CLASSIFICATION OF LIDAR DATA USING WINDOW-BASED TECHNIQUES

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Shuhang Li

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2016

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Classification of LiDAR Data using Window-based Techniques

By

Shuhang Li

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Anne Denton
Chair

Stephanie Day

Changhui Yan

Approved:

November 17th 2016
Date

Brian M. Slator
Department Chair

ABSTRACT

Given LiDAR maps, we focus on identifying anthropologically relevant ditches automatically on the map. Archeologists can identify these features visually at the site, but approaches based on remotely sensed data would be preferable. This paper proposes an algorithm that uses window-based technique to read the characteristics of each region from maps, whose ditches are already identified, regressively, and then builds histograms to represent the different characters of each region. A classification model is then built based on the histograms and used to predict future data. The goal is to produce a large training data set using window-based technology and use it to classify future data. We demonstrated our algorithm successfully identifies target regions efficiently on real LiDAR maps.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation through grants PFI-1114363 and IIA-1355466.

Anne Denton, Dept. of Computer Science, NDSU

Matthew Radermacher, Dept. of Geoscience, NDSU

Stephanie Day, Dept. of Geoscience, NDSU

Seth Quintus, Dept. of Geoscience, NDSU

Jeffrey Clark, Dept. of Geoscience, NDSU

Donald Schwert, Dept. of Geoscience, NDSU

Nolita Motu, Dept. of Geoscience, NDSU

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES vi

LIST OF FIGURES vii

1. INTRODUCTION 1

1.1. Problem Statement 2

2. RELATED WORK 5

3. GEOSCIENCE AND ANTHROPOLOGY CONCEPTS 7

4. DATA MINING CONCEPTS 8

4.1. Classification 8

4.2. Preprocessing 10

4.3. Weka 11

4.4. J48 decision tree 12

5. ALGORITHMS 15

5.1. J48 decision tree 15

5.2. Window 16

5.3. Histogram 21

5.4. Target value 32

5.5. Write the file 32

5.6. Calculate the result 35

6. EXPERIMENTS AND RESULTS 36

7. CONCLUSION 43

REFERENCES 44

LIST OF TABLES

Table Page

1. Result of classification 9

2. Results of experiments 36

LIST OF FIGURES

Figure Page

1. Slide windows 3

2. Window with a ditch 3

3. Aspect histogram based on the window 4

4. Header section 10

5. Data section 11

6. GUI of Weka 12

7. Part of J48 decision tree 14

8. Part of an aspect file 16

9. Ditch example 20

10. Separated blue windows 20

11. Separated red windows 20

12. The aspect histogram 26

13. The slope histogram 26

14. The curvature histogram 26

15. First aspect histogram 28

16. Second aspect histogram 28

17. Final aspect histogram 29

18. A screenshot of a training dataset 34

19. Shea site image with result 37

20. Biesterfeldt site image with result 38

21. Peterson site image with result 38

22. Nelson site image with result 39

23. Biesterfieldt tree 40

24. LaRMounds tree 40
25. Nelson tree 41
26. Peterson tree 41
27. Shea tree 42
28. Sprunk tree 42

1. INTRODUCTION

Data mining allows users to classify, group, or identify patterns in data. Data mining technologies have been used by business companies for years to analyze market reports and find the patterns of the customers. Together with remote sensing technologies, new opportunities are emerging rapidly. Light detection and ranging (LiDAR) is a technology that uses the time taken for a pulse of light to reach the target and return to measure the distance between the device and target. LiDAR constitutes a very important innovation for data collection and interpretation in archaeology [OPI13]. Airborne LiDAR devices are installed on aircraft and use a laser beam that will scan from side to side as the aircraft flies over the area. Such devices measure between 20 to 100 thousand points per second to build an accurate, high-resolution model of the ground and the features upon it [CRU06]. The development of LiDAR technologies helps researching historical sites and allows archeologists, civil engineers and historians to study and understand how people lived their daily lives centuries ago. Even when LiDAR data are available, visual inspection can be inconclusive and complicated by the massive size of the area. Specialized automated techniques exist for recognizing specific archeological artifacts, but they do not generalize to other features.

We examine the possibility of using a data-mining approach, based on window histograms to help identifying the ditches. In this research, we extract information from windows based on previous data sets for which ditches are already identified and build machine learning models to classify future data sets. This research builds models based on several training datasets and tests them on different datasets. Besides helping archeologists to find sites, it also provides a potential relationship between archeological sites and archeological characteristics. In this paper, we use the J48 classification tree algorithm in Weka to do the classification.

In principle, the approach could be directly applied to other types of archeological features. If we used a dataset as training set in which mounds have been identified, the algorithm would still work in principle. However, we did not test the generalizability.

1.1. Problem Statement

The problem of this research is to successfully identify the relationship between archeological characteristics and the target features. We are using three characteristics: slope, aspect, and curvature to build models and find features. Our approach is to divide datasets into windows with fixed size and extract these attributes from each window for classification. We assume there is a relationship between the three characteristics and the feature we need to identify, and let the program to find the relationships, build a decision tree model based on it, and test the model.

There is some existing research with a focus on similar problems. Oner Ulvi Celepcikay and Christoph F. Eick proposed a framework to discover interesting geological areas [CEL09]. George Vosselman designed a computational approach to distinguish building and vegetation areas in raw laser data [VOS00]. However, both of these approaches are based on specific assumptions resulting in limitations on the data for which they can be used.

In our research, we use data mining techniques based on window histograms to classify the datasets. Matthew Radermacher from Geoscience Department, NDSU preprocessed the data we used from the LiDAR digital elevation model source. Then we apply sliding window extraction to the whole map to divide the data. Each piece of data contains all the information of a window. Figure 1 shows how we slide the window. Based on the data inside each window three histograms were made. Each histogram is based on a single attribute inside the window. For example, a window is represented in Figure 2. The two grey lines are the edges of a ditch.

Black arrows are used to represent what the aspect values might look like. In Figure 3, an aspect histogram of this window is represented. The edges effects the values of the 3rd and 6th columns a lot so a pattern appears in the histogram. Finally we put the preprocessed histograms into Weka to use decision tree for classification.

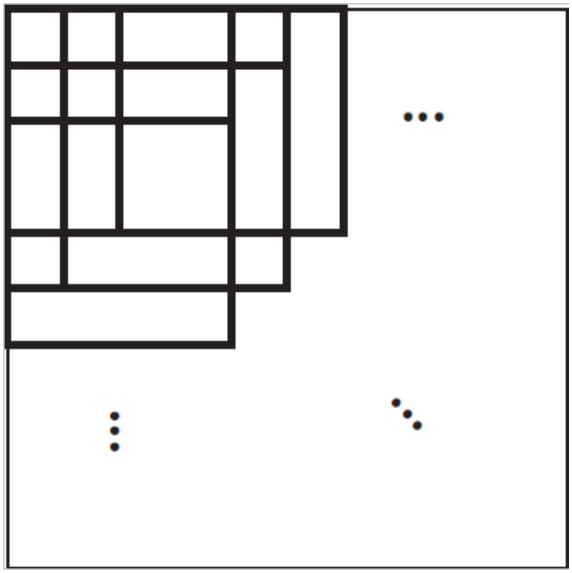


Figure 1. Slide windows

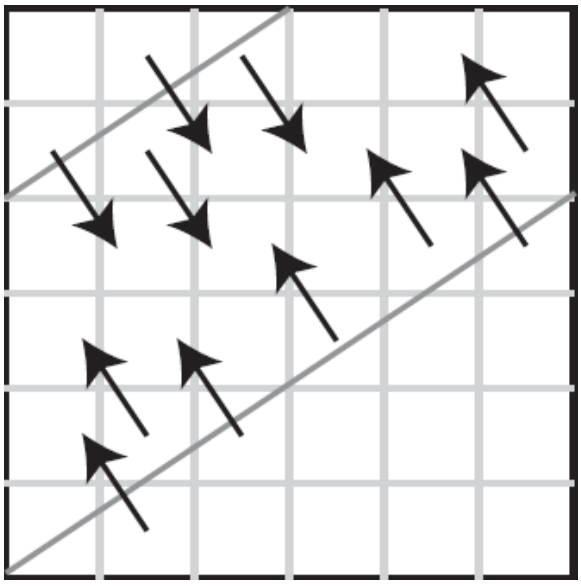


Figure 2. Window with a ditch

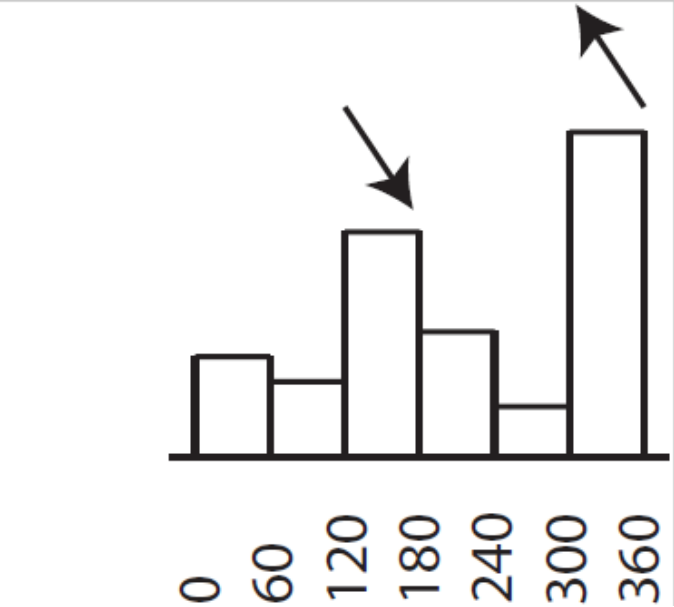


Figure 3. Aspect histogram based on the window

Unlike the existing researches, we do not make assumptions on how the relationship should be for the algorithm. The algorithm is responsible to find the relationships between target features and extracted archeological attributes. The algorithm will generate models based on the relationships and use the model for classification. Therefore, our algorithm is more adaptive than the existing ones. If successful, we can use the same algorithm to find ditches, mounds, and other features with minimal changes.

2. RELATED WORK

There are several researches that are related with our work, such as filtering raw laser data [VOS00], and produce a framework to discover interesting regions [CEL09]. In 2009, Oner Ulvi Celepcikay and Christoph F. Erick published a paper to provide a region regression framework to discover interesting geological regions. In this paper, they proposed a regional regression framework that employs representative-based clustering to discover interesting regions and their associated regional regression functions, without using any predefined boundaries. They also developed two fitness functions: an R-squared-based fitness function and an AIC-based fitness function that are used to guide the search for regions with strong regional linear relationships between the response variable and the independent variables.

In 2000, George Vosselman from Delft University of Technology published a paper to filter raw laser data. This research is to distinguish buildings and vegetation captured in the image based on slope [VOS00]. However, he used a mathematical approach to build the classification model instead of a data mining approach. Therefore, the model is developed to solve a single task.

In 1986, John Canny published a paper to propose a research which produce an edge detector to identify edges with higher accuracy in an image. In the paper, they defined detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. Then they add a third criterion to ensure the detector response only once to a single edge. The criterion was used in numerical optimization to derive detectors for several common image features, including step edges. At last, they derived a single operator shape which is optimal at any scale with the uncertainty principle between

detection and localization performances that was found when specializing the analysis to step edges. This research is based on 2D pictures only.

3. GEOSCIENCE AND ANTHROPOLOGY CONCEPTS

To find ditches, we decided to use aspect, slope, and curvature. Slope is a value that describes both the direction and the steepness of a line. It is often presented using the letter m. In a right handed coordinate system, if the line is increasing from left to right, m is positive, if the line is decreasing from left to right, m is negative. The slope can be calculated using the following equation: $m = \frac{\Delta y}{\Delta x}$. However, in geoscience area, slope is always greater or equal to 0, which equals to the absolute value of m. Aspect is the direction that the slope is facing. It ranges from 0 to 360. Curvature is the amount by which the ditch deviates from being flat or straight. The curvature of a circle with radius R will be large if R is small and small if R is large. Thus the curvature can be defined to be the reciprocal of the radius. Feature means if a specific pixel is part of the target area, whether it is a ditch, a mound, or anything else needs to be classified.

4. DATA MINING CONCEPTS

Data mining models are often used in other systems to accomplish certain purposes, including market research, risk management, and image processing. Here are some terms that are used in data mining area and in this paper:

Attributes: For representing the cleaning, transforming, and aggregating of attributes used as input in the models.

Interfaces and APIs: For linking data mining components with other languages and systems.

Models: For representing data mining and statistical data and output results.

Process: For producing, deploying, and using the models [GRO02].

In this research, we choose Weka to provide data mining support and Java to program the algorithm. Java is used in reading the input, preprocess the data, and output the results. We choose Java because it's one of the most adaptive languages. It can be used on Windows, Mac, and Linux systems. Weka is not only one of the most powerful data mining tools, but also provides an open-source API for Java.

4.1. Classification

In this research, classification is used to find the ditches. It is an important data mining function with broad application that can be used to classify the various kinds of data according to the features of item with respect to the predefined set of classes [PAT13]. A classification process begins with a data set in which the categories are already identified. The classification algorithm will then find the relationships between the attributes and the target. Different classification algorithms use different techniques to find these relationships. A classification model will then be built based on the relationships that are found by the algorithm. After

building the model, different data sets in which the target values are unknown will be input into the model. The model will then categorize the items in each data set into different categories.

A classification algorithm can be tested by comparing the predicted target values to identified target values in a data set. For testing the algorithm, historical data sets can be divided into two groups. Training data sets that are used for building the model and testing data sets that are used for testing the model.

The result of a finished classification process is represented in the form of a 2×2 table represented in Table 1. Each cell contains an integer greater or equal to zero.

Table 1. Result of classification

True Positive value	False Negative value
False Positive value	True Negative value

Meanings of the values:

a. True Positive value (TP): the number of instances that are correctly classified as positive.
b. False Negative value (FN): the number of instances that are wrongly classified as negative.
c. False Positive value (FP): the number of instances that are wrongly classified as positive.
d. True Negative value (TN): the number of instances that are correctly classified as negative.

The following equations are derived from Table 1:

a. $\frac{TP}{TP+FN}$(Recall): the fraction of relevant instances that are retrieved by the model.
b. $\frac{TP}{TP+FP}$(Precision): the fraction of retrieved instances by the model that are relevant.

4.2. Preprocessing

Before putting datasets into Weka to process, they have to be preprocessed into a particular form to satisfy Weka's requirements. Weka can only read an ARFF (Attribute-Relation File Format) file which is an ASCII text file that describes a list of instances sharing a set of attributes. It is consisted of two distinct sections: a Header section and a Data section. The Header section contains a name of the relation, a list of the attributes, and their types. The Data section contains values of the corresponding attributes. An example Header section is shown in Figure 4 and an example Data section is shown in Figure 5. Note that in this example, the target region defined in the Header section as: @ATTRIBUTE class. It contains three different values: Iris-setosa, Iris-versicolor, and Iris-virginica. The lines starting with % are comments.

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Figure 4. Header section

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Figure 5. Data section

4.3. Weka

Weka, or Waikato Environment for Knowledge Analysis, is an open-source free software that holds a collection of machine learning algorithms for data mining. It was developed by University of Waikato, New Zealand. The algorithms inside it can be used directly through the software or being called using APIs with Java code. Weka contains different algorithms for classification, regression, or clustering. It can also visualize the results or the generated classifier while being accessed through the graphical user interface.

Figure 6 shows the graphical user interface of Weka. On the top left corner there's a list of buttons to choose which file needs to be read. After the reading is finished successfully, users can choose their purpose by selecting a tab on the top. The bird on the bottom right corner indicates if the program is doing data mining or not. After the process is finished, results will be displayed in the white textbox above the bird.

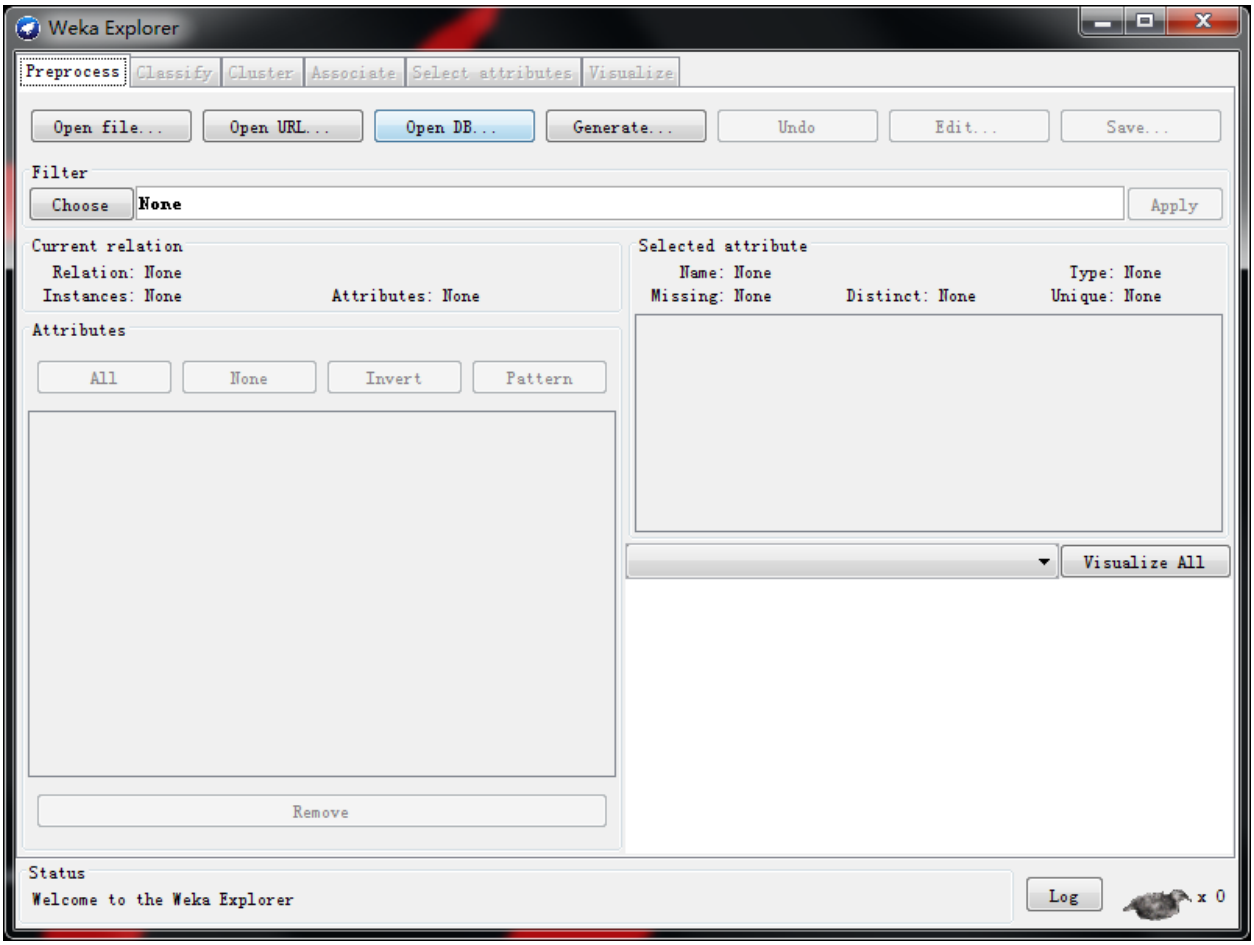


Figure 6. GUI of Weka

4.4. J48 decision tree

A J48 decision tree is an algorithm for producing a binary tree model to classify data sets. It is also called C4.5 decision tree outside Weka. It is a univariate decision tree, which means each internal node is constructed based on one of the many attributes. The whole model has a root node, several leaf nodes, several internal nodes, and branches that connect between higher level nodes and lower level nodes. Each internal node represents a decision based on the feature it contains and each leaf node represents an outcome. After the model is built, each tuple in the data set that needs to be classified will be applied to it. Each tuple starts from the root node, goes downward into internal nodes, and finally ends up inside one of the leaf nodes. The result will be

the outcome of the leaf node it ends up in. J48 decision tree is considered as one of the most useful approaches in classification problems [PAT13]. It can provide several advantages over other decision trees:

1. J48 decision tree can handle both continuous and discrete attributes. In order to handle continuous attributes, it creates a threshold and then splits the list into those whose attribute values are above the threshold and those that are less or equal to it.
2. J48 decision tree allows attribute values to be marked as "?" for missing. Missing attribute values are simply not used in gain and entropy calculations.
3. J48 goes back through the tree once it has been created and attempts to remove branches that do not help by replacing them with leaf nodes [SIN14].

Figure 7 shows a part of a J48 decision tree. The root node is the leftmost node which starts at curve_positive attribute. If the value of this attribute is greater than 37, it will go to the next node on the right and continue. If the value is smaller or equal to 37, it will go straight down until it meets the node with the decision: curve_positive <= 37. It will then go to the next right node and continue.

```
curve_positive > 37
|   slope_1 <= 9
|   |   aspect_180 <= 0: false (18523.0/4.0)
|   |   aspect_180 > 0
|   |   |   aspect_180 <= 1
|   |   |   |   curve_significantly_positive <= 13: false (284.0)
|   |   |   |   curve_significantly_positive > 13
|   |   |   |   |   slope_4 <= 41: false (41.0)
|   |   |   |   |   slope_4 > 41
|   |   |   |   |   |   slope_6 <= 2
|   |   |   |   |   |   |   curve_significantly_negative <= 87
|   |   |   |   |   |   |   |   aspect_135 <= 0
|   |   |   |   |   |   |   |   |   curve_significantly_positive <= 14: false (6.0)
|   |   |   |   |   |   |   |   |   curve_significantly_positive > 14
|   |   |   |   |   |   |   |   |   |   slope_2 <= 15: true (15.0/1.0)
|   |   |   |   |   |   |   |   |   |   slope_2 > 15: false (12.0/2.0)
|   |   |   |   |   |   |   |   aspect_135 > 0: true (3.0)
|   |   |   |   |   |   |   curve_significantly_negative > 87: false (8.0)
|   |   |   |   |   |   slope_6 > 2: false (4.0)
|   |   |   aspect_180 > 1
|   |   |   |   slope_1 <= 6: false (2136.0/1.0)
|   |   |   |   slope_1 > 6
|   |   |   |   |   curve_positive <= 75: false (380.0/3.0)
|   |   |   |   |   curve_positive > 75
|   |   |   |   |   |   aspect_360 <= 191
|   |   |   |   |   |   |   slope_2 <= 56: false (3.0)
|   |   |   |   |   |   |   slope_2 > 56: true (5.0)
|   |   |   |   |   |   aspect_360 > 191: false (41.0)
|   slope_1 > 9
|   |   slope_5 <= 40
```

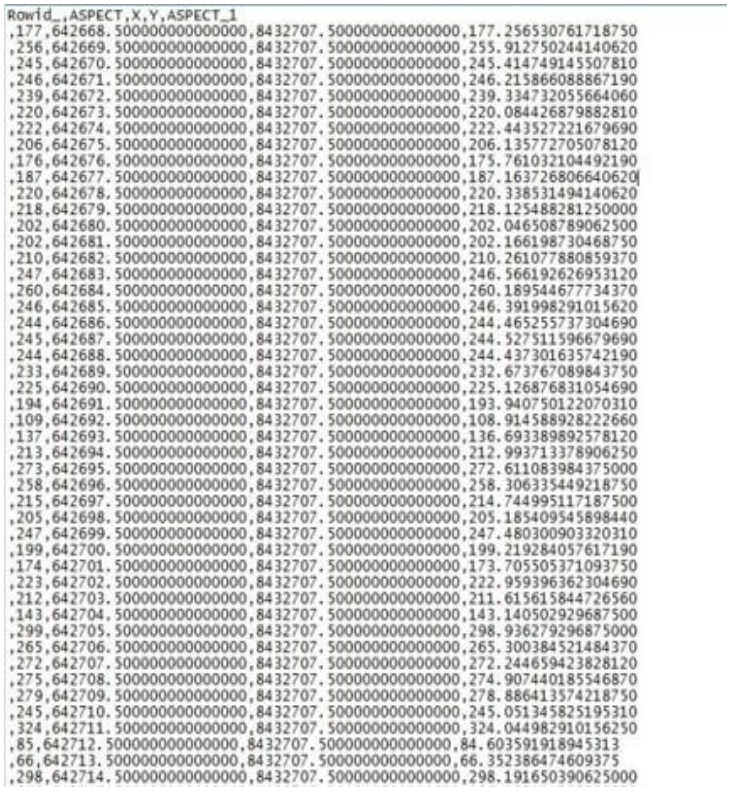
Figure 7. Part of J48 decision tree

5. ALGORITHMS

5.1. J48 decision tree

The reason first reason for using J48 decision tree is that we are using multiple independent variables to predict the target value. Secondly, there are relationships among the independent variables, some of the variables may need to be checked more than once depending on the state of other variables. For example, a window will be classified as containing ditch (true) if more than 100 pixels inside it have the aspect value between 45 and 90 degrees and less than 80 pixels have the curve value that is significantly negative, or it contains less than 70 pixels with significantly negative curve value and more than 50 pixels with aspect value between 0 and 45 degrees. In this example, the variable of significantly negative is checked twice to get the correct results. Taking away any decisions in the above example will give a different result.

In order to classify the data, four text files will be extracted from each map. Each file contains one of the data described in Chapter 3. Every file has five columns. The first column is the ID of the rows. The second column is the rounded value of the data, which is an integer. The third column is the x coordinate of the pixel. The forth column is the y coordinate of the pixel. The fifth column is the exact value of the data. Only last three columns are used in this paper. These text files will be used as input files for the algorithm to convert into a single file that can be processed by Weka. Part of an aspect file derived from a map is presented in Figure 8.



```
Rowid_,ASPECT,X,Y,ASPECT_1
,177,642668.500000000000000,8432707.500000000000000,177.256530761718750
,256,642669.500000000000000,8432707.500000000000000,255.912750244140620
,245,642670.500000000000000,8432707.500000000000000,245.414749145507810
,246,642671.500000000000000,8432707.500000000000000,246.215866088867190
,239,642672.500000000000000,8432707.500000000000000,239.334732055664060
,220,642673.500000000000000,8432707.500000000000000,220.084426879882810
,222,642674.500000000000000,8432707.500000000000000,222.443527221679690
,206,642675.500000000000000,8432707.500000000000000,206.135772705078120
,176,642676.500000000000000,8432707.500000000000000,175.761032104492190
,187,642677.500000000000000,8432707.500000000000000,187.163726806640620
,220,642678.500000000000000,8432707.500000000000000,220.338531494140620
,218,642679.500000000000000,8432707.500000000000000,218.125488281250000
,202,642680.500000000000000,8432707.500000000000000,202.046508789062500
,202,642681.500000000000000,8432707.500000000000000,202.166198730468750
,210,642682.500000000000000,8432707.500000000000000,210.261077880859370
,247,642683.500000000000000,8432707.500000000000000,246.566192626953120
,260,642684.500000000000000,8432707.500000000000000,260.189544677734370
,246,642685.500000000000000,8432707.500000000000000,246.391998291015620
,244,642686.500000000000000,8432707.500000000000000,244.465255737304690
,245,642687.500000000000000,8432707.500000000000000,244.527511596679690
,244,642688.500000000000000,8432707.500000000000000,244.437301635742190
,233,642689.500000000000000,8432707.500000000000000,232.673767089843750
,225,642690.500000000000000,8432707.500000000000000,225.126876831054690
,194,642691.500000000000000,8432707.500000000000000,193.940750122070310
,109,642692.500000000000000,8432707.500000000000000,108.914588928222660
,137,642693.500000000000000,8432707.500000000000000,136.693389892578120
,213,642694.500000000000000,8432707.500000000000000,212.993713378906250
,273,642695.500000000000000,8432707.500000000000000,272.611083984375000
,258,642696.500000000000000,8432707.500000000000000,258.306335449218750
,215,642697.500000000000000,8432707.500000000000000,214.744995117187500
,205,642698.500000000000000,8432707.500000000000000,205.185409545898440
,247,642699.500000000000000,8432707.500000000000000,247.480300903320310
,199,642700.500000000000000,8432707.500000000000000,199.219284057617190
,174,642701.500000000000000,8432707.500000000000000,173.705505371093750
,223,642702.500000000000000,8432707.500000000000000,222.959396362304690
,212,642703.500000000000000,8432707.500000000000000,211.615615844726560
,143,642704.500000000000000,8432707.500000000000000,143.140502929687500
,299,642705.500000000000000,8432707.500000000000000,298.936279296875000
,265,642706.500000000000000,8432707.500000000000000,265.300384521484370
,272,642707.500000000000000,8432707.500000000000000,272.244659423828120
,275,642708.500000000000000,8432707.500000000000000,274.907440185546870
,279,642709.500000000000000,8432707.500000000000000,278.886413574218750
,245,642710.500000000000000,8432707.500000000000000,245.051345825195310
,324,642711.500000000000000,8432707.500000000000000,324.044982910156250
,85,642712.500000000000000,8432707.500000000000000,84.603591918945313
,66,642713.500000000000000,8432707.500000000000000,66.352386474609375
,298,642714.500000000000000,8432707.500000000000000,298.191650390625000
```

Figure 8. Part of an aspect file

In this research, a window-based technique is used to integrate the text files. Since all text files are extracted from a rectangle shaped map, a rectangle shaped window is used. The window starts at the top left corner of the map and moves exactly one pixel from left to right covering the same set of rows. After it reaches the rightmost position, the window moves down a row and starts from the leftmost column again. Each time the window moves, the data inside it will be integrated and stored into a single data set.

5.2. Window

Sliding window technique is the backbone of this algorithm. It is used for dividing the data of a whole map into small pieces to produce histograms for classification. First of all, a whole map will be divided into several windows, each containing a fixed number of pixels. The window size will not change during model building and testing phases. For a window with 15

rows and 10 columns, the total number of pixels inside each window will always be 150 and it can't be changed during the process. The reason behind this is that if the size can be changed for each window, the values inside the columns of the histograms will also be changed. Changing the values would have a heavy influence on the J48 decision tree because the tree itself uses the values to classify the target features as shown in Figure 5. This also means the window size of testing data sets has to be the same as the window size of training data sets. The first window will be created at the top left corner of the map, which means the coordinate of the top left corner of the window is the same as the top left corner of the map. The algorithm will then record all the data inside this window. After the recording process, the window will be moved by exactly one pixel to the right, covering the same set of rows. Another recording process will then begin. When the right side of the window touches the right edge of the map, the last recording process of this row will begin. After the recording finishes, the window will move down a row and to the leftmost edge of the map. This loop will continue until the window reaches the bottom right corner of the map. At this time, all the data inside every possible window with the fixed window size inside the map is recorded. During this research, we created a customized variable type called pixel as the base.

Pseudo code to define customized variable type: pixel

```
Type Pixel
Dim aspect As BIGDECIMAL
Dim slope As BIGDECIMAL
Dim curve As BIGDECIMAL
Dim feature As INTEGER
End Pixel
```

Pseudo code to read text files

ReadFile

Dim pixels As Pixel[][]

Dim filepath As STRING

Dim inputfile As FILE

Dim column As INTEGER

Dim row As INTEGER

Dim x As INTEGER

Dim y As INTERGER

Dim max_x As BIGDECIMAL

Dim max_y As BIGDECIMAL

Dim min_x As BIGDECIMAL

Dim min_y As BIGDECIMAL

filepath ← "input file path"

inputfile ← new File(filepath)

max_x ← inputFile.return(max x coordinate)

max_y ← inputFile.return(max y coordinate)

min_x ← inputFile.return(min x coordinate)

min_y ← inputFile.return(min y coordinate)

column ← max_x-min_x

row ← max_y-min_y

pixels.length ← row

pixels[].length ← column

for each line in inputFile do

x ← line.return(x)-min_x

y ← line.return(y)-min_y

pixels[y][x].setaspect(line.returnaspect())

pixels[y][x].setcurve(line.returncurve())

pixels[y][x].setslope(line.returnslope())

pixels[y][x].setfeature(line.returnfeature())

end for

This moving window technique results in overlap between window areas. Considering overlapping windows is important because a ditch can be in any shape and occur anywhere inside a map, if the overlapping is avoided, it is possible to ignore critical conditions that can efficiently classify the ditch. For example, in Figure 9, black lines surround the target ditch, and blue lines and the lines mark the six windows produced by different algorithms. As it is shown, all the windows contains part of the target ditch. However, if the algorithm is a non-overlapping one, it might produce the four windows marked by blue lines. After separating them in Figure 10, the ditch parts can hardly be identified except the bottom left one. If we are using an overlapping approach, we can produce every windows including the two marked with red lines. In Figure 11, the parts inside red windows are more likely to be identified as ditches. Therefore, using overlapping approach to avoid this loss of information.

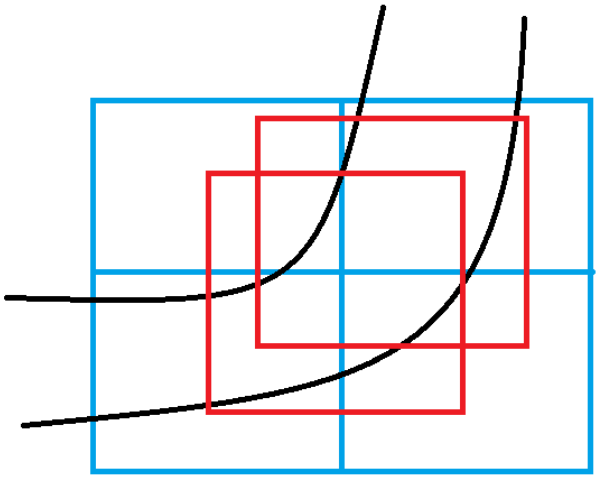


Figure 9. Ditch example

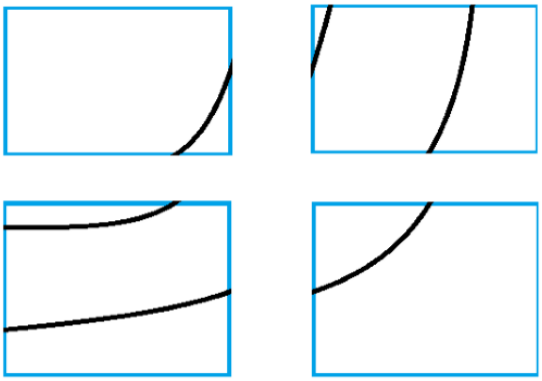


Figure 10. Separated blue windows

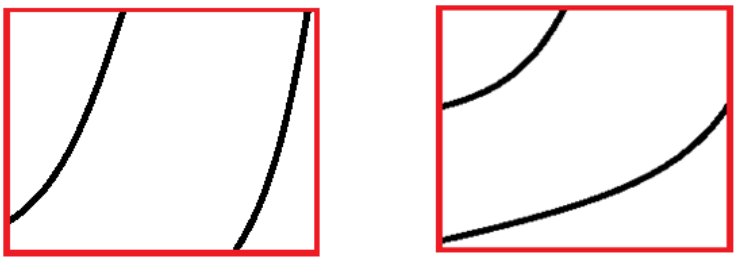


Figure 11. Separated red windows

5.3. Histogram

One histogram will be produced for each of the three attributes inside each window. For the values of aspect, it is reasonable to divide the ranges inside the histogram equally based on the range of aspect values. In the first approach, the aspect histogram has 6 columns, each column covers 60 degrees: 0 to 60, 60 to 120, 120 to 180, 180 to 240, 240 to 300, and 300 to 360. For slope, first all the slope data in the whole map will be sorted. Then the data will be divided into six parts will equal number of elements. For example, an ordered array like: 0, 1, 3, 6, 8, 15, 25, 30, 31, 35, 50, 100 will be divided into the following parts: 0 to 1, 3 to 6, 8 to 15, 25 to 30, 31 to 35, and 50 to 100. For curvature, the histogram has five parts: significantly negative, negative, around zero, positive, and significantly positive. Significantly negative covers the range less than -20, negative covers the range from -20 to -5, around zero covers the range from-5 to 5, positive covers the range from 5 to 20, and significantly positive covers the range greater than 20.

The value of each attribute of each pixel inside a window will be distributed into the histogram column corresponding to the attribute. For example, if a pixel has the following attributes:

1. Aspect: 100
2. Slope: 10
3. Curvature: 3

It will add one to the current value of the second column of the aspect histogram, and the third column of the curvature histogram in the windows this pixel belongs to. For slope histogram, the column that needs to be changed varies based on the results of calculation. In Figure 12, 13, and 14, histograms based on aspect, slope and curvature are shown.

Pseudo code to define customized variable type: Histogram

```
Type Histogram
Dim aspect As INTEGER[8]
Dim slope As INTEGER[6]
Dim curve As INTEGER[5]
Dim feature As BOOLEAN
End Histogram
```

The function takes only one input argument, which is a pixel type two-dimensional array, and output a histogram type variable. Pseudo code to generate histogram

```
Generate Histogram
Generate_Histogram(Pixel[][] source)
Dim slopes As BIGDECIMAL[]
Dim results As BIGDECIMAL[]
Dim h As Histogram[]
Dim i As INTEGER
i ← 0
for Pixel p in source
    slopes[i] ←p.getslope()
    i ← i+1
end for
results ← Normalize(slopes)
for Pixel p in source
    i ← calculate_index(p.get_x(),p.get_y())
```

h[i].set_aspect(p.get_aspect())

h[i].set_curve(p.get_curve())

h[i].set_slope(p.get_slope(), results)

h[i].set_feature(p.get_feature)

end for

The set functions are responsible to do some basic calculations, such as calculate values in curvature histogram and slope histogram. However, the normalization process will be completed in another function. Pseudo code of a list of set functions in Histogram

set_slope(BIGDECIMAL source, BIGDECIMAL normalized_value)

Dim k As INTEGER

for INTEGER i from 0 to normalized_value.length

k ← i

if source <= normalized_value[i]

break from loop

end if

end for

slope[k] ← slope[k]+1

set_curve(BIGDECIMAL source)

if -5<i<5

curve[0] ← curve[0]+1

end if

if -20<=i<=-5

curve[1] ← curve[1]+1

```
end if
if i<=20
        curve[2] ← curve[2]+1
end if
if 5<=i<=20
        curve[3] ← curve[3]+1
end if
if 20<i
        curve[4] ← curve[4]+1
end if
set_feature(INTEGER source)
if(source!=0)
        feature ← true
set_aspect(BIGDECIMAL source)
Dim i As BIGDECIMAL ← 45
Dim temp As BIGDECIMAL ← source.divide(i, 1, BigDecimal.ROUND_CEILING)
int k ← 0
if temp.remainder(1)=0 and temp!=0
        k ← temp-1
else
        k ← temp.ROUND_FLOOR
end if
aspect[k] ← aspect[k]+1
```

Normalize function for slope data is responsible for only sort the values and divide it into 6 parts with equal number of elements. It will return an array which is consist of the last element in each part. Pseudo code to normalize slope

```
Normalize function for slope
BIGDECIMAL[] Normalize_Slope(BIGDECIMAL slope[])
Dim results[] As BIGDECIMAL[6]
Dim field As INTEGER
Dim sorted_slope As BIGDECIMAL[]
sorted_slope ← sort(slope)
field ← sorted_slope.length/6
for INTEGER i from 1 to 6
    results[i-1] ← sorted_slope[field*i]
end for
return results[]
```

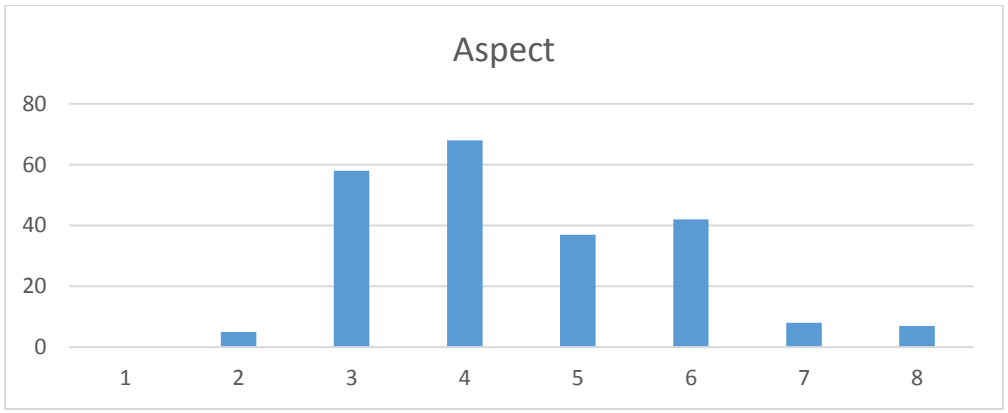


Figure 12. The aspect histogram

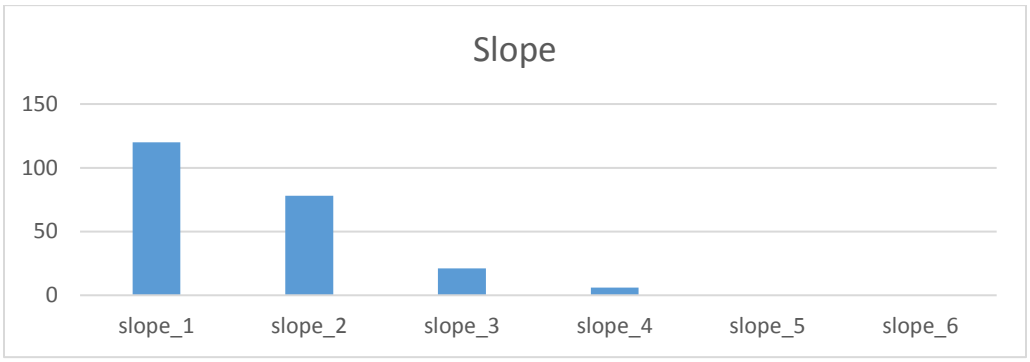


Figure 13. The slope histogram

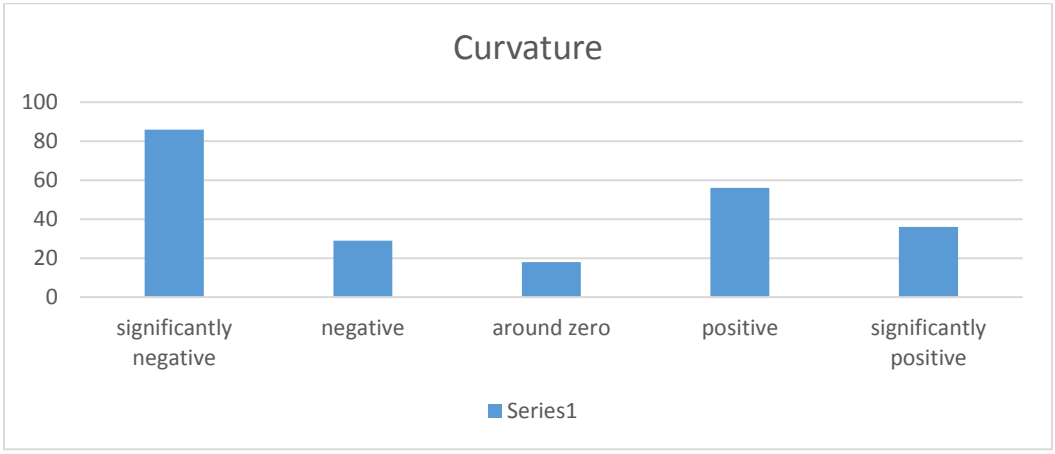


Figure 14. The curvature histogram

The initial design did not have a high enough accuracy so we improved the resolution of aspect values. So instead of using a six column histogram for aspects, an eight column one is used. Each covers 45 degree. However, a ditch can face any direction. A model built based on a ditch facing northeast will not catch the critical information for a dataset whose ditch faces northwest. The reason is that without normalizing aspect values, the pattern in the model will be totally different with the pattern in the testing data set. To minimize this affect, two histograms are prepared. Both of them have eight columns. One of them starts from 0 degree. The other one starts from 22.5 degree and covers the region from 337.5 to 360 plus 0 to 22.5 degree for the last column. After recording them, the minimal values of both histograms will be found and compared. Then the one with the smallest minimal value will be used for further processing. After identifying the histogram that will be using, the algorithm will then shift the columns to let the first column be the column with the minimal value. The steps are shown in the following pictures:

1. Get both aspect histograms:

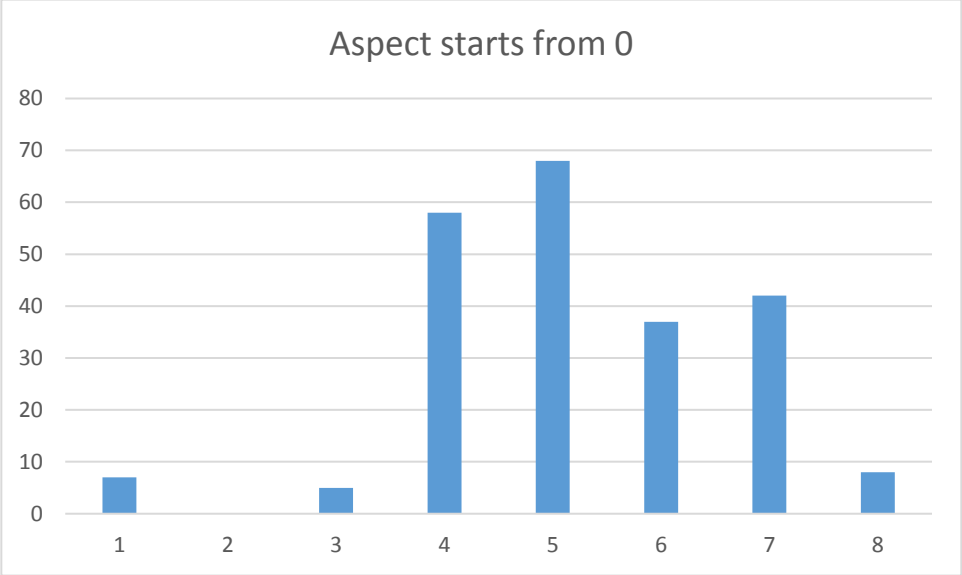


Figure 15. First aspect histogram

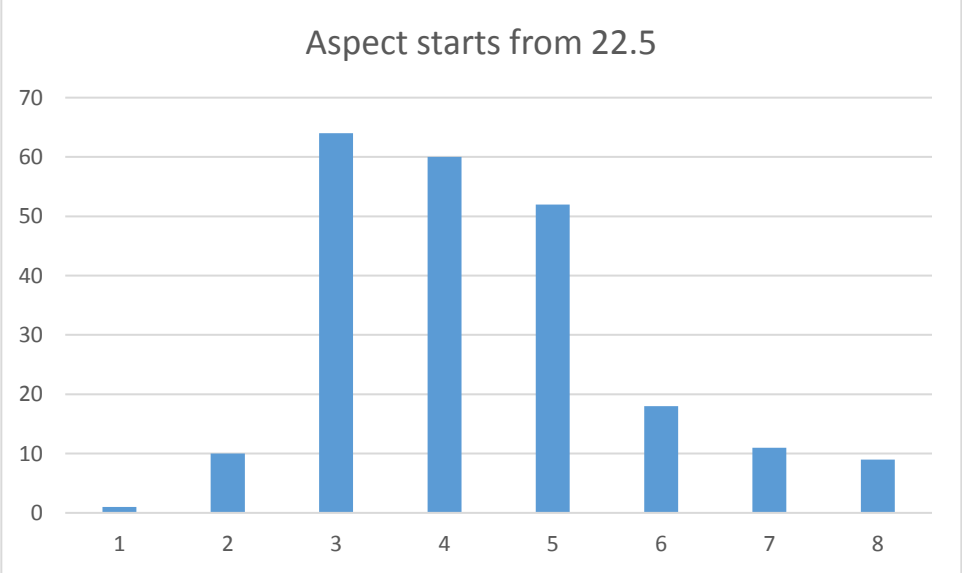


Figure 16. Second aspect histogram

2. Find the minimal value of both histograms. In this case the histogram starts from 0 degree has the smallest minimum on its second column: 0.
3. Shift the columns of the histogram. Keep the original order.

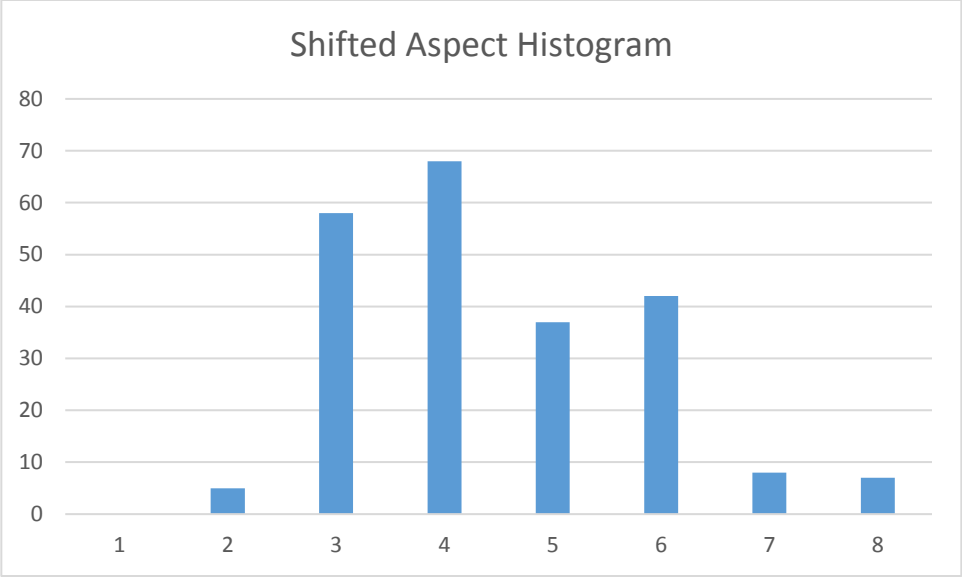


Figure 17. Final aspect histogram

Pseudo code to apply double aspect histograms approach:

In order to make two histograms based on aspect, we need to change the set_aspect method in the previous pseudo code to store the exact values of the pixels inside a window instead of storing a histogram. Change set_aspect function to:

set_aspect(INTEGER[] source)

aspect ← source

Then we need to add additional functions to produce a histogram with the aspect values which contains 16 columns, each covers 22.5 degree. The pseudo code for this process is:

set_tempaspect(BIGDECIMAL source)

Dim i As BIGDECIMAL ← 22.5

Dim temp As BIGDECIMAL ← source.divide(i, 1, BigDecimal.ROUND_CEILING)

int k ← 0

if temp.remainder(1)=0 and temp!=0

    k ← temp-1

else

k ← temp.ROUND_FLOOR

end if

Based on the 16-column histogram, we can produce two 8-column histograms. For example, A is the 16-column histogram, B and C are two 8-column histograms. A[n], B[n], and C[n] represent the values in the n+1th column in histogram A, B, and C.

To produce B, we use the following equation: $B[n] = \begin{cases} A[15] + A[0], & n = 0 \\ A[2n-1] + A[2n], & 0 < n < 8 \end{cases}$.

To produce C, the equation will be: $C[n] = A[2n] + A[2n+1], 0 \leq n < 8.$

Double aspect histogram approach:

Refine aspect

Histogram[] refine_aspect(Histogram[] source)

Dim aspect As INTEGER[8]

for INTEGER i from 0 to source.length

aspect ← choose_aspect(source[i])

source[i].set_aspect(aspect)

end for

return source

INTEGER[] choose_aspect(Histogram source)

Dim result As INTEGER[8]

Dim temp As INTEGER[]

Dim tempasp1 As INTEGER[8]

Dim temasp2 As INTEGER[8]

Dim normalizedtemp1 As INTEGER[8]

```
Dim normalizedtemp2 As INTEGER[8]
Dim a As INTEGER
Dim b As INTEGER
Dim c As INTEGER
a ← 15
b ← 0
c ← 1
temp ← source.get_tempaspect()
for INTEGER j from 0 to tempasp1.length
        if j>=1
                a ← j*2-1
                b ← j*2
                c ← j*2+1
        end if
        tempasp1[j] ← source[a]+source[b]
        tempasp2[j] ← source[b]+source[c]
end for
normalizedtemp1 ← normalize(tempasp1)
normalizedtemp2 ← normalize(tempasp2)
if normalizedtemp1[0] < normalizedtemp2[0]
        result ← normalizedtemp1
else
        result ← normalizedtemp2
```

end if

return result

The normalize method in the above pseudo code will shift the columns in histogram X so that X[0] will have the minimum value among all the columns in X. But it will not change the order of the columns in X.

5.4. Target value

There are always multiple pixels inside a window, some of them may be part of the ditch, others may not. A window may be considered as part of the ditch if any pixel is part of a ditch. But this causes a lot of false positives because a lot of information that might not define a ditch might be considered as critical information. A lot of windows that contain only one pixel belongs to a ditch are treated equally important as the windows fully belong to a ditch. So the algorithm was changed to that only the center pixel inside a window is used to define if the whole window belongs to a ditch. Therefore, it ensures that a window will contain more pixels that belongs to a ditch to be identified as a target. The number of true positive decreases inevitably by restrict the conditions. But the decrease rate of false positives is much higher than true positives.

5.5. Write the file

The data has to be in a specific form in order to put it into Weka and analyze. Before putting the information into Weka, histograms of all windows inside a map must be written into an ARFF file. In an ARFF file, the structure and types of data contained are defined in the first part. The data starts after the mark @Data. Each row represents a set of histograms of a window. In this research, the first eight columns contain the value of aspect histogram. The following five columns contain the value of curvature histogram. Then there are six columns contain the value of slope histogram. The last column contains the target value of the window. For all the

unpredicted or testing data sets, the target value of every row is set to false. For training data sets, the target values are written as being processed in the previous section. Figure 18 shows a screenshot of a training data set.

```
% This is a test
@Relation testoutTraining
@ATTRIBUTE aspect_45 NUMERIC
@ATTRIBUTE aspect_90 NUMERIC
@ATTRIBUTE aspect_135 NUMERIC
@ATTRIBUTE aspect_180 NUMERIC
@ATTRIBUTE aspect_225 NUMERIC
@ATTRIBUTE aspect_270 NUMERIC
@ATTRIBUTE aspect_315 NUMERIC
@ATTRIBUTE aspect_360 NUMERIC
@ATTRIBUTE curve_significantly_negative NUMERIC
@ATTRIBUTE curve_negative NUMERIC
@ATTRIBUTE curve_around_zero NUMERIC
@ATTRIBUTE curve_positive NUMERIC
@ATTRIBUTE curve_significantly_positive NUMERIC
@ATTRIBUTE slope_1 NUMERIC
@ATTRIBUTE slope_2 NUMERIC
@ATTRIBUTE slope_3 NUMERIC
@ATTRIBUTE slope_4 NUMERIC
@ATTRIBUTE slope_5 NUMERIC
@ATTRIBUTE slope_6 NUMERIC
@ATTRIBUTE class {true,false}

@Data

0,2,2,4,9,10,27,171,196,15,0,14,0,86,54,50,27,8,0,true
0,2,2,3,10,10,27,171,192,18,0,15,0,86,46,49,29,15,0,true
0,2,2,2,8,10,27,174,185,19,0,21,0,83,39,50,32,21,0,true
1,2,2,2,9,10,27,172,182,20,0,23,0,79,35,48,40,23,0,true
1,1,2,2,7,10,26,176,178,21,0,26,0,77,34,49,42,23,0,true
1,2,2,2,6,8,29,175,176,22,0,27,0,74,35,51,42,23,0,true
2,3,3,4,6,8,29,170,175,23,0,27,0,72,37,52,41,23,0,true
2,3,3,4,5,8,30,170,176,22,0,27,0,70,39,54,40,22,0,true
2,3,4,4,4,8,29,171,178,21,0,26,0,66,42,57,40,20,0,true
2,3,4,4,6,8,29,169,178,21,0,26,0,61,45,59,42,18,0,false
0,1,3,6,9,9,28,169,180,19,0,26,0,54,48,61,44,18,0,false
0,1,3,6,8,9,24,174,182,17,0,26,0,47,47,67,46,18,0,false
0,1,3,6,7,9,16,183,184,16,0,25,0,41,53,68,45,18,0,false
0,0,3,6,7,9,9,191,183,16,0,26,0,39,55,70,44,17,0,false
0,0,3,5,5,7,7,198,185,15,0,25,0,38,55,72,44,16,0,false
1,1,3,4,5,7,9,195,185,14,0,26,0,38,53,75,44,15,0,false
1,2,3,4,5,6,10,194,185,13,0,27,0,40,53,75,45,12,0,false
1,2,3,4,6,6,12,191,189,13,0,23,0,41,55,73,47,9,0,false
1,2,2,4,6,6,13,191,190,13,0,22,0,43,54,76,42,10,0,false
1,2,2,4,6,6,14,190,192,13,0,20,0,41,51,77,42,14,0,false
1,1,2,2,6,6,14,193,193,13,0,19,0,41,47,74,49,14,0,false
0,1,2,2,5,5,12,198,195,13,0,17,0,40,44,75,52,14,0,false
1,1,2,3,6,6,12,194,196,13,0,16,0,41,43,77,49,15,0,false
1,2,3,4,6,6,12,191,195,14,0,16,0,42,42,77,49,15,0,false
1,2,4,4,5,7,10,192,195,14,0,16,0,44,41,78,47,15,0,false
1,3,3,4,5,9,10,190,195,14,0,16,0,47,41,77,44,16,0,false
2,2,3,6,6,10,10,186,193,16,0,16,0,45,41,75,45,19,0,false
2,2,6,6,6,10,10,183,189,18,0,18,0,43,36,74,49,23,0,false
2,2,6,6,7,9,11,182,188,19,0,18,0,39,36,70,54,26,0,false
2,3,6,7,7,9,12,179,186,21,0,18,0,36,34,68,59,28,0,false
2,3,5,6,6,8,12,183,189,19,0,17,0,32,35,65,63,30,0,false
2,3,4,5,5,9,13,184,189,20,0,16,0,28,35,65,67,30,0,false
2,3,3,4,4,9,13,187,185,26,0,14,0,29,38,66,63,29,0,false
```

Figure 18. A screenshot of a training dataset

5.6. Calculate the result

After we have both the training and testing data sets, it is time to put them into Weka to calculate the results. Since Java is the language being used, we automated the process by using Weka-Java API. The Java algorithm calls the functions from the API and uses the data sets as input. After getting the results, it will write the coordinates of the center points of the Windows that are being classified as containing part of the ditch.

6. EXPERIMENTS AND RESULTS

We did the experiments on 6 data sets and get the result in the table below.

Table 2. Results of experiments

Site	Recall	Precision	Weighted Average Recall	Weighted Average Precision
Biesterfieldt	24.6%	4.9%	91.8%	97.4%
LaRMount	2.9%	1.7%	96.0%	97.0%
Nelson	4.1%	0.7%	92.1%	97.5%
Peterson	2.0%	2.6%	95.6%	95.1%
Shea	5.0%	23.1%	97.1%	95.6%
Sprunk	6.9%	3.2%	97.5%	98.4%

Weighted average recall for each data set is calculated using the following equation:

$$Weighted\ Average\ Recall$$

$$= \left[\frac{TP}{TP+FN} * (TP+FP) + \frac{TN}{TN+FP} * (TN+FN)\right] / (TP+FP+FN+TN)$$

Weighted average precision for each data set is calculated using the following equation:

$$Weighted\ Average\ Precision$$

$$= \left[\frac{TP}{TP+FP} * (TP+FP) + \frac{TN}{TN+FN} * (TP+FP)\right] / (TP+FP+FN+TN)$$

From the results we can see that weighted average precision and recall are both above 90% for all the testing data sets. That means above 90% of all instances are correctly classified. However, recall and precision are much lower comparing to weighted average recall and precision. This indicates that in every data set, the area that isn't part of a ditch is much larger than the area that is part of a ditch.

Among all the data sets, Biesterfieldt has above average recall and Shea has above average precision. For Biesterfieldt data set, we have a higher recall but a low precision, which means the instances that are classified as positive cover 24.6% of the target field but 95.1% of the instances belongs to the area that is not a target field. For Shea data set, we have a 5.0% recall and 23.1% precision, which means the instances that are classified as positive only covers 5.0% of the target field but only 76.9% belongs to the area that is not a target field.

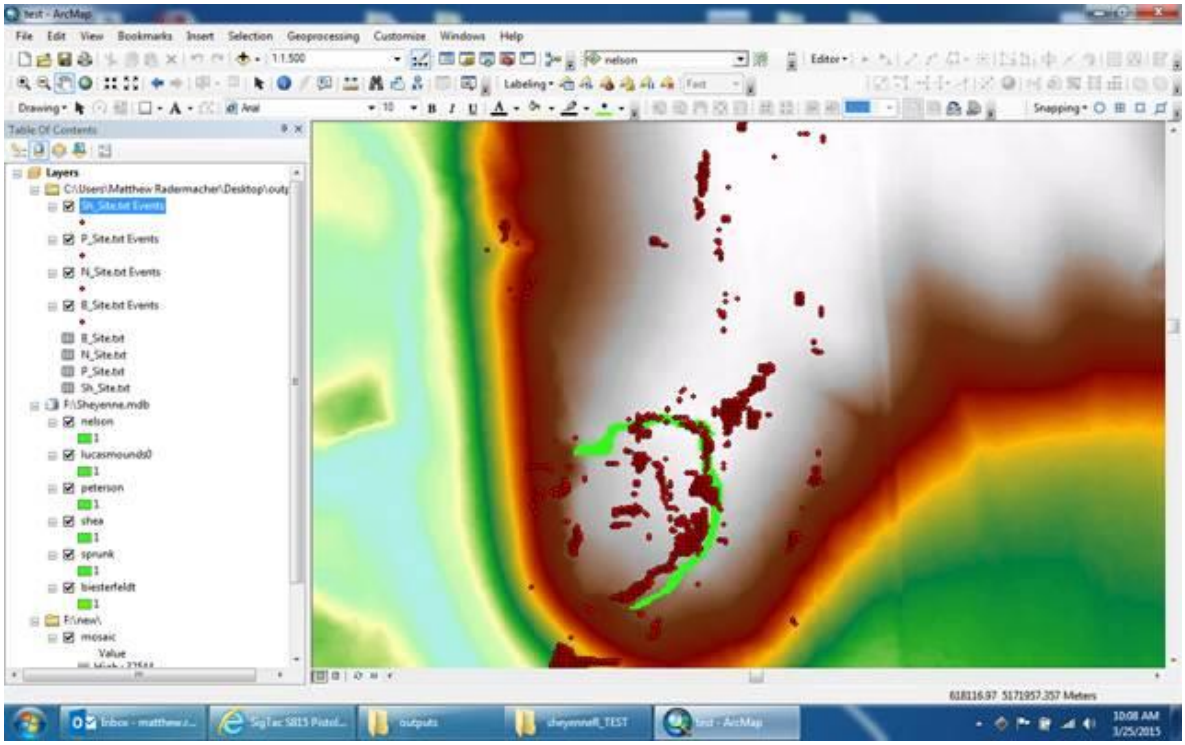


Figure 19. Shea site image with result

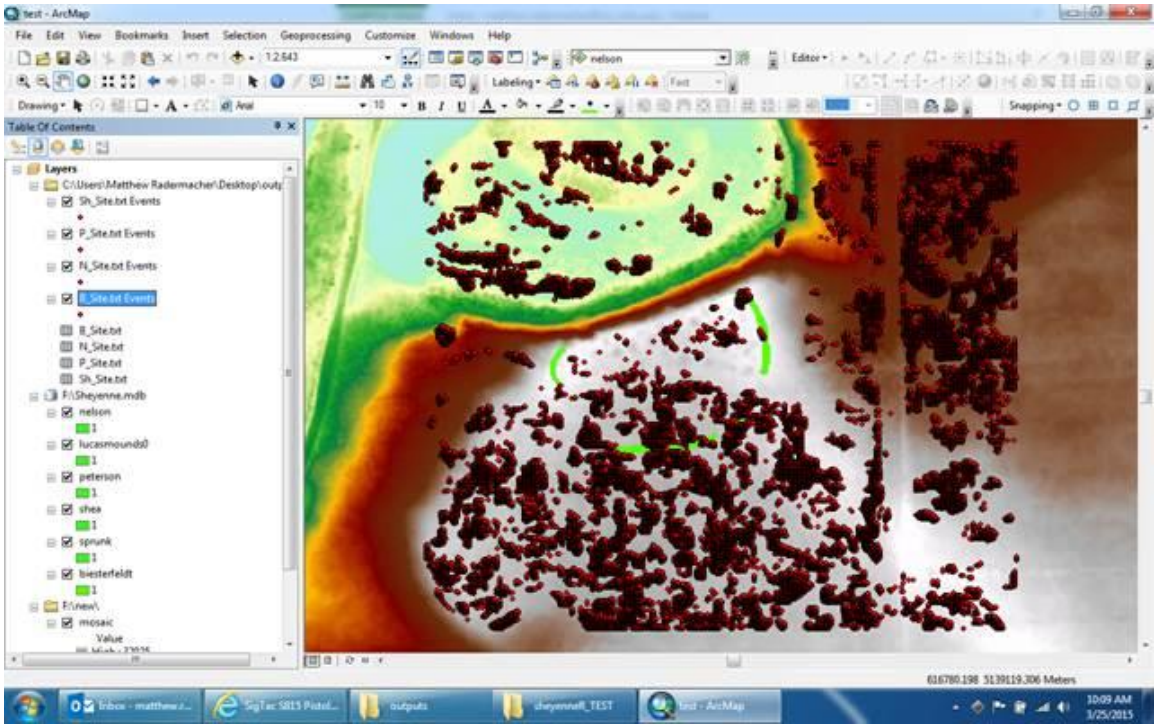


Figure 20. Biesterfeldt site image with result

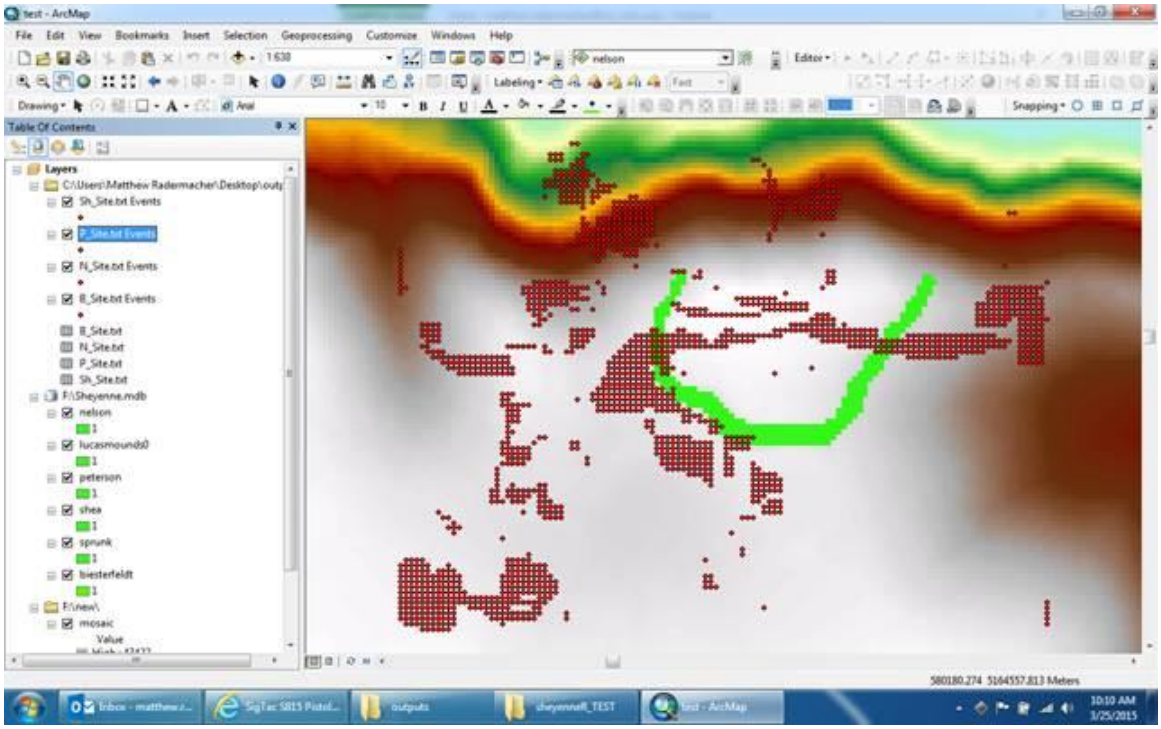


Figure 21. Peterson site image with result

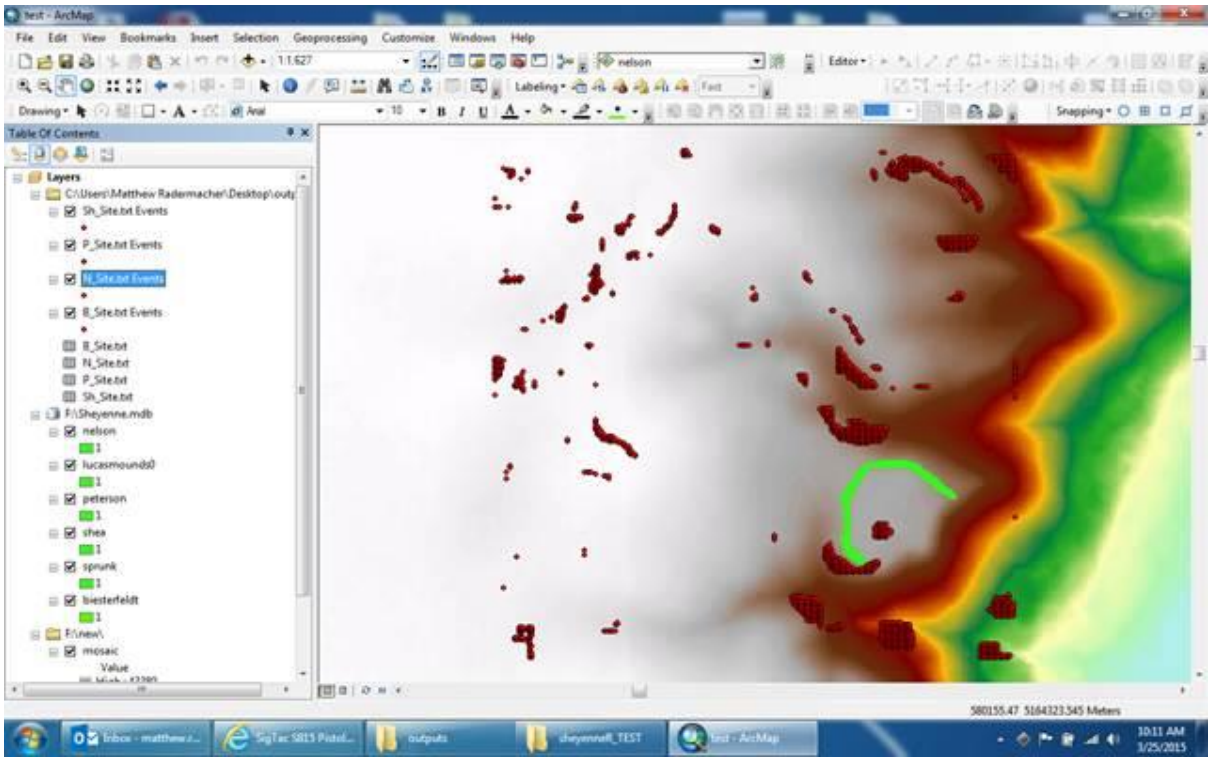


Figure 22. Nelson site image with result

As shown in the figures below, 5 of the 6 trees start from normalized slope data, continue to curvature data and finally the aspect data. Based on this pattern, slope has the most significant influence on identifying the ditches, curvature has less influence, and aspect data has the least influence. The pattern is different from what we expect but since building models are fully automated by Weka and we don't have full control on the process, it is difficult for us to change the model into expected pattern and classify the data. In the Appendix, a full decision tree model for classifying Shea site data is included. We include this tree because the site has the best result to identify the ditch area.

```
slope_5 <= 29
|   slope_3 <= 133
|   |   curve_positive <= 61
|   |   |   curve_significantly_positive <= 102
|   |   |   |   slope_5 <= 16
|   |   |   |   |   slope_6 <= 223
|   |   |   |   |   |   curve_positive <= 49
|   |   |   |   |   |   |   slope_3 <= 113
|   |   |   |   |   |   |   |   curve_significantly_negative <= 192
|   |   |   |   |   |   |   |   |   curve_negative <= 60
|   |   |   |   |   |   |   |   |   |   aspect_90 <= 182: false (161646.0/58.0)
|   |   |   |   |   |   |   |   |   |   aspect_90 > 182
|   |   |   |   |   |   |   |   |   |   |   slope_6 <= 0
|   |   |   |   |   |   |   |   |   |   |   |   slope_3 <= 100: false (54147.0/30.0)
|   |   |   |   |   |   |   |   |   |   |   |   slope_3 > 100
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_270 <= 5: false (1511.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_270 > 5
|   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_4 <= 44: false (377.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_4 > 44
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   curve_negative <= 19
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_225 <= 6: false (14.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_225 > 6: true (8.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   curve_negative > 19: false (67.0)
|   |   |   |   |   |   |   |   |   |   |   slope_6 > 0
```

Figure 23. Biesterfieldt tree

```
curve_positive <= 28
|   curve_significantly_positive <= 106
|   |   slope_2 <= 3: false (68964.0)
|   |   slope_2 > 3
|   |   |   slope_3 <= 133
|   |   |   |   curve_significantly_positive <= 102
|   |   |   |   |   slope_1 <= 1
|   |   |   |   |   |   aspect_45 <= 10
|   |   |   |   |   |   |   slope_5 <= 26
|   |   |   |   |   |   |   |   aspect_90 <= 169
|   |   |   |   |   |   |   |   |   curve_significantly_negative <= 22
|   |   |   |   |   |   |   |   |   |   slope_5 <= 6: false (922.0)
|   |   |   |   |   |   |   |   |   |   slope_5 > 6
|   |   |   |   |   |   |   |   |   |   |   aspect_360 <= 2: false (254.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   aspect_360 > 2
|   |   |   |   |   |   |   |   |   |   |   |   slope_5 <= 17
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 <= 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_225 <= 1: true (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_225 > 1: false (23.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 > 1: true (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   slope_5 > 17: true (13.0)
|   |   |   |   |   |   |   |   |   curve_significantly_negative > 22
```

Figure 24. LaRMounds tree

```
slope_5 <= 26
|   slope_3 <= 135
|   |   curve_positive <= 61
|   |   |   slope_5 <= 16
|   |   |   |   slope_6 <= 223
|   |   |   |   |   curve_positive <= 49
|   |   |   |   |   |   aspect_315 <= 11
|   |   |   |   |   |   |   curve_positive <= 12
|   |   |   |   |   |   |   |   slope_2 <= 102: false (16192.0/1.0)
|   |   |   |   |   |   |   |   slope_2 > 102
|   |   |   |   |   |   |   |   |   aspect_90 <= 182
|   |   |   |   |   |   |   |   |   |   slope_5 <= 8
|   |   |   |   |   |   |   |   |   |   |   aspect_90 <= 1
|   |   |   |   |   |   |   |   |   |   |   |   aspect_45 <= 1: false (1640.0)
|   |   |   |   |   |   |   |   |   |   |   |   aspect_45 > 1
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 <= 6: false (178.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 > 6
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_225 <= 5: false (34.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_225 > 5
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_4 <= 15: false (10.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_4 > 15
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 <= 7: true (12.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 > 7: false (3.0)
|   |   |   |   |   |   |   |   |   |   |   aspect_90 > 1: false (9743.0/2.0)
|   |   |   |   |   |   |   |   |   |   slope_5 > 8
```

Figure 25. Nelson tree

```
slope_5 <= 27
|   slope_3 <= 133
|   |   curve_positive <= 61
|   |   |   curve_significantly_positive <= 102
|   |   |   |   slope_5 <= 16
|   |   |   |   |   slope_6 <= 223
|   |   |   |   |   |   curve_positive <= 49
|   |   |   |   |   |   |   slope_3 <= 113
|   |   |   |   |   |   |   |   curve_significantly_positive <= 100
|   |   |   |   |   |   |   |   |   aspect_45 <= 1
|   |   |   |   |   |   |   |   |   |   slope_5 <= 9: false (76683.0/3.0)
|   |   |   |   |   |   |   |   |   |   slope_5 > 9
|   |   |   |   |   |   |   |   |   |   |   slope_2 <= 137: false (70908.0/39.0)
|   |   |   |   |   |   |   |   |   |   |   slope_2 > 137
|   |   |   |   |   |   |   |   |   |   |   |   slope_3 <= 50: false (1331.0)
|   |   |   |   |   |   |   |   |   |   |   |   slope_3 > 50
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_360 <= 15: false (69.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_360 > 15
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_45 <= 0: true (7.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_45 > 0: false (2.0)
|   |   |   |   |   |   |   |   |   aspect_45 > 1
|   |   |   |   |   |   |   |   |   |   curve_positive <= 12
|   |   |   |   |   |   |   |   |   |   |   aspect_135 <= 3: false (8511.0)
```

Figure 26. Peterson tree

```
slope_5 <= 25
|   curve_positive <= 61
|   |   curve_significantly_positive <= 102
|   |   |   slope_5 <= 16
|   |   |   |   slope_6 <= 223
|   |   |   |   |   curve_positive <= 49
|   |   |   |   |   |   slope_2 <= 121
|   |   |   |   |   |   |   curve_negative <= 60
|   |   |   |   |   |   |   |   aspect_90 <= 180: false (179068.0/45.0)
|   |   |   |   |   |   |   |   aspect_90 > 180
|   |   |   |   |   |   |   |   |   aspect_360 <= 2
|   |   |   |   |   |   |   |   |   |   aspect_135 <= 6: false (49901.0/23.0)
|   |   |   |   |   |   |   |   |   |   aspect_135 > 6
|   |   |   |   |   |   |   |   |   |   |   slope_6 <= 1: false (15373.0/21.0)
|   |   |   |   |   |   |   |   |   |   |   slope_6 > 1
|   |   |   |   |   |   |   |   |   |   |   |   aspect_225 <= 4: false (749.0)
|   |   |   |   |   |   |   |   |   |   |   |   aspect_225 > 4
|   |   |   |   |   |   |   |   |   |   |   |   |   curve_negative <= 39
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_45 <= 5: false (171.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_45 > 5
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_315 <= 2
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_1 <= 0: false (37.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_1 > 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_1 <= 2: true (9.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_1 > 2: false (5.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_315 > 2: false (64.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   curve_negative > 39
```

Figure 27. Shea tree

```
slope_5 <= 26
|   slope_3 <= 133
|   |   curve_positive <= 61
|   |   |   curve_significantly_positive <= 102
|   |   |   |   slope_5 <= 16
|   |   |   |   |   slope_3 <= 120
|   |   |   |   |   |   slope_6 <= 223
|   |   |   |   |   |   |   curve_positive <= 49
|   |   |   |   |   |   |   |   curve_negative <= 60
|   |   |   |   |   |   |   |   |   slope_2 <= 122
|   |   |   |   |   |   |   |   |   |   slope_5 <= 12: false (152737.0/28.0)
|   |   |   |   |   |   |   |   |   |   slope_5 > 12
|   |   |   |   |   |   |   |   |   |   |   curve_negative <= 22: false (25276.0/6.0)
|   |   |   |   |   |   |   |   |   |   |   curve_negative > 22
|   |   |   |   |   |   |   |   |   |   |   |   aspect_360 <= 0
|   |   |   |   |   |   |   |   |   |   |   |   |   slope_1 <= 0: false (7755.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   slope_1 > 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_6 <= 1: false (2967.0/10.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   slope_6 > 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   curve_around_zero <= 2
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   curve_significantly_negative <= 161: true (11.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   curve_significantly_negative > 161
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 <= 10: false (26.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   aspect_180 > 10: true (6.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   curve_around_zero > 2: false (55.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   aspect_360 > 0: false (48711.0/39.0)
|   |   |   |   |   |   |   |   |   slope_2 > 122
```

Figure 28. Sprunk tree

7. CONCLUSION

Although this research didn't get the results we expected, it is still useful for future works. The algorithm for preprocessing can still be studied and refined to get more accurate results. The training data sets can be changed to manually-made ones to build a more accurate and stable model.

For future researches, we have the following suggestions:

1. Add or delete some attributes that are used in this research.
2. Change the values used to divide the attributes for classification.
3. Make the window size changes dynamically to adapt different situations.
4. Use manually made training data sets to build better models.

By implementing the suggestions above, it is possible to improve the results and get an algorithm that has the ability to adapt different situations with minimal changes with high accuracy.

REFERENCES

[CRU08] S. Crutchley, "Light Detection and Ranging in the Witham Valley, Lincolnshire: an Assessment of New Remote Sensing Techniques", *Archaeological Prospection* 13: 251–257, 2006.

[CEL09] O. Celepcikay, and C. Eick, "REG^2: A Regional Regression Framework for Geo-Referenced Datasets", *Proc. 17th ACM SIGSPATIAL International Conference on Advances in GIS(GIS): 326-335, Nov 2009.*

[CAN86] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. Pami-8, No. 6: 679-697, Nov 1986.*

[PAT13] T. Patil, and S. S. Shereka, "Performance Analysis of Naïve Bayes and J48 Classification Algorithm for Data Classification", *International Journal Of Computer Science And Applications, Vol. 6, No. 6: 256-261. Apr 2013.*

[SIN14] S. Singh, and P. Gupta, "Comparative Study ID3, CART and C4.5 Decision Tree Algorithm: A Survey", *International Journal of Advanced Information Science and Technology (IJAIST), Vol. 27: 97-102. Jul 2014.*

[VOS00] G.Vosselman, "Slope Based Filtering of Laser Altimetry Data", *IAPRS, Vol. XXXIII, Amsterdam. 2000.*

[OPI13] R. Opitz, and D. Cowley, Eds, "Interpreting Archaeological Topography: Airborne Laser Scanning, 3D Data and Ground Observation", *London:Oxbow Books. 2013.*

[GRO02] R. Grossman, M. Hornick, and G. Meyer. "Data Mining Standards Initiatives", *ACM 0002-0782/0800. 2002.*