | | | GageHt < 10.18
| | | | | GageHt < 9.69 : '(6.82-9.06]' (0.71/0.03)
| | | | | GageHt >= 9.69
| | | | | TempCelsius < 25.15
| | | | | TempCelsius < 23.15 : '(6.82-9.06]' (1.78/0)
| | | | | TempCelsius >= 23.15
| | | | | TempCelsius < 24.3 : '(6.82-9.06]' (1.38/0.02)
| | | | | TempCelsius >= 24.3 : '(6.82-9.06]' (0.82/0)
| | | | | TempCelsius >= 25.15 : '(6.82-9.06]' (1.52/0.36)
| | | | | GageHt >= 10.18
| | | | | TempCelsius < 25.15
| | | | | TempCelsius < 23.15 : '(6.82-9.06]' (1.79/0)
| | | | | TempCelsius >= 23.15
| | | | | TempCelsius < 24.3 : '(6.82-9.06]' (1.36/0.02)
| | | | | TempCelsius >= 24.3 : '(6.82-9.06]' (0.85/0)
| | | | | TempCelsius >= 25.15 : '(6.82-9.06]' (1.68/0.53)
| | | | | GageHt >= 10.32
| | | | | TempCelsius < 25.15
| | | | | TempCelsius < 23.15 : '(6.82-9.06]' (7.22/0)
| | | | | TempCelsius >= 23.15
| | | | | TempCelsius < 23.7
| | | | | TempCelsius < 23.55 : '(6.82-9.06]' (2.02/0.02)
| | | | | TempCelsius >= 23.55 : '(6.82-9.06]' (1.32/0)
| | | | | TempCelsius >= 23.7
| | | | | TempCelsius < 23.85 : '(4.58-6.82]' (0.02/0)
| | | | | TempCelsius >= 23.85
| | | | | TempCelsius < 24.45
| | | | | GageHt < 10.57 : '(6.82-9.06]' (0.42/0.01)
| | | | | GageHt >= 10.57 : '(6.82-9.06]' (1.68/0.08)
| | | | | TempCelsius >= 24.45
| | | | | TempCelsius < 24.95 : '(6.82-9.06]' (2.8/0)
| | | | | TempCelsius >= 24.95 : '(6.82-9.06]' (0.81/0.04)
| | | | | TempCelsius >= 25.15
| | | | | TempCelsius < 28.65
| | | | | TempCelsius < 27.85
| | | | | GageHt < 10.68
| | | | | TempCelsius < 27.35
| | | | | GageHt < 10.41 : '(6.82-9.06]' (0.57/0.09)
| | | | | GageHt >= 10.41
| | | | | GageHt < 10.66 : '(6.82-9.06]' (1.93/0.46)
| | | | | GageHt >= 10.66 : '(6.82-9.06]' (0.18/0.06)
| | | | | TempCelsius >= 27.35 : '(6.82-9.06]' (0.92/0.03)

```

```

| | | | | | | | | | | | | | | GageHt >= 10.68 : '(6.82-9.06]' (1.5/0.17)
| | | | | | | | | | | | | | | TempCelsius >= 27.85 : '(4.58-6.82]' (0.31/0.04)
| | | | | | | | | | | | | | | TempCelsius >= 28.65 : '(6.82-9.06]' (1.18/0.07)
| | | | | | | | | | | | | | | Conductance >= 960.5
| | | | | | | | | | | | | | | Conductance < 992
| | | | | | | | | | | | | | | TempCelsius < 28.35 : '(4.58-6.82]' (1.28/0.15)
| | | | | | | | | | | | | | | TempCelsius >= 28.35 : '(6.82-9.06]' (1.03/0.03)
| | | | | | | | | | | | | | | Conductance >= 992
| | | | | | | | | | | | | | | GageHt < 10.32 : '(4.58-6.82]' (1.18/0.06)
| | | | | | | | | | | | | | | GageHt >= 10.32
| | | | | | | | | | | | | | | GageHt < 10.73 : '(4.58-6.82]' (1.92/0.14)
| | | | | | | | | | | | | | | GageHt >= 10.73 : '(4.58-6.82]' (0.35/0.03)
| | | | | | | | | | | | | | | DischargeRate >= 499.5
| | | | | | | | | | | | | | | Conductance < 455.5 : '(6.82-9.06]' (0.93/0.06)
| | | | | | | | | | | | | | | Conductance >= 455.5
| | | | | | | | | | | | | | | Conductance < 481
| | | | | | | | | | | | | | | GageHt < 7.76
| | | | | | | | | | | | | | | TempCelsius < 28.05 : '(4.58-6.82]' (0.19/0.08)
| | | | | | | | | | | | | | | TempCelsius >= 28.05
| | | | | | | | | | | | | | | TempCelsius < 28.35
| | | | | | | | | | | | | | | TempCelsius < 28.25
| | | | | | | | | | | | | | | TempCelsius < 28.15 : '(9.06-11.3]' (1/0)
| | | | | | | | | | | | | | | TempCelsius >= 28.15 : '(9.06-11.3]' (1/0)
| | | | | | | | | | | | | | | TempCelsius >= 28.25 : '(9.06-11.3]' (1.01/0.01)
| | | | | | | | | | | | | | | TempCelsius >= 28.35 : '(4.58-6.82]' (0.01/0)
| | | | | | | | | | | | | | | GageHt >= 7.76 : '(6.82-9.06]' (0.24/0.1)
| | | | | | | | | | | | | | | Conductance >= 481
| | | | | | | | | | | | | | | TempCelsius < 29.8
| | | | | | | | | | | | | | | TempCelsius < 29.25 : '(4.58-6.82]' (0.46/0.23)
| | | | | | | | | | | | | | | TempCelsius >= 29.25 : '(9.06-11.3]' (2/0)
| | | | | | | | | | | | | | | TempCelsius >= 29.8 : '(11.3-13.54]' (1/0)
| | | | | | | | | | | | | | | pH >= 8.55
| | | | | | | | | | | | | | | DischargeRate < 257.5 : '(6.82-9.06]' (1.15/0.08)
| | | | | | | | | | | | | | | DischargeRate >= 257.5
| | | | | | | | | | | | | | | Conductance < 488.5
| | | | | | | | | | | | | | | TempCelsius < 29.35 : '(4.58-6.82]' (0.3/0.15)
| | | | | | | | | | | | | | | TempCelsius >= 29.35 : '(13.54-15.78]' (2/0)
| | | | | | | | | | | | | | | Conductance >= 488.5 : '(11.3-13.54]' (1.15/0.15)
| | TimeOfYear = autumn
| | | TimeOfDay = morning
| | | | pH < 7.15
| | | | | TempCelsius < 21.05 : '(4.58-6.82]' (1/0)
| | | | | TempCelsius >= 21.05 : '(6.82-9.06]' (1/0)
| | | | | pH >= 7.15
| | | | | Conductance < 283.5 : '(9.06-11.3]' (0.33/0)
| | | | | Conductance >= 283.5 : '(4.58-6.82]' (2/0)
| | | TimeOfDay = afternoon
| | | | GageHt < 6.58
| | | | | TempCelsius < 21.05

```

```

| | | | | | Conductance < 428.5 : '(4.58-6.82]' (1.61/0.23)
| | | | | | Conductance >= 428.5 : '(9.06-11.3]' (1.23/0.23)
| | | | | | TempCelsius >= 21.05 : '(9.06-11.3]' (1.2/0.1)
| | | | | | GageHt >= 6.58
| | | | | | pH < 7.25
| | | | | | TempCelsius < 21.05
| | | | | | | Conductance < 424 : '(4.58-6.82]' (3.62/0)
| | | | | | | Conductance >= 424 : '(6.82-9.06]' (2/0)
| | | | | | | TempCelsius >= 21.05 : '(6.82-9.06]' (4/0)
| | | | | | | pH >= 7.25 : '(6.82-9.06]' (1/0)
| | | | TimeOfDay = evening
| | | | | Conductance < 408.5
| | | | | Conductance < 271.5
| | | | | Conductance < 269.5
| | | | | | | Conductance < 267.5 : '(4.58-6.82]' (1/0)
| | | | | | | Conductance >= 267.5 : '(4.58-6.82]' (1.33/0.33)
| | | | | | | Conductance >= 269.5 : '(9.06-11.3]' (0.33/0)
| | | | | | | Conductance >= 271.5 : '(4.58-6.82]' (3/0)
| | | | | | | Conductance >= 408.5 : '(6.82-9.06]' (5/0)
| | | | TimeOfDay = night
| | | | | Conductance < 437 : '(4.58-6.82]' (8/0)
| | | | | Conductance >= 437 : '(6.82-9.06]' (1/0)

```

Size of the tree : 61007

Time taken to build model: 0.69 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	38746	78.7696 %
Incorrectly Classified Instances	10443	21.2304 %
Kappa statistic	0.7023	
Mean absolute error	0.0494	
Root mean squared error	0.1894	
Relative absolute error	34.3158 %	
Root relative squared error	70.6146 %	
Total Number of Instances	49189	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.450	0.000	0.783	0.450	0.571	0.593	0.723	0.431	'(-inf-2.34]'
0.598	0.003	0.727	0.598	0.656	0.656	0.875	0.545	'(2.34-4.58]'
0.645	0.029	0.710	0.645	0.676	0.643	0.919	0.664	'(4.58-6.82]'
0.840	0.144	0.764	0.840	0.800	0.683	0.897	0.794	'(6.82-9.06]'
0.783	0.093	0.810	0.783	0.796	0.696	0.880	0.789	'(9.06-11.3]'
0.812	0.026	0.865	0.812	0.838	0.806	0.908	0.803	'(11.3-13.54]'
0.614	0.006	0.670	0.614	0.641	0.635	0.827	0.500	'(13.54-15.78]'

	0.489	0.001	0.520	0.489	0.504	0.503	0.799	0.302	'(15.78-18.02]'
	0.286	0.000	0.421	0.286	0.340	0.347	0.655	0.167	'(18.02-20.26]'
	0.667	0.000	0.750	0.667	0.706	0.707	0.832	0.527	'(20.26-inf)'
Weighted Avg.	0.788	0.090	0.788	0.788	0.787	0.702	0.893	0.770	

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	<-- classified as
18	6	2	8	6	0	0	0	0	0	a = '(-inf-2.34]'
1	363	176	57	8	2	0	0	0	0	b = '(2.34-4.58]'
1	107	3152	1516	93	14	0	0	1	0	c = '(4.58-6.82]'
1	16	935	14785	1786	58	11	4	0	0	d = '(6.82-9.06]'
2	6	131	2628	12958	759	55	15	0	0	e = '(9.06-11.3]'
0	1	38	289	1048	6806	196	7	0	1	f = '(11.3-13.54]'
0	0	4	49	75	215	586	24	1	0	g = '(13.54-15.78]'
0	0	2	12	12	9	25	64	7	0	h = '(15.78-18.02]'
0	0	1	3	2	2	2	9	8	1	i = '(18.02-20.26]'
0	0	1	0	0	0	0	0	2	6	j = '(20.26-inf)'