PROCESSING GEOGRAPHIC INFORMATION SYSTEMS DATA FOR DATA MINING

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

By

Sarthak Ahuja

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

Major Department:
Computer Science

May 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

PROCESSING GEOGRAPHIC INFORMATION SYSTEMS DATA FOR DATA MINING

By

Sarthak Ahuja

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:

Dr. Anne Denton
Chair

Dr. Kendall Nygard
Member

Dr. David Franzen
Member

Approved:

06-04-2013
Date

Dr. Brian M. Slator
Department Chair

ABSTRACT

Spatial data, including image data, are typically downloaded into GIS systems for processing purposes. The GIS data are optimized for establishing spatial relationships among objects. Spatial data can be produced rapidly from a variety of sources and the use of spatial data to improve agricultural management has become common [1]. However, most GIS systems are limited in their data mining capabilities. Data mining software provides advanced prediction capabilities for record-based data. The research goal of this project is to create a tool that would allow input of images and metadata, then process them using geospatial software to convert it to a record format such that data mining can be performed. This process opens the possibility of applying data mining techniques to agricultural data, for which such techniques are not yet in common usage. This paper proposes one such tool for classification of spatial data sets using J48 and Random Forest techniques.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my committee chair, Dr. Anne Denton, for choosing me for this research. Her constant guidance was indispensable for the completion of this project, and my enthusiasm toward the topic is attributed to her.

I would like to thank my committee members, Dr. Kendall Nygard and Dr. David Franzen for expressing interest and taking the time to evaluate this research. I am honored to have them on my committee.

This material is based upon work supported by the National Science Foundation Partnerships for Innovation program under Grant No. 1114363.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GCP	Ground Control Points
GIS	Geographic Information systems
GUI	Graphical User Interface
LSF	Least Square Fitting
MSS	Multi Spectral Scanner
NN	Neural Networks
PPR	Projection Pursuit Regression
RMSE	Root Mean Squared Error
SEP	Standard Error of Prediction
SMLR	Stepwise Multiple Linear Regression
TM	Thematic Mapper

1. INTRODUCTION

Precision agriculture is a term that characterizes the use of detailed spatial information in the direct management of modern agriculture [2]. A branch of precision agriculture involves studying images of fields obtained from remote sensing. Remotely sensed image collection results in large data sets that can be analyzed [3]. If this data is grouped and classified according to patterns with a reasonable accuracy it could help optimize field practices. These practices could include directing fertilizer application rates, reducing farmer input costs, and losses of excessive nitrogen due to leaching or nitrous oxide emissions of.

Agriculture has a major role to play in the economy. “The value of global agricultural products (excluding animal products) grew by 5.9% since 2010 to a value of $1.7 trillion. In 2015, the global agricultural products are forecast to have a value of $2.3 trillion; an increase of 34.1% since 2010 [4].” Information technology has established itself as a significant part of all industries, and the agricultural industry is no different. Together with satellite imagery and geospatial tools, it has proven to be a boon to farmers and many consider it indispensable in modern agriculture.

The input data used in this project consist of images obtained from Landsat satellites, called Landsat images. Yield information is available independently through growers in the same fields as the images were obtained. These images have the geometric quality to ensure that geo-location (identification of the real-world geographic location of an object) is sufficiently accurate. The images used here are from satellite, Landsat 5 that was launched on March 1, 1984. Landsat 6 failed to reach orbit, and Landsat 7 images have a faulty scan line corrector. The images used in this research have been produced using the Thematic Mapper (TM) sensor, which is an upgrade on the previously used Multispectral Scanner (MSS) sensor. The TM sensor records seven bands, each of which has a distinct agricultural significance such as detecting chlorophyll absorption, differentiating soil-vegetation, and other useful features [5]. Figure 1 shows a Band 1 Thematic Mapper image in gray scale format.

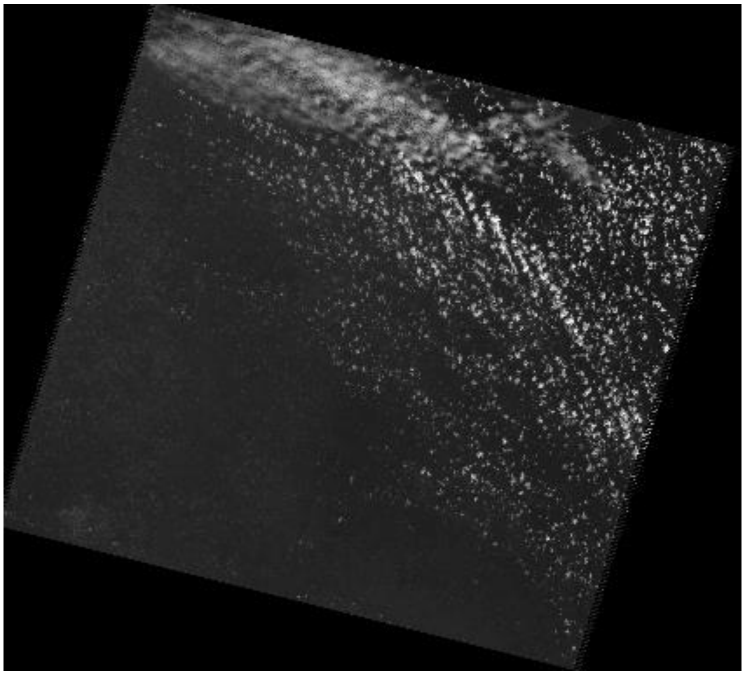


Figure 1. A Landsat 5 gray scale image.

This project shows how data mining can be interfaced with geo-spatial data, and demonstrates the construction of a mathematical model from Landsat images and selected known yield data from the same fields. Yield prediction, which is discussed later in the paper, can be helpful for farming. The prediction models used in this paper are available as part of data mining frameworks. “Data Mining is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems [6].” Data mining includes clustering, in which clusters (groups) of data are formed that are more similar to each other than to those in other clusters. Data mining also includes classification, which generalizes the structure of attributes based on a known class label.

The resulting classification model can be applied to new data collected in a similar format. This paper will demonstrate two classification techniques for yield prediction. Yield will be used as the class label, or in other words, the attribute that is being predicted.

1.1. Motivation

The goal of precision farming is partly to provide tools and methods to manage farming practices spatially according to crop needs, and in doing so, protect the environment from contaminants resulting from over-application of fertilizer and chemical inputs, while increasing farmer efficiency for their economic improvement. It also helps farmers to maintain farming records, make better informed decisions, be able to back track and to improve quality of their products [2]. These are the reasons that inspire studies on yield prediction.

1.2. Problem Statement

What is desired in this project is an ability to apply data mining algorithms on data as incorporated into a Geographic Information system (GIS). The long-term goal of this project would be to utilize a data mining framework alongside GIS framework, or to have the ability to use data mining framework alongside a GIS framework. An alternative to this might be to use a subset of GIS functionality that simulates GIS behavior and merge it with the data mining framework to produce the desired result. It is also needed that some measures be output from the application of data mining tools to check whether a model is suitable for prediction. To understand how agricultural data varies when image attributes vary, plots of relationships between agricultural spatial data and attribute configurations are required.

1.3. Research Strategy

Each band within a Landsat 5 image data set has certain agricultural significance; it is conceivable that certain combinations of the seven band values at a given point can predict the yield at that point. This paper proposes to create a mathematical model with the seven band values and yield value as attributes. The idea then is straight forward; to obtain band values at points on the image whose yields are known, map band values to yield values, and build a model using data mining techniques. However, there is a challenge in the first step itself. The band values of points on an image can be obtained at the pixel level whereas the yield data is obtained based on latitudes and longitudes. The first goal then is to approximate the latitudes and longitudes to the closest pixel, which is 30 meters by 30 meters in size. The geospatial coordinates (longitudes/latitudes) have to be converted to pixel/line coordinates with respect to the image itself. This is aided by the geographical metadata that accompanies a Landsat image. After that the data mining step will follow. The system therefore interfaces the data mining infrastructure with GIS (Geographic Information systems) infrastructure. Figure 2 shows a block diagram that gives an overall view of the system. Each curved box represents data and each square box represents a function.

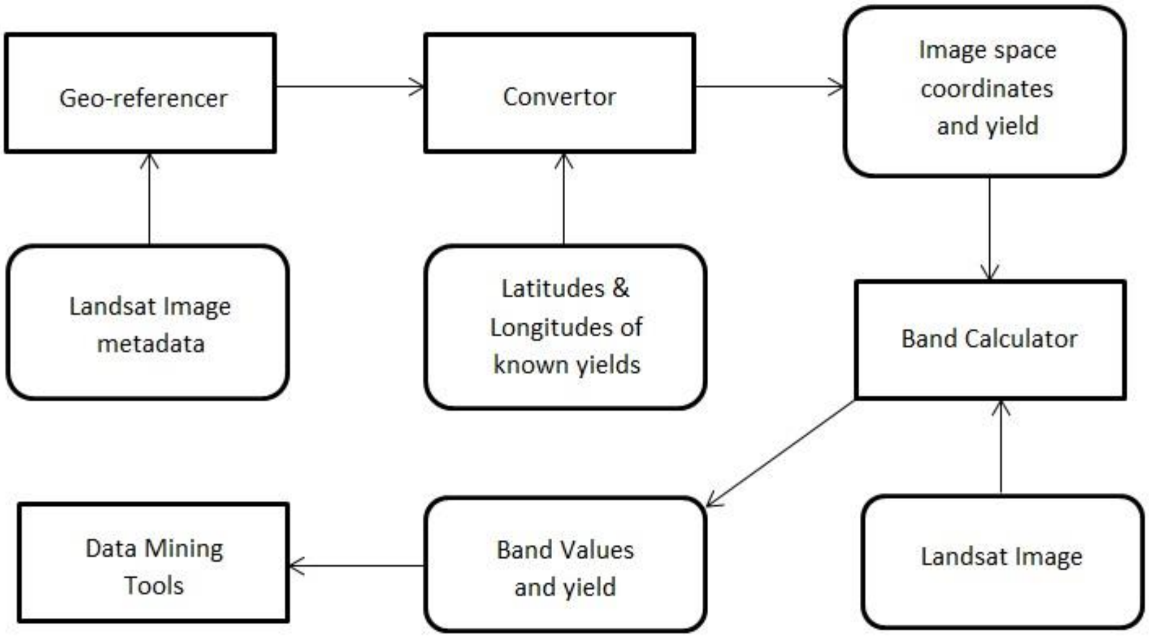


Figure 2. Block diagram showing the functionality and data input of the system.

1.4. Outline of the paper

This paper consists of six chapters. The first chapter gives the introduction to the problem, the motivation and goals that are targeted. It also briefly discusses data mining which is a key aspect in this research. The second chapter discusses the high-level approach that is used for achieving the target along with the various steps involved. The third chapter shows the class design, sequence diagram, flowcharts, Graphical User Interface (GUI), and the resulting performance of the system. The results obtained from the system are discussed in chapter 4. Chapter 5 presents about other related works that address similar problems and chapter 6 gives the final conclusions inferred from the research. The program output for a test data set is provided in the appendix.

2. APPROACH

Based on the discussion under research strategy, the project has been divided into four steps which are georeferencing the Landsat image (any band), using its metadata, converting latitudes and longitudes of points with known agricultural yield to corresponding pixel/line coordinates using the geo-referenced information, exporting yield values along with band values of the pixel/line coordinates obtained for each band from their respective images in a record based file, and subjecting the exported file to data mining algorithms available from Weka (software developed by the University of Waikato), a collection of machine learning algorithms which can either be applied directly to a data set or called from Java code to obtain results.

The required inputs to the system are:

1. Seven images of the same location, each representing seven different bands that can be downloaded from http://landsat.usgs.gov. From the download options, the level 1 product has to be chosen. These images are grayscale or black-and-white images.
2. Metadata that has the georeferencing information in it. This data has the latitudes and longitudes of the four corners of the image, the length and the width of the image and is available in the "'scene ID'_MTL.txt" file that is included in the level 1 product that was downloaded. The scene ID is specific to the downloaded file. Example, LT50290272010217EDC00.
3. Known data containing coordinates of points and their corresponding yields. This comes from a record-based file obtained from American Crystal Sugar Company. The unit of yield is tons per acre. This file is in .dbf format which can be read by GIS software.

It is important that the images downloaded were captured about the same time as the date the yield was measured. It is also required that the images include the coordinates given in the known data file. The coordinates and the time of capture can be specified on the download web site.

2.1. Georeferencing

Georeferencing is locating a position in physical space [7]. This project requires us to map latitude/longitude to pixel, line, or X, Y coordinates with the origin being the top left corner of the image, so that the X, Y values of points with known yields can be computed. These are needed in order to calculate the band values, the points' coordinates relative to the image or image space coordinates are needed.

"Ground Control Points (GCPs) are points on the earth's surface with a known location which can be used to geo reference image data sources for remotely sensed images [8]." These are points of which geo-spatial coordinates and image space coordinates are known. A set of GCPs can be used to create a transformation matrix of a polynomial nature. This matrix can be used for converting coordinates from one system to another. The more the GCPs are available, the higher the precision of the transformation matrix. The scene_id_MTL.txt file that comes as a part of the downloaded level 1 product includes the latitudes and longitudes of four corners of the image and also includes its length and width. The four corners have been used as GCPs with their X, Y coordinates calculated from the length and width are:

Upper Left corner (origin): (0, 0)

Lower Left corner: (0, length)

Upper Right corner: (width, 0)

Lower Right corner: (width, length)

This coordinate system is understood by Java which is the platform used for developing this project.

2.2. Conversion

From the transformation matrix created, coordinates can be transformed from one coordinate system to another using polynomial, spline, or adjust transformation. Polynomial transformation has been used in this project. It uses a polynomial that is generated from GCPs and the least squares fitting (LSF) algorithm. This algorithm is used where the number of equations is more than the number of unknowns [9]. The LSF algorithm aims at creating a generalized formula that can be applied to any point while adjusting the position of some GCPs. The minimum number of GCPs required is three for first order polynomial and six for the second. Since there are four GCPs available from metadata, a first order polynomial will be the obvious choice. The following equations can be used for converting coordinates using first order or affine polynomials.

$$x' = Ax + By + C$$

$$y' = Dx + Ey + F$$

Where

x is the column count in image space

y is the row count in image space

x' is the horizontal value in coordinate space (longitude)

y' is the vertical value in coordinate space (latitude)

A is the width of cell in map units

B and D are rotation terms

C is the x' value of the center of upper-right cell.

E is negative of height of cell in map units

F is y’ value of the center of upper right cell [10].

Image space coordinates are calculated from latitudes and longitudes of points with known yield values.

2.3. Export

Having the image space coordinates available, the seven band values are computed for them from seven different images each one representing one of the seven bands. These images are also included in the level 1 product and have the naming convention as “‘scene_id’_B‘x’.TIF” where x is the band number ranging from 1 to 7. The band value of a point is the red, green or blue value at its location (X, Y) because the three are the same for a grayscale image at a given point. These band values range from 0 to 255. The yield values obtained from the known data file are real numbers with units of tons per acre. Since, data mining classification tools available from Weka cannot operate upon real numbered classes, the yield values are assigned yield categories based upon the following scheme in Table 1. This scheme has been framed considering that the usual yield values lie between 20-30 tons per acre and few others between 30-53 tons per acre. Since, data mining has to be done on band values and yield categories, coordinates are omitted and the seven band values and yield categories are exported to a record based file.

Table 1

Key of Classes or Yield Categories and the Range of Yield Values Covered by Them.

YIELD VALUE	YIELD CATEGORY
<20	A
20-22	B
23-25	C
26-28	D
29-31	E
32-34	F
35-37	G
38-40	H
41-43	I
44-46	J
47-49	K
50-52	L
>=53	M

2.4. Data-Mining

At this point, the data is ready to be mined upon with band 1, band 2, band 3, band 4, band 5, band 6, band 7 values and yield (which is the class) attributes. The first step for classification is to choose the test data set. This project discusses calculating the classification error on the training data set and cross-validation. Leave-one-out cross-validation is a technique for evaluating classification accuracy by dividing the training data set into k partitions, using one partition as test data and the other k-1 partitions as training data. This process is repeated k times such that every partition is used as test data once with others as training data. The final result is the average of the k results. For classification, there are many techniques available in Weka. This project makes use of two classification techniques, J48 Tree and Random Forest.

2.4.1. J48 Tree

The J48 tree is the java implementation of C4.5 algorithm [11]. This method builds a decision tree based on the available data and is used to predict a target value (dependent variable) based on different attributes (independent variables). Information gains are computed for all attributes at every split value (or threshold value) and the combination with the highest informational gain is chosen [12]. It then partitions the data set into subsets according to the value of the chosen attribute. The steps are repeated for each of the smaller data sets. The processing stops for a data set when all instances from that data set belong to a common class in which case, a tree leaf labeled with that class is returned or when there are no more instances remaining, or there are no more attributes left (leaf labeled with the most frequent class or the disjunction of all the classes is returned). The information gain for an attribute can be calculated using the following formulas:

$$\text{Entropy, } E(S) = \sum_{i=1 \text{ to } n} -\Pr(C_i) * \log_2 \Pr(C_i)$$

$$\text{Gain, } G(S, A) = E(S) - \sum_{j=1 \text{ to } m} \Pr(A_j)E(S_{Aj})$$

Here, S is the total data set, E(S) is the information entropy of S, A is the attribute whose information gain is being calculated, G(S, A) is the gain of S after a split on attribute A, n is the number of classes in S (in this paper n will be 13), $\Pr(C_i)$ is the probability of class C_i in S, m is the number of different values of attribute A in S, $\Pr(A_j)$ is the probability of instances that have A_j value in S and $E(S_{Aj})$ is the information entropy of subset of S that have A_j value [13].

The advantages of using a J48 tree are:

1. For problems where attribute importance is key, it is more accurate than other classifiers like the Naïve Bayes classifier, Neural Network, and Support Vector Machine Classifier.

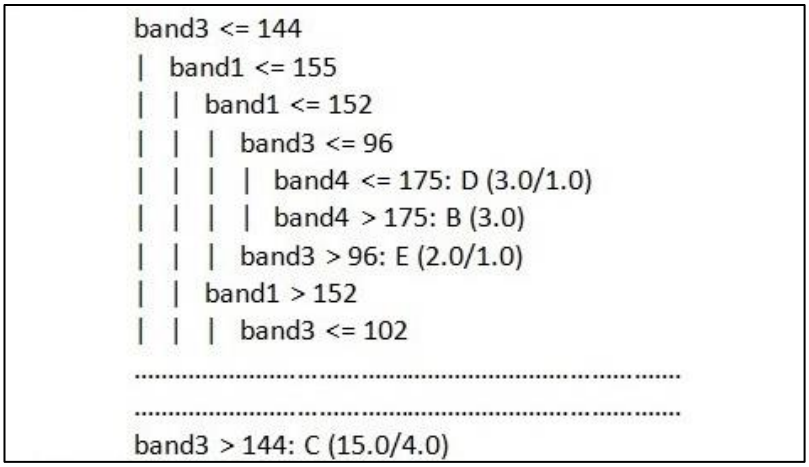
2. The tree can be built quite quickly for a small data set.
3. Builds models that can be easily interpreted
4. It is easy to implement.

The disadvantages are:

1. Small variation in data can lead to different decision trees especially when the variables are close to each other in value.
2. Does not work very well on a small training set.

It is suitable for real world problems as it deals with numeric attributes and missing values. The algorithm can be used for building smaller or larger, more accurate decision trees, and the algorithm is quite time efficient.

The J48 tree (a portion) built using Weka looks like the tree shown in Figure 3. Here, the attribute with the highest information gain for the overall tree is band3 with a threshold value of 144. The nodes with an alphabet represent a leaf node with the alphabet being the class of the leaf. The class names are followed by brackets containing the total number of instances assigned to that node and how many of those instances were incorrectly classified. If an attribute vector has a band3 value greater than 144, it will be assigned the class C otherwise, it will traverse through the other branch and will check if its band1 value is less than or equal to 155. This will continue until a leaf node is reached and its corresponding class will be assigned to the attribute vector.



```
band3 <= 144
|  band1 <= 155
|  |  band1 <= 152
|  |  |  band3 <= 96
|  |  |  |  band4  <= 175: D (3.0/1.0)
|  |  |  |  band4  > 175: B (3.0)
|  |  |  band3 > 96: E (2.0/1.0)
|  |  band1 > 152
|  |  |  band3 <= 102
.........................................................................
.........................................................................
band3 > 144: C (15.0/4.0)
```

Figure 3. A portion of a J48 tree output from Weka tools.

2.4.2. Random Forest

Random Forest machine learner consists of many individual learners or decision trees who vote their classifications to conclude an overall classification. These trees grow in a well-defined fashion.

For every tree, a portion of the training data (usually one third) is sampled with replacement and the modified data set, called in-bag data, is used for training, and the original data portion, called out-of-bag data, is used to get a true evaluation of classification error with the addition of trees. Also, at each step a small number of attributes are chosen at random (for each node) which determine decisions on their respective nodes. The best attribute and split value is picked by finding the combination (attribute and its value) with the highest information gain. Hence, it also evaluates the importance of all variables or attributes in the data and optimizes (rearranges attributes) accordingly [14]. The forest is thus optimized before the testing data set is supplied to it.

In Weka, the number of trees and the number of attributes (or features) at a node can be set by the programmer. In this research default values of 10 trees and 4 features has been used. Each tree has a different selection of features (chosen randomly) and a different training data set (due to different sampling). Upon creating a model the Random Forest algorithm also computes the overall out-of-bag error which is the proportion of incorrectly classified instances from the out-of-bag data.

The advantages of using Random Forest are:

1. It runs efficiently on large data sets.
2. It can evaluate the importance of each attribute and produce results accordingly.
3. It evaluates itself as it is built.
4. Research shows that, Random Forest Classification is one of the most accurate methods available today [15].

3. DESIGN AND EXECUTION

3.1. Design

The classes have been designed such that each one covers a certain independent functionality hence, ensuring high cohesion. LandsatMining is the class with the main method or the entry point of the application. The Transform class is an abstract class as it would never have an object of its own and its constructor parses GCP data from the meta data file and points with known yield values from yield report file. Class AffineTransform extends from Transform and it contains the method to convert longitudes and latitudes to image space coordinates X and Y. If this project is required to include other transformation methods like Spline Transform, etc. in the future, they may be implemented in their respective classes that extend from Transform since any method would need the same parsing steps that are defined in the Transform() constructor. They would also need to implement the method LatLontoXY() as it is declared as an abstract method in Transform. Hence, the system is extensible and easy to maintain. Classes KnownYieldXY and KnownYieldLatLong are included to create objects that hold yield values along with the coordinates they pertain to, in pixel/line and latitude/longitude coordinate systems respectively. These classes also aid in interfacing functions in different classes, so that changing a function doesn't affect the other. This makes the system loosely coupled. The ExportBands class is responsible for getting seven band values for the converted coordinates and exporting them (along with their yield categories) to a record-based file. Class Classifiers' constructor implements reading the exported file and preparing the data based upon the type of testing data set chosen by the user while its derived classes, ClassifierJ48 and ClassifierRandForest have functions to perform their respective classification techniques on them. Figure 4 shows the different classes, their data members, methods, associations and inheritance.

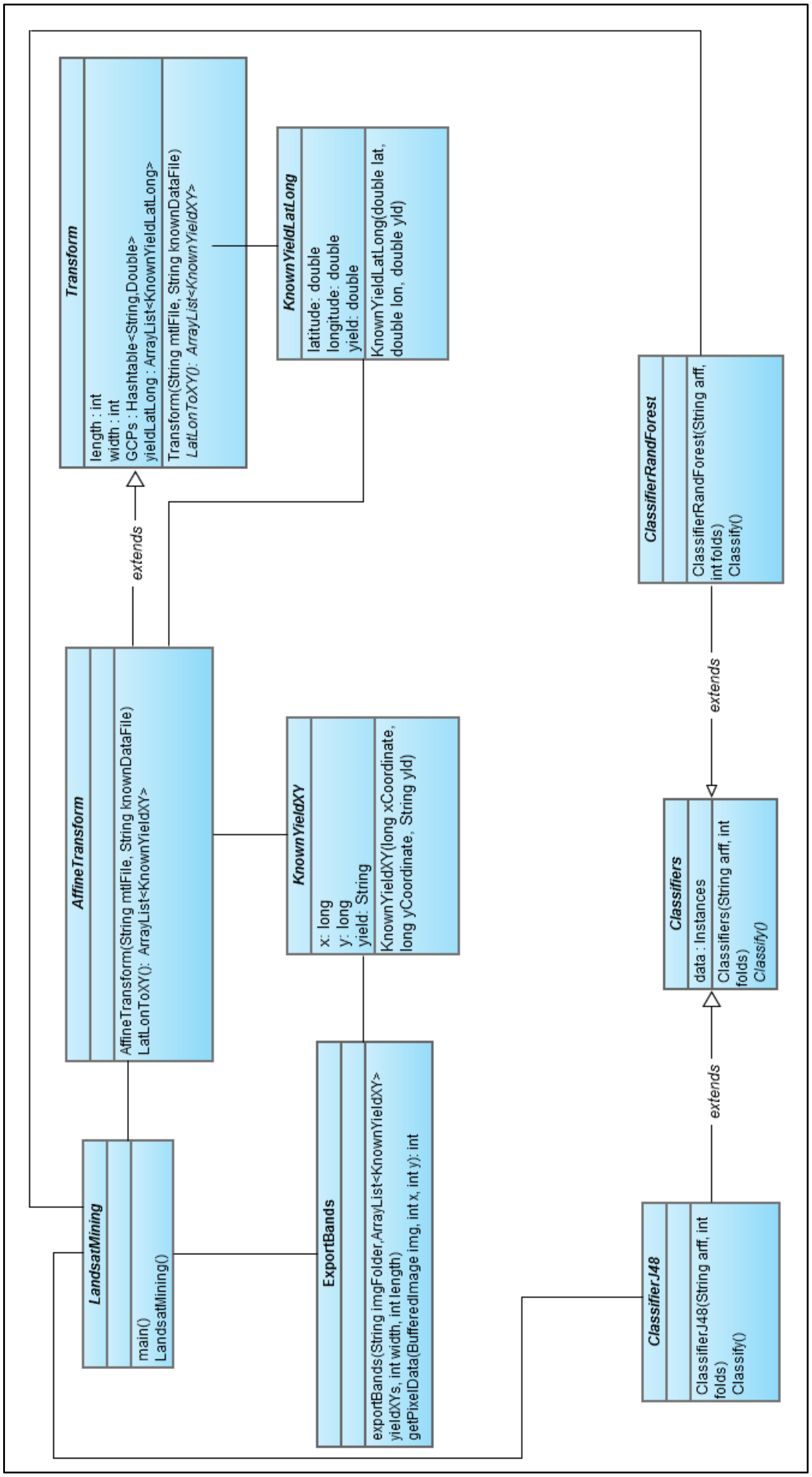

Transform
length : int
width : int
GCPs : Hashtable<String,Double>
yieldLatLong : ArrayList<KnownYieldLatLong>
Transform(String mtlFile, String knownDataFile)
LatLonToXY(): ArrayList<KnownYieldXY>
KnownYieldLatLong
latitude: double
longitude: double
yield: double
KnownYieldLatLong(double lat, double lon, double yld)
extends
AffineTransform
AffineTransform(String mtlFile, String knownDataFile)
LatLonToXY(): ArrayList<KnownYieldXY>
KnownYieldXY
x: long
y: long
yield: String
KnownYieldXY(long xCoordinate, long yCoordinate, String yld)
LandsatMining
main()
LandsatMining()
ExportBands
exportBands(String imgFolder,ArrayList<KnownYieldXY> yieldXYs, int width, int length)
getPixelData(BufferedImage img, int x, int y): int
ClassifierRandForest
ClassifierRandForest(String arff, int folds)
Classify()
extends
Classifiers
data : Instances
Classifiers(String arff, int folds)
Classify()
extends
ClassifierJ48
ClassifierJ48(String arff, int folds)
Classify()

Figure 4. UML class diagram showing the classes and how they relate to each other.

3.2. Sequence of Events

Figure 5 shows a sequence diagram of the system. As stated earlier, the entry point of the application is in the LandsatMining class, the system execution starts with the LandsatMining method creating an object of AffineTransform. While doing so, paths to the meta data file and the known yield data file are supplied as required by its constructor. Then a call is made to the function LatLontoXY function which georeferences using data parsed in the constructor and converts latitudes and longitudes of points of known yield to pixel and line coordinates. These are collected in an array list together with yield categories and are returned to LandsatMining method.

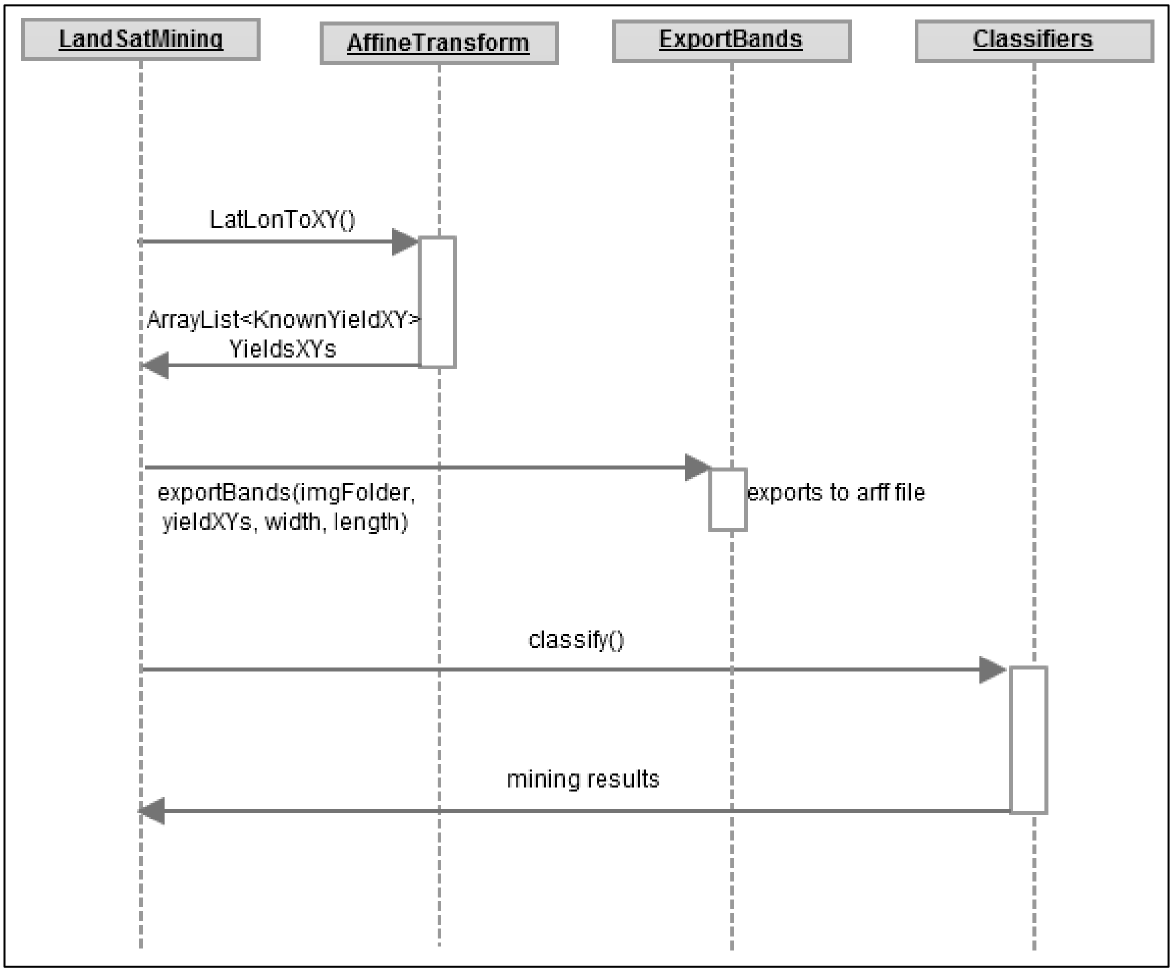


Figure 5. UML sequence diagram showing function calls and the flow of data.

It then calls exportBands method in ExportBands class with the array list as an argument to calculate band values and export these values along with yield categories to an arff (format understandable to Weka) file. Depending on the algorithm chosen by the user, a call is made to the appropriate class to perform data mining upon the exported data and the results are sent back.

3.3. Execution

The project is implemented in Java using Eclipse IDE, which makes use of some external libraries. It uses GDAL for georeferencing and converting geospatial coordinates to image space coordinates. For reading TIFF images (images with .TIF extension), which is the format the grayscale landsat TM images are available in, it requires jai_codec-1.1.3-alpha, imageio-ext-utilities-1.1.7, jai_imageio-1.1-alpha and jai_core-1.1.3 libraries. The default imageio library that comes as part of java installation does not support TIFF images. Finally, it uses the Weka library for performing data mining. All these libraries can be freely downloaded.

As stated earlier, the file with coordinates of known yield values is input to the system. This file as obtained from American Crystal Sugar Company is in .dbf format which cannot be read by this system because of encoding incompatibility. Hence, it is important that the file format be changed to .txt before it is input to the system. This can be done by opening the .dbf file in Microsoft Excel and saving it as tab delimited .txt file. In the export step, the record file that is generated has an .arff format, which is the format that can be read by the Weka tools. This file is generated in the location specified by the user. The same file location is supplied to the classification algorithms.

The different programs in the project have been developed with storage and speed efficiency in mind. Flowcharts to the individual project files are shown in Figures 6, 7, 8, 9 and 10.

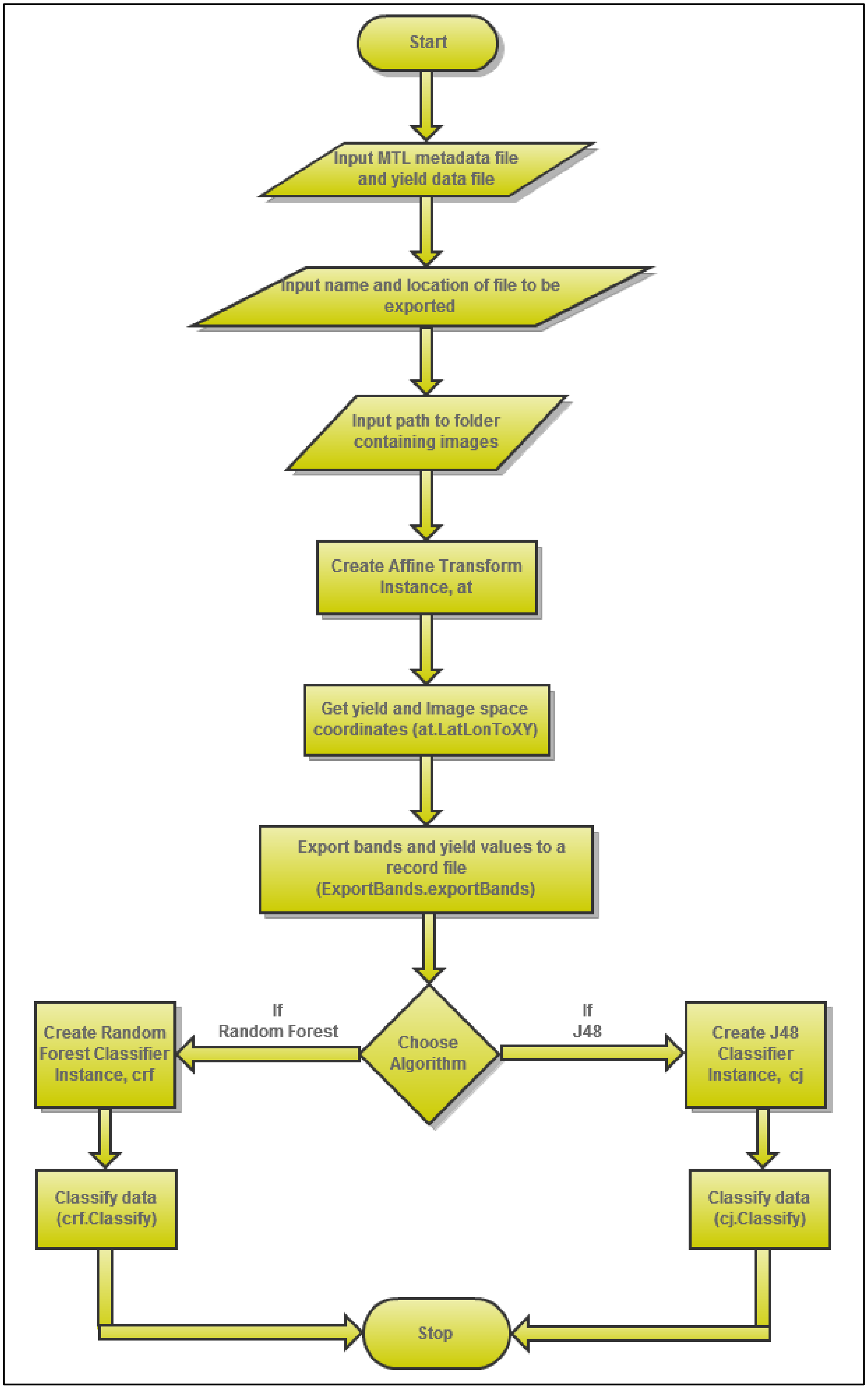

Start
Input MTL metadata file and yield data file
Input name and location of file to be exported
Input path to folder containing images
Create Affine Transform Instance, at
Get yield and Image space coordinates (at.LatLonToXY)
Export bands and yield values to a record file (ExportBands.exportBands)
Choose Algorithm
If Random Forest
If J48
Create Random Forest Classifier Instance, crf
Create J48 Classifier Instance, cj
Classify data (crf.Classify)
Classify data (cj.Classify)
Stop

Figure 6. LandsatMining.LandsatMining() flowchart.

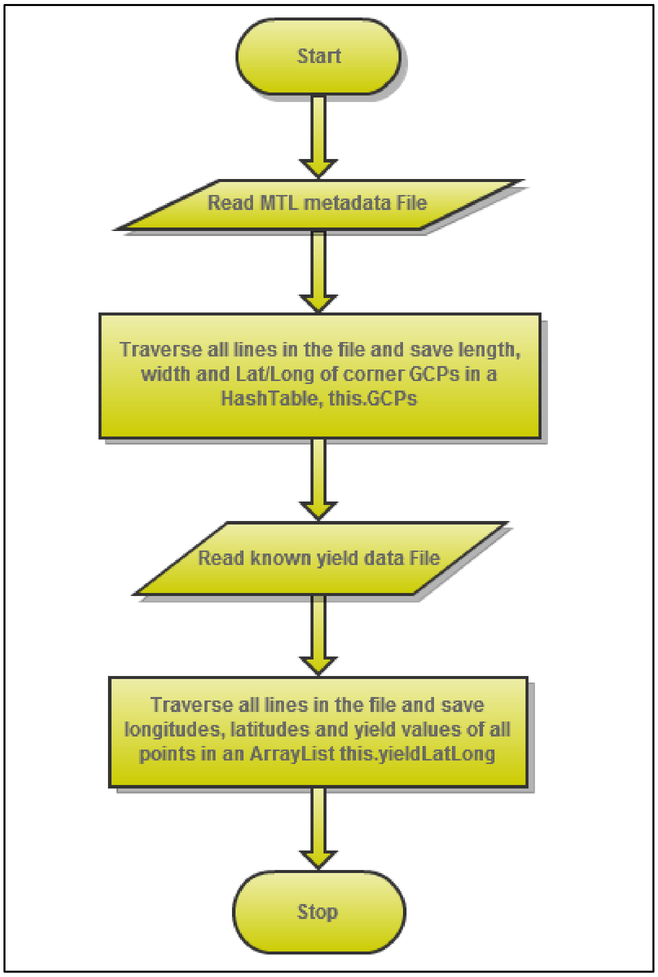

Start
Read MTL metadata File
Traverse all lines in the file and save length, width and Lat/Long of corner GCPs in a HashTable, this.GCPs
Read known yield data File
Traverse all lines in the file and save longitudes, latitudes and yield values of all points in an ArrayList this.yieldLatLong
Stop

Figure 7. Transform.Transform() flowchart.

In the metadata file, the four corner GCPs are situated in lines that abide by the format, ‘PRODUCT_’xx‘_CORNER_’LAT/LON, where xx denotes one of the possible corners, UL, UR, LL and LR, which stand for Upper Left, Upper Right, Lower Left and Lower Right, respectively. In some files, the format is ‘CORNER_’xx‘_’LAT/LON‘_PRODUCT.’ The length of the image is contained in the line that starts with either “REFLECTIVE_LINES” or “PRODUCT_LINES_REF” while the width is contained in the line starting with “REFLECTIVE_SAMPLES” or “PRODUCT_SAMPLES_REF.” All possibilities are taken care of by the parser in Transform constructor. In the yield data file, the yield, latitude and longitude values are situated in the 13th, 23rd and 22nd columns, in parallel order.

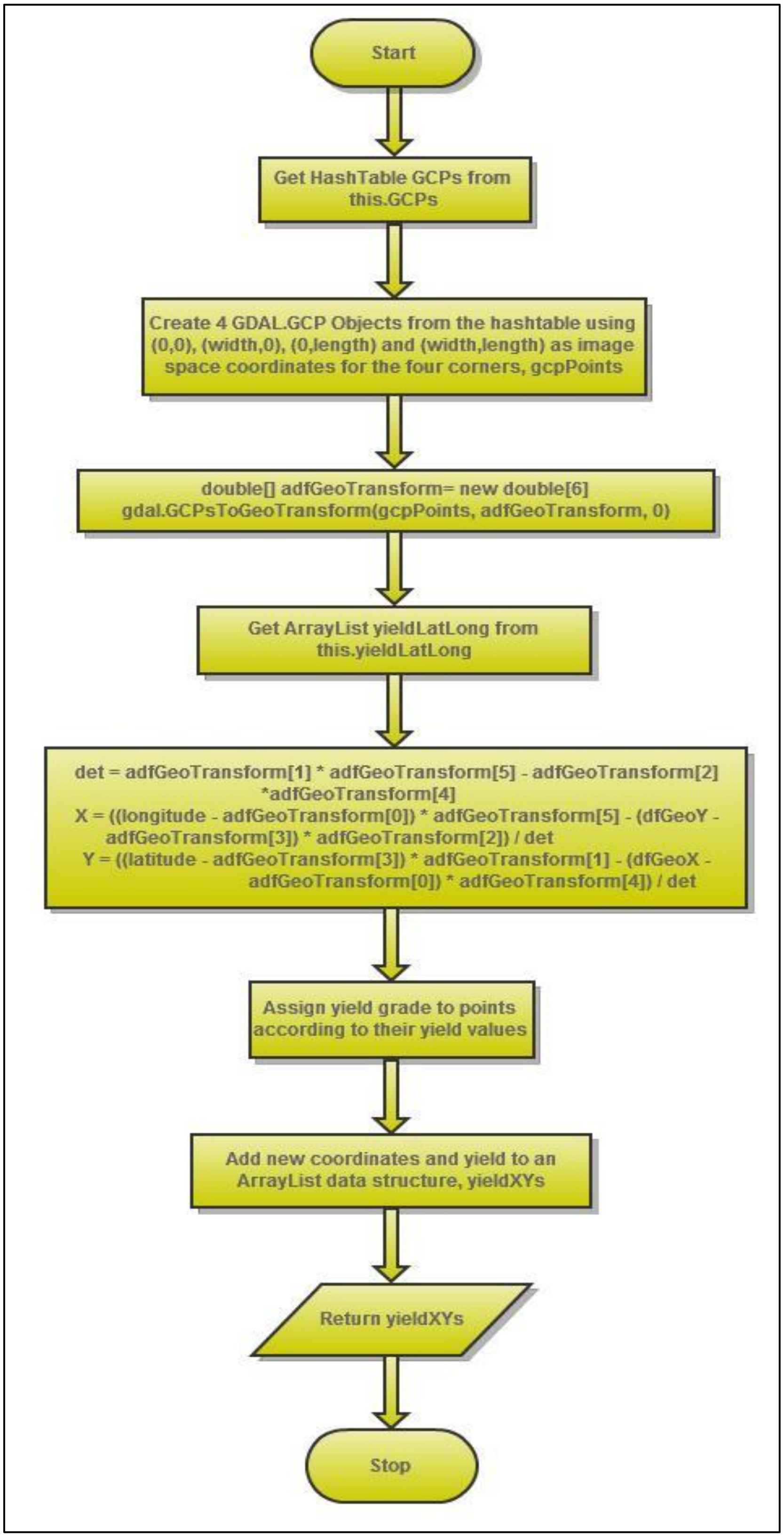

Start
Get HashTable GCPs from this.GCPs
Create 4 GDAL.GCP Objects from the hashtable using (0,0), (width,0), (0,length) and (width,length) as image space coordinates for the four corners, gcpPoints
double[] adfGeoTransform= new double[6]
gdal.GCPsToGeoTransform(gcpPoints, adfGeoTransform, 0)
Get ArrayList yieldLatLong from this.yieldLatLong
det = adfGeoTransform[1] * adfGeoTransform[5] - adfGeoTransform[2] *adfGeoTransform[4]
X = ((longitude - adfGeoTransform[0]) * adfGeoTransform[5] - (dfGeoY - adfGeoTransform[3]) * adfGeoTransform[2]) / det
Y = ((latitude - adfGeoTransform[3]) * adfGeoTransform[1] - (dfGeoX - adfGeoTransform[0]) * adfGeoTransform[4]) / det
Assign yield grade to points according to their yield values
Add new coordinates and yield to an ArrayList data structure, yieldXYs
Return yieldXYs
Stop

Figure 8. AffineTransform.LatLontoXY() flowchart.

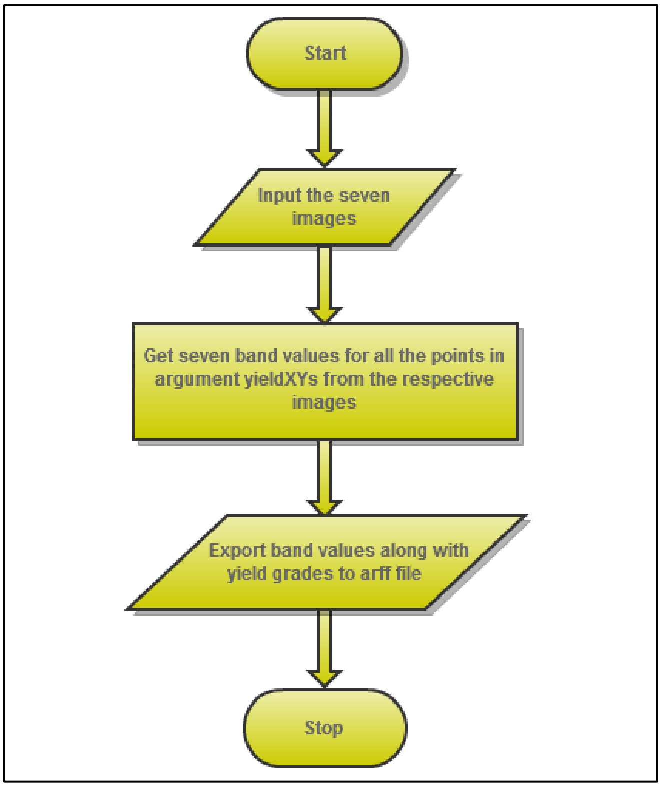

Start
Input the seven images
Get seven band values for all the points in argument yieldXYs from the respective images
Export band values along with yield grades to arff file
Stop

Figure 9. ExportBands.exportBands() flowchart.

The exportBands function described in Figure 9 exports seven band values and yield categories in a file that has an .arff extension. The reason behind choosing this format is that .arff is the only file format that can be read by the data mining tool, Weka. This format has some specifications that have been complied with. The file starts with the keyword “@relation” followed by the name of the relationship. After a line break, it must have all attributes listed in separate lines prefixed by “@attribute” and suffixed by the data type of the attribute. For the first seven attributes, band1, band2, band3, band4, band5, band6 and band7, the data types are real numbers and are specified by the word “real.” The last attribute yield which is also the class label, the data type is an enumeration with alphabetic A to M as possibilities and is described as (A, B, C, D, E, F, G, H, I, J, K, L, M) in the file. After another line break, the file should have “@data” written followed by the actual data with each record separated by a line and each attribute value separated by a comma (,).

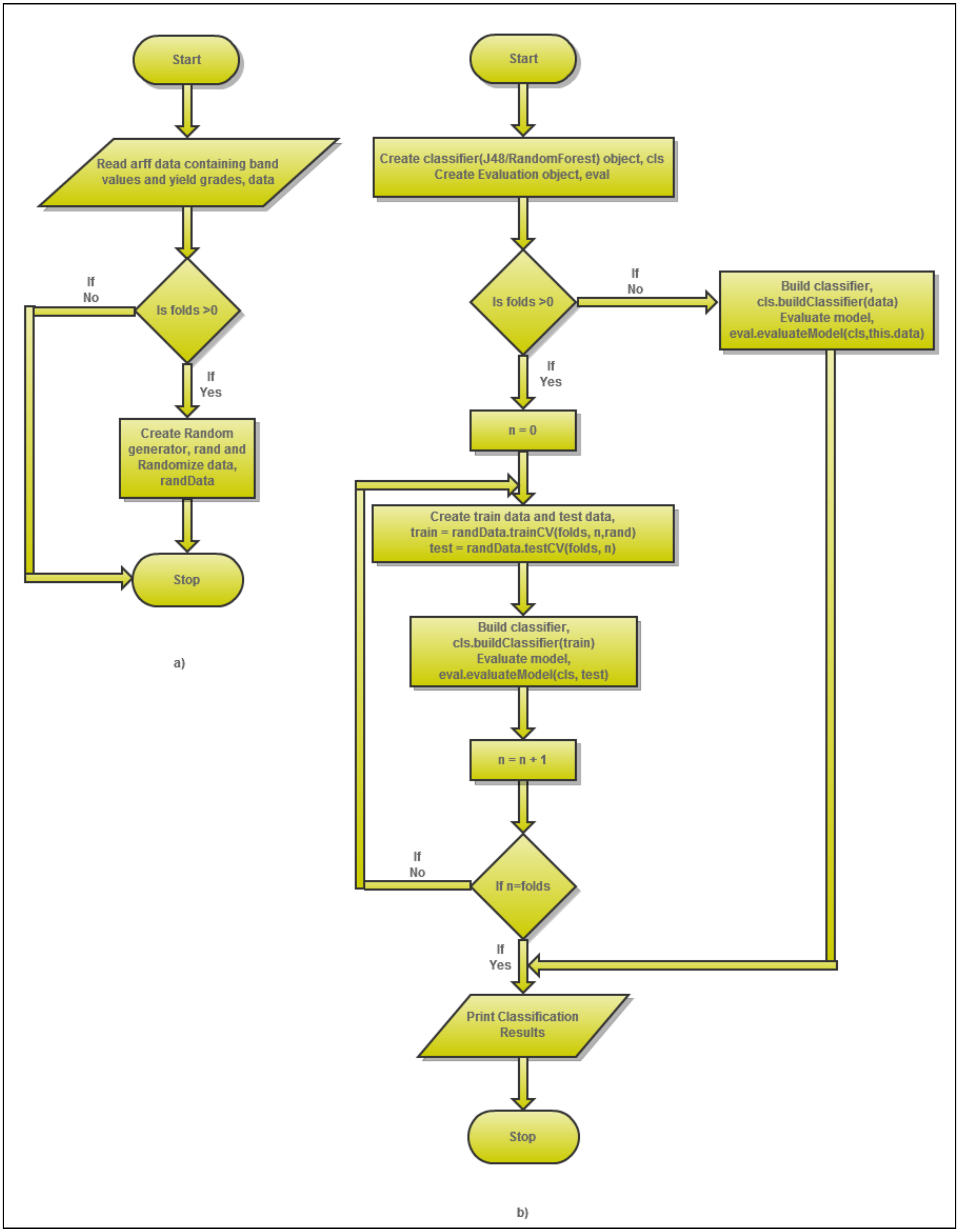
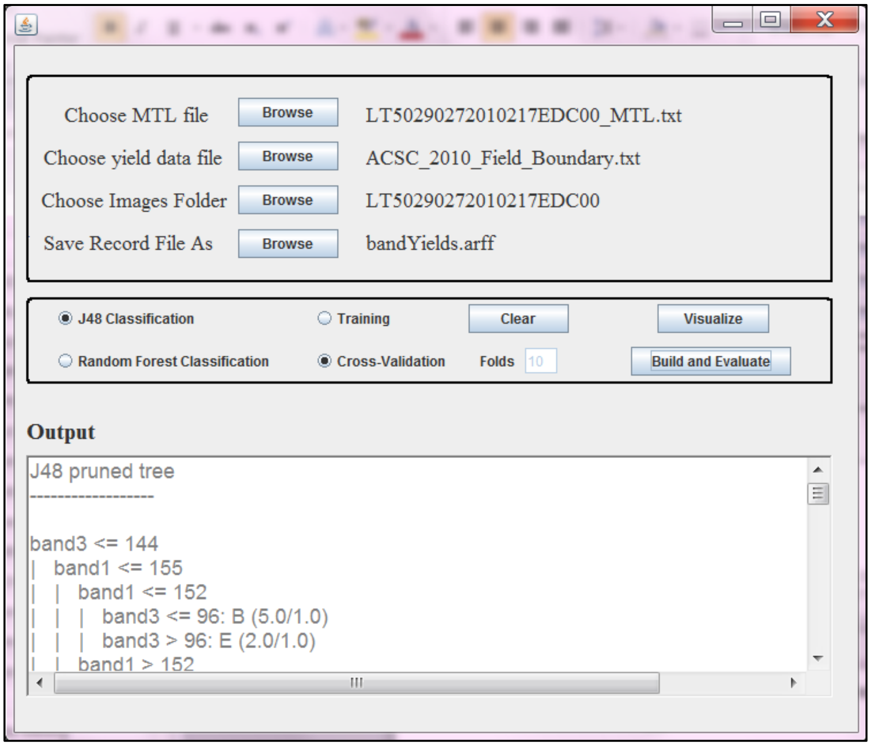

Start
Read arff data containing band values and yield grades, data
Is folds >0
If No
If Yes
Create Random generator, rand and Randomize data, randData
Stop
a)
Start
Create classifier(J48/RandomForest) object, cls
Create Evaluation object, eval
Is folds >0
If No
Build classifier, cls.buildClassifier(data)
Evaluate model, eval.evaluateModel(cls,this.data)
If Yes
n = 0
Create train data and test data,
train = randData.trainCV(folds, n,rand)
test = randData.testCV(folds, n)
Build classifier, cls.buildClassifier(train)
Evaluate model, eval.evaluateModel(cls, test)
n = n + 1
If n=folds
If No
If Yes
Print Classification Results
Stop
b)

Figure 10. a) Classifiers.Classifiers() flowchart; *b)* Classifiers.Classify() flowchart.

After the .arff file is exported, it undergoes classification using the J48 or Random Forest classification techniques. This choice is given in LandsatMining.LandsatMining() function as a user input. Figure 10 a) shows the constructor of the base class Classifiers that reads the data file and prepares it for classification, depending upon whether the user wants to use the training data set as the testing data set, or use cross-validation. If the user chooses the former, the Classifier object is created with 0 as the ‘folds’ argument. The Classifiers class and its derived classes understand that if the value of folds is 0, the training data set is to be used as a testing data set; otherwise it uses cross-validation with the value of folds as the number of folds. Figure 10 b) shows a general flowchart for method Classify either called upon the ClassifierJ48 object, or the ClassifierRandForest object.

3.4. Graphical User Interface

The GUI for this project is built using Java Swing. When the program is run, a window appears (Figure 11 a)) which is divided into three sections. From the first section, the user is able to select locations to the metadata file, the yield data file, and the folder containing the seven band images. The user also chooses a file name for the file that will contain the exported record data and the location where the file will be saved. In the second section, algorithm and test data sections set are made. The plots between different band values and yield values (shown in 4.1) can be seen by clicking on the ‘Visualize’ button. Clicking the ‘Build and Evaluate’ button gets the model (tree) and its evaluation in an output window situated in the third section. The GUI also gives constructive feedback in case of errors. For example, if the number of folds is set to 1 which is not accepted by either algorithm when cross-validating, a message is displayed to the user in red as shown in Figure 11 b). The clear button allows the user to clear all the fields in the interface.



Choose MTL file: LT50290272010217EDC00_MTL.txt
Choose yield data file: ACSC_2010_Field_Boundary.txt
Choose Images Folder: LT50290272010217EDC00
Save Record File As: bandYields.arff

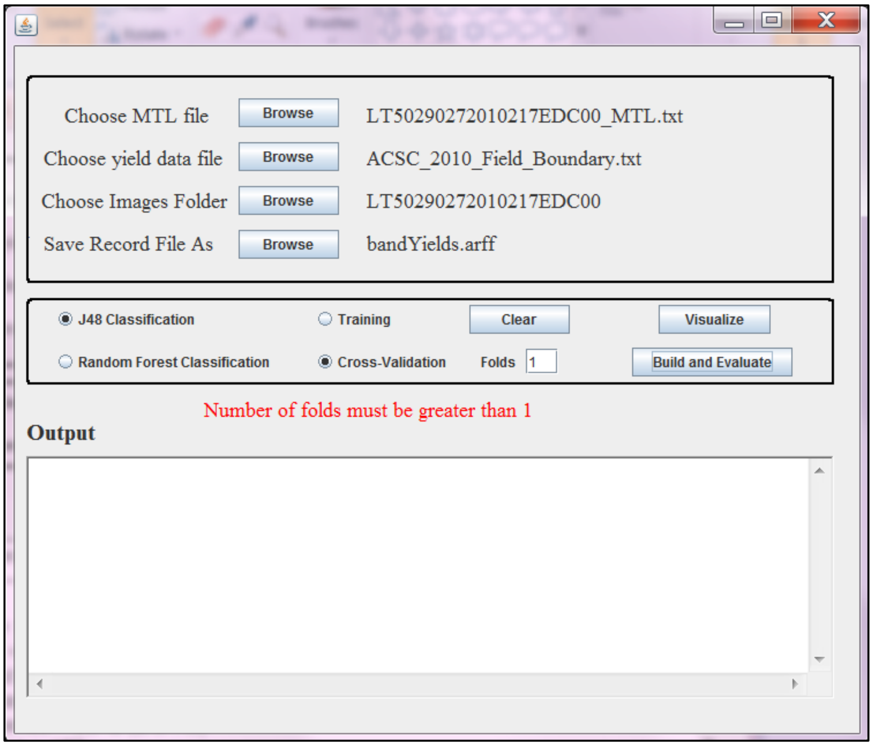
● J48 Classification ○ Training
○ Random Forest Classification ● Cross-Validation Folds 10

Output

```
J48 pruned tree
------------------

band3 <= 144
|   band1 <= 155
|   |   band1 <= 152
|   |   |   band3 <= 96: B (5.0/1.0)
|   |   |   band3 > 96: E (2.0/1.0)
|   |   band1 > 152
```

Figure 11 a). GUI showing J48 algorithm being run with training data set as test data set.



Choose MTL file: LT50290272010217EDC00_MTL.txt
Choose yield data file: ACSC_2010_Field_Boundary.txt
Choose Images Folder: LT50290272010217EDC00
Save Record File As: bandYields.arff

● J48 Classification ○ Training
○ Random Forest Classification ● Cross-Validation Folds 1

Number of folds must be greater than 1

Output

Figure 11 b). GUI showing error message on an attempt to run the program with folds = 1.

3.5. Performance

The run time complexities of data parsing, converting coordinates, and exporting are linearly dependent on the number of records and can be described as O (n). The complexity of the J48 algorithm is O (n * log(n)) for training and O (m * log(m)) for testing where n is the number of records in the training data set, and m is the number of records in the testing data set [16]. When using the training data set as the testing data set, the values of m and n will be the same, bringing the overall complexity to O (n * log(n)). During cross-validation, both training and testing algorithms will run k times on data sets k times smaller than the original data set where k is the number of folds. That means, the complexity will be O (k*(n/k)*log(n/k)). Since, k is a constant, the overall complexity for J48 algorithm will be O (n * log(n)). Similarly, the overall complexity of Random Forest Algorithm is O (t*Sqrt((log t)+1)* n * log n) for both kinds of testing data sets where t is the number of trees and n is the number of records [17].

4. RESULTS

4.1. Visualization

The data that was mined upon in this project (see the appendix for details) had 1326 instances which had seven band values and yield values that were grouped into 13 categories A, B, C, D, E, F, G, H, I, J, K, L, M each (see Table 1 for details). To visualize, or to check for visible patterns in the data, yield categories were plotted against band values for all seven bands. Figures 12 a), 12 b), 12 c), 12 d), 12 e), 12 f), 12 g) show these plots for band 1, band 2, band 3, band 4, band 5, band 6 and band 7, respectively. A class color key is provided on each page to map colors to the class (yield category). Since most of the values lie between 20-30 tons per acre, we are most interested in classes B, C, D and E.

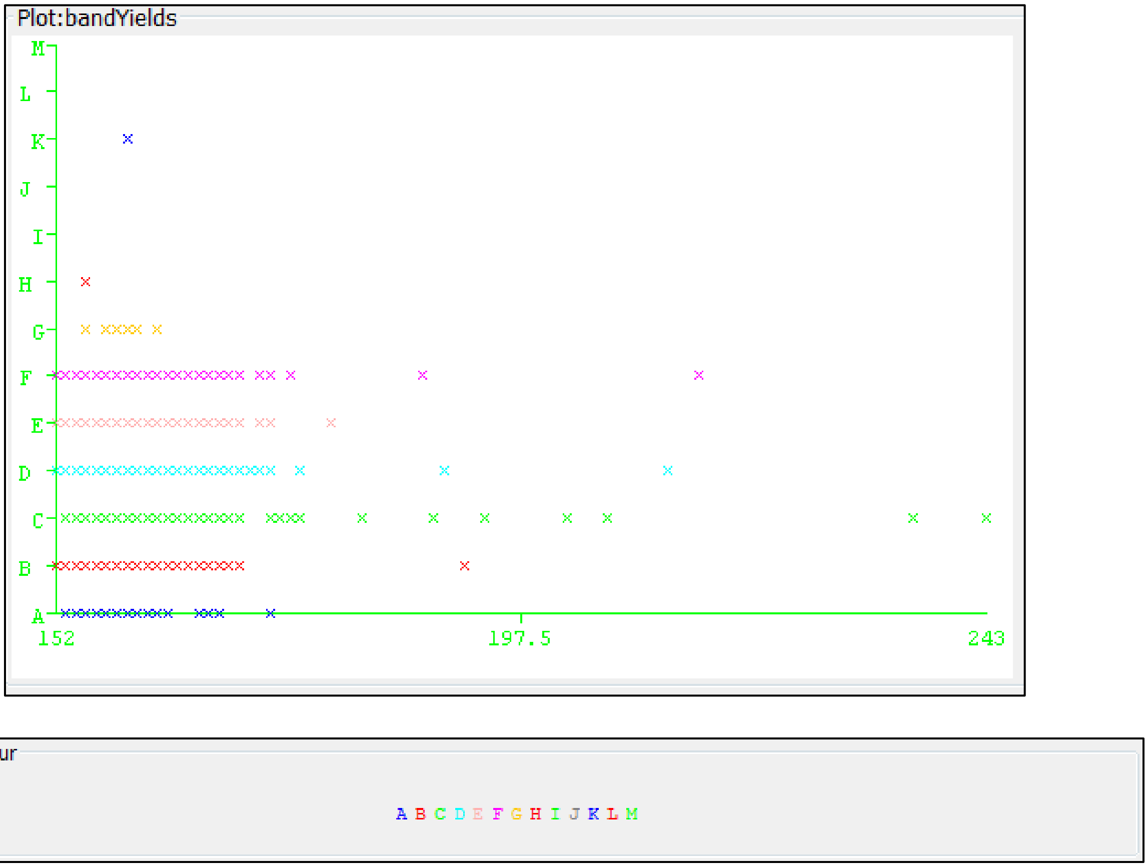


Figure 12 a). Yield categories plotted against band 1 values.

Yield categories plotted against band 1 in Figure 12 a) do not show any particularly prominent pattern except that there are some class C points (23-25 tons per acre) against band values of around 170 while other classes are sparse in that region.

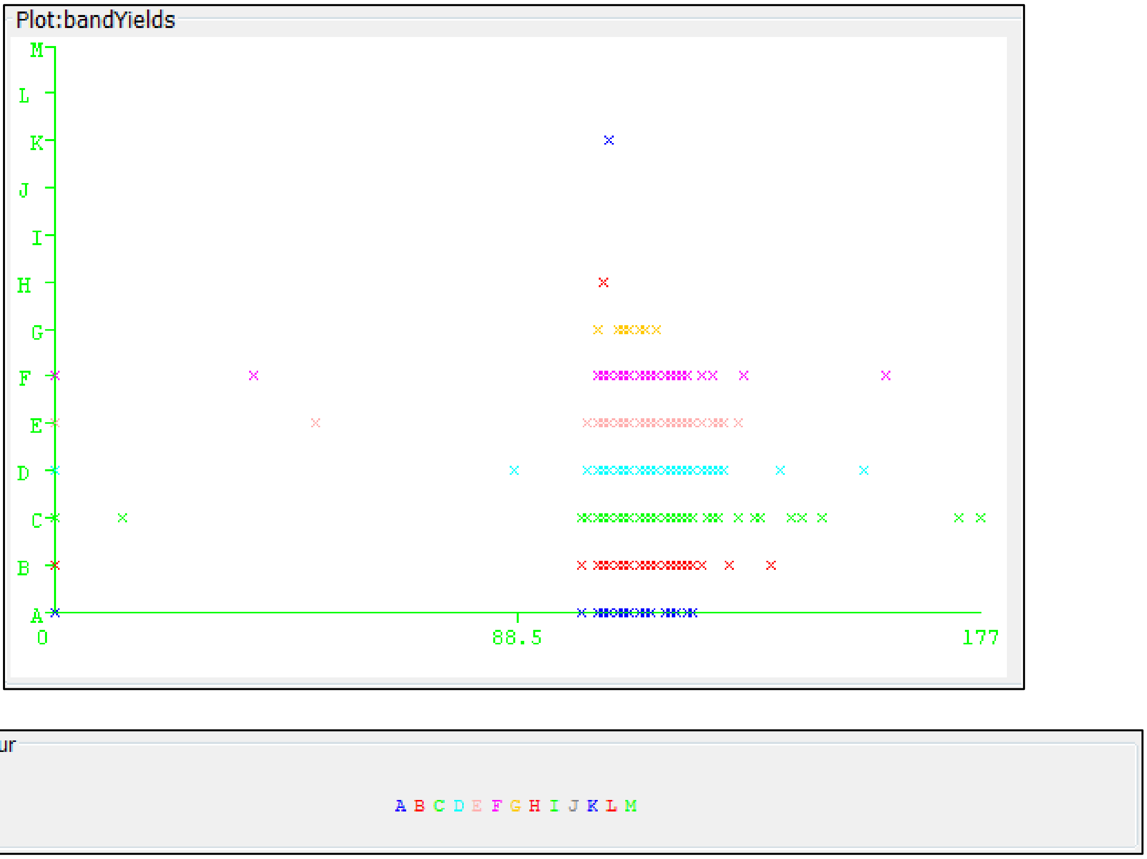


Figure 12 b). Yield categories plotted against band 2 values.

Graphs plotted against bands 2, 3 and 4 (Figures 12 b), 12 c), 12 d)) do not have any notably visible patterns. That is not to say that yield is not dependent on them, the dependence just cannot be seen in the visualization. This is why we need computerized mathematical models that extract information. Graphs for bands 5, 6 and 7 (Figures 12 e), 12 f), 12 g)) show that class B (20-22 tons per acre) and class C (23-25 tons per acre) have a narrower range of band values as compared to class D (26-28 tons per acre), class E (29-31 tons per acre) and class F (32-24 tons per acre). Band 7 (Figure 12 g)) plot shows a very narrow range of 70-115 for class B.

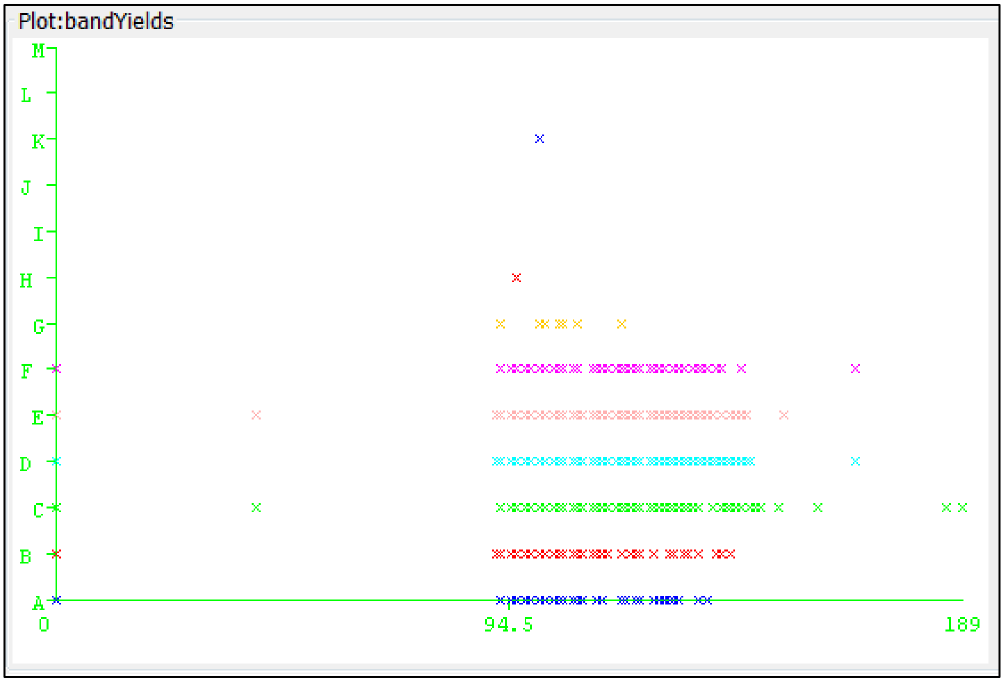


Figure 12 c). Yield categories plotted against band 3 values.

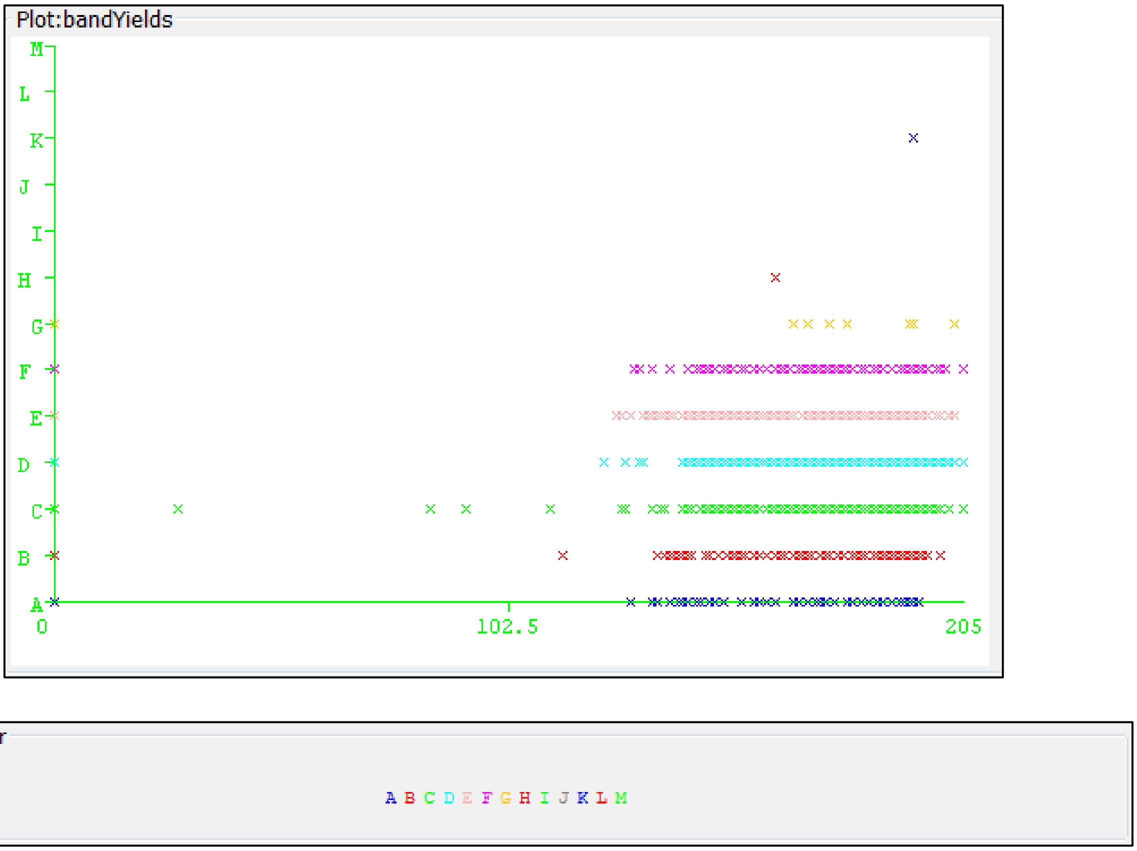


Figure 12 d). Yield categories plotted against band 4 values.

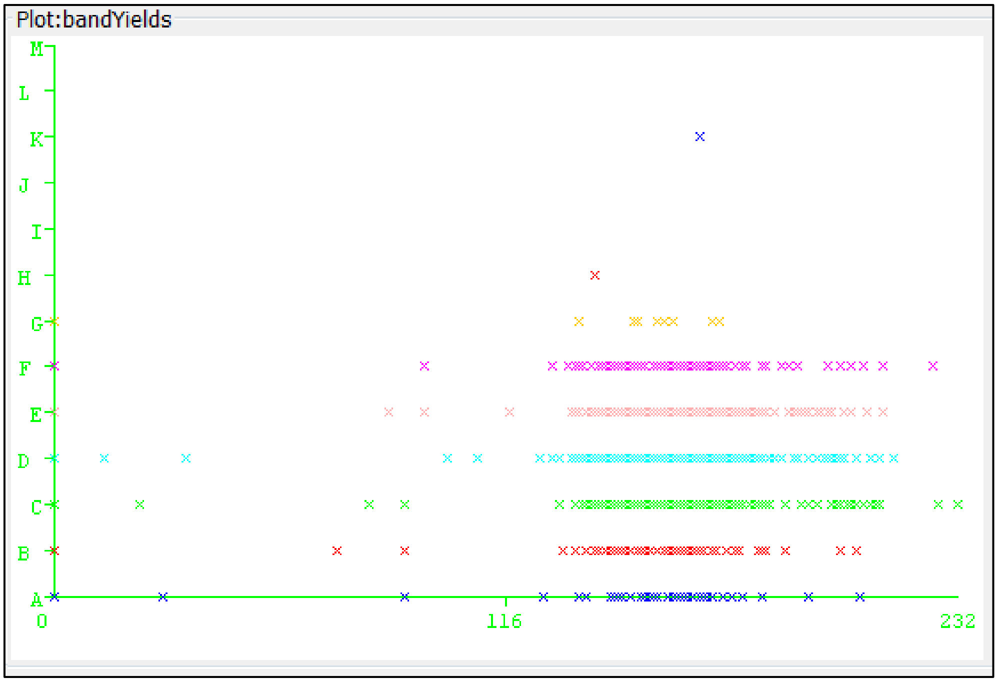


Figure 12 e). Yield categories plotted against band 5 values.

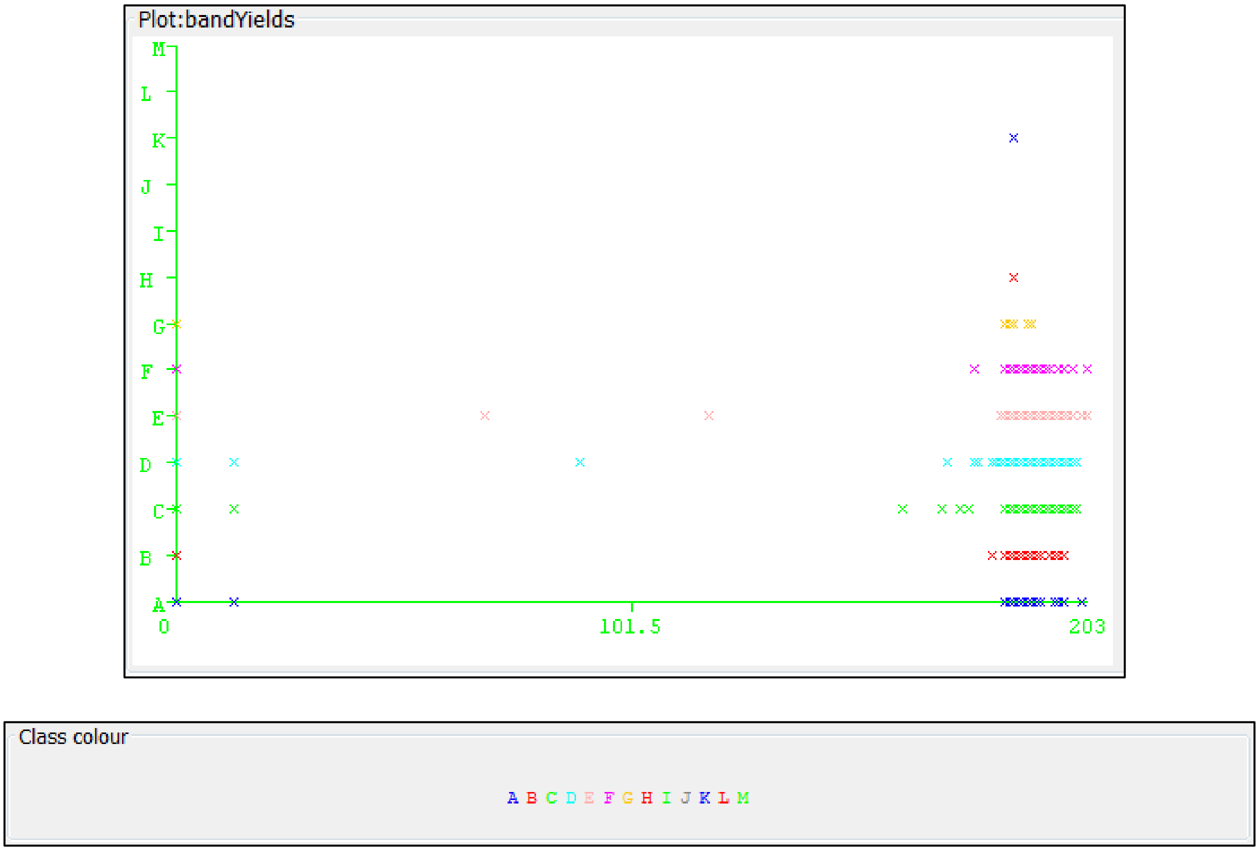


Figure 12 f). Yield categories plotted against band 6 values.

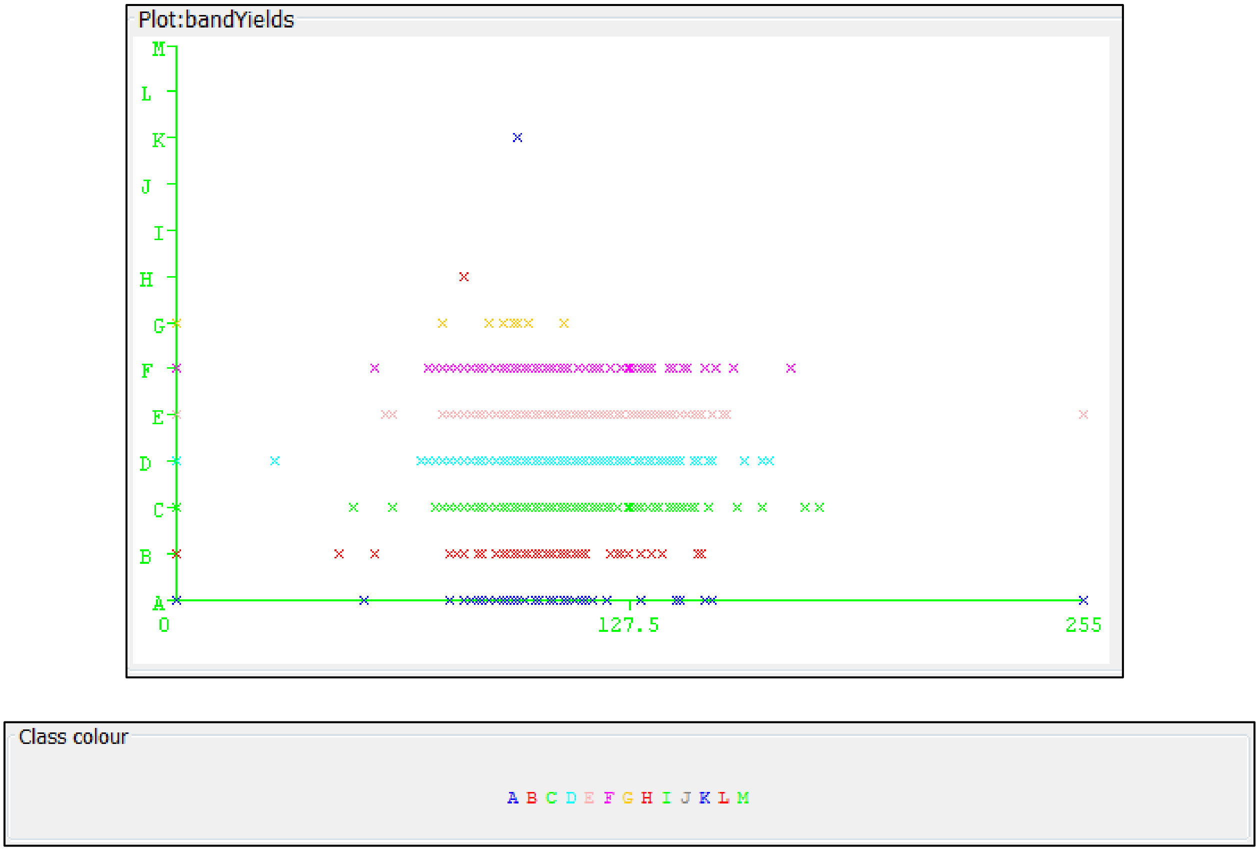


Figure 12 g). Yield categories plotted against band 7 values.

4.2. Classification

As seen in the previous section, visualization does indicate a hint of patterns between yield and band values reflected in the image. However, that information is only one way to understand the dependence. It is not very helpful in making yield predictions. To make yield predictions, mathematical models are needed, which can be accomplished by classification. When the exported .arff file is supplied to the classification algorithm, a summary of results is output to the user. These results contain the number of correctly classified instances, the corresponding percentage, number of incorrectly classified instances and the corresponding percentage, kappa statistic, mean absolute error, root mean squared error, relative absolute error, root relative squared error. and the total number of instances. The output of this application for scene id ‘LT50290272010217EDC00’ is given in the appendix. Figure 13 shows the plot of percentages of correctly classified instances for three test data sets which are training data set, 5-fold cross-validation and 10-fold cross-validation for J48 and Random Forest methods.

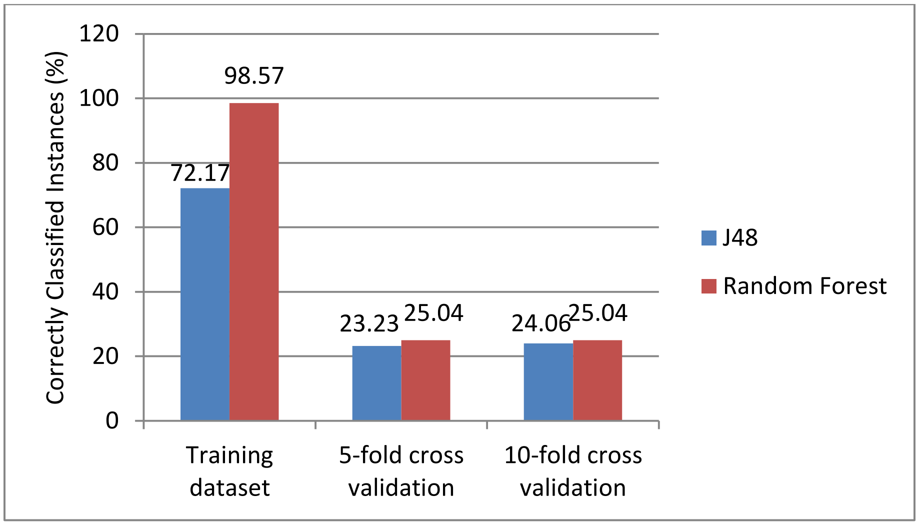


Figure 13. Correctly classified instances (%) by J48 and Random Forest algorithms using different data sets for testing.

Another significant parameter that is obtained from the output is the root mean squared error because it reflects the average magnitude of error and gives higher weightage to large errors. Figure 14 shows the plot of root mean squared error using both algorithms on three kinds of testing data sets.

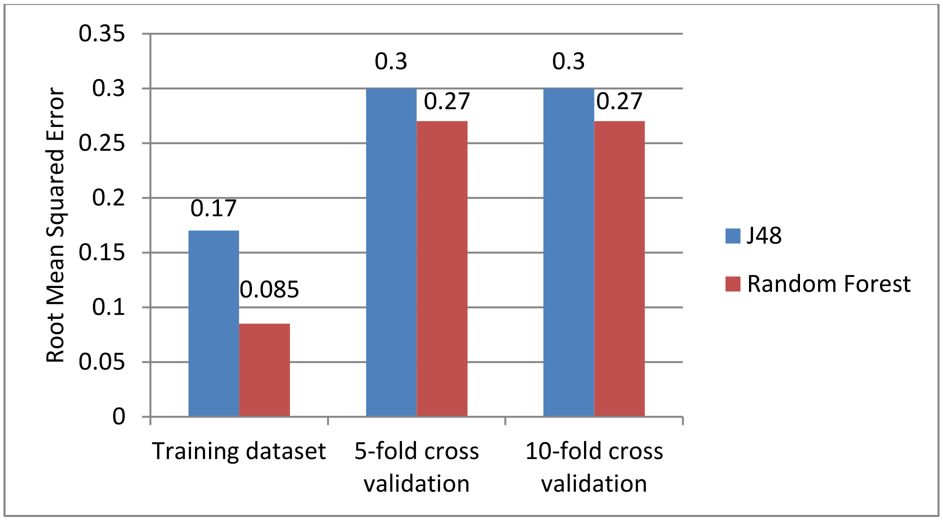


Figure 14. Root mean squared error by J48 and Random Forest algorithms using different data sets for testing

5. RELATED WORK

There has been research on using different types of data in order to predict yield. One such research project [18] was based on soil and property data along with topographical data for three fields located near Centralia in Central Missouri. The topographic data was obtained for each field using a Nikon Topgun A200LG total station surveying instrument. This data was then processed using a series of methods including block kriging and terrain model analysis. It mainly focused on two objectives—to evaluate and compare the predictive accuracies of various function approximation techniques, including supervised feed—forward neural networks (NN), projection pursuit regression (PPR), and stepwise multiple linear regression (SMLR), in relating crop yields to topography, and soil parameters and yield estimation.

NN methods produced the minimal standard error of prediction (SEP) results of all the methods in every site-year. The rprop NN technique was consistently superior to the other techniques, producing minimal SEP results in 6 out of the 10 site-years. Nonlinear techniques, both NN and PPR, showed only small gains over SMLR in site-years with small data sets and in site-years when water stress was minimal.

Another interesting research project [19] involves developing highly flexible models to estimate various soil and crop parameters based on data obtained from airborne hyperspectral imagery. These images of corn in eastern Canada (under different fertilization rates and various weed management protocols) were obtained from a compact airborne spectral imager. These images were used to estimate spectral reflectance values (for wavelengths, 408 to 947 nms) for each pixel with the help of ENVI software. Statistical and artificial neural networks were used to develop the yield prediction models. Principal component analysis was used to reduce the number of input variables. Greater accuracy (about 20% validation RMSE) was obtained with ANN as compared to any of the three empirical methods based on normalized difference vegetation index, simple ratio or photochemical reflectance index. No clear difference was observed between SMLR and ANN methods.

6. CONCLUSION AND FUTURE WORK

As shown in chapters 2 and 3, the goal of using data mining framework alongside spatial data with the help of Geospatial Data Abstraction Library (GDAL) has been achieved. GIS (simulated by GDAL library) was extended to have data mining functionality. This involved obtaining georeferencing information from the GIS for converting longitudes and latitudes of points with known agricultural values into pixel level coordinates, exporting those values and band values at the converted coordinates, and applying data mining models on them for classification. From the visualization in chapter 4, it can be speculated that band values do impact the yield at a given point. The software was also able to produce measures that reflect the accuracy of the prediction model. Hence, this paper presents a prototype of a system that allows data mining to interface with geo-spatial information.

In terms of what can be done next, the project can incorporate pattern mining which would help find association rules among band configurations. Using regression techniques for data mining would yield a model with low root mean squared error. Lessening the number of yield categories/classes along with using more GCPs should make the system more accurate. Another promising improvement would be achieved by using images obtained from hyperspectral sensors such as AVIRIS, HyMap, Hyperion, etc. These sensors measure reflectance in more bands that have narrower bandwidths [20].

REFERENCES

[1] N. Naga Saranya, M. Hemalatha, & James Ward. "Integrating Spatial Data Mining Technique to Identify Potential Landsat Data using K-Means and BPNN Algorithm." *International Journal of Computer Applications*, 30, pp. 16-21, Sep 2011.

[2] "Precision agriculture." Internet: http://en.wikipedia.org/wiki/Precision_agriculture, May 27, 2013 [May 30, 2013].

[3] Georg Ruß, "Data Mining of Agricultural Yield Data: A Comparison of Regression Models," 9th Industrial Conference, ICDM 2009, Leipzig, Germany, 2009, pp. 24-37.

[4] "Global agricultural product market shows strong growth." Internet: http://www.allaboutfeed.net/Process-Management/Management/2012/1/Global-agricultural-product-market-shows-strong-growth-AAF012664W/, Jan. 16, 2012 [May 30, 2013].

[5] "Landsat Multispectral Scanner (MSS) and Thematic Mapper (TM)." Internet: http://iic.gis.umn.edu/finfo/land/landsat2.htm, Apr. 7, 2011 [May 30, 2013].

[6] "Data Mining." Internet: http://en.wikipedia.org/wiki/Data_mining, May 27, 2013 [May 30, 2013].

[7] "Georeference." Internet: http://en.wikipedia.org/wiki/Georeference, Mar. 17, 2013 [May 30, 2013].

[8] "Ground Control Point." Internet: http://marswiki.jrc.ec.europa.eu/wikicap/index.php/Ground_Control_Point, Jan. 28, 2008 [May 30, 2013].

[9] "Least Squares." Internet: http://en.wikipedia.org/wiki/Least_squares, Mar. 30, 2012 [May 30, 2013].

[10] "ARCGIS 9.2 online help Georeferencing a raster data set." Internet: http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Georeferencing_a_raster_data set, Sep. 22, 2008 [May 30, 2013].

[11] Brian Walshe. "Using AstroWeka to Classify SuperCOSMOS data." Internet: http://astroweka.sourceforge.net/example/index.html#use, [May 30, 2013].

[12] "Classification Methods." Internet: http://www.d.umn.edu/~padhy005/Chapter5.html, [May 30, 2013].

[13] "Decision Trees – C4.5." Internet: http://octaviansima.wordpress.com/2011/03/25/decision-trees-c4-5/, Mar. 25, 2011 [May 30, 2013].

[14] Leo Breiman. "Random Forests-Random Features." Internet: http://www.stat.berkeley.edu/~breiman/random-forests.pdf, [May 30, 2013].

[15] Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, & Bradley P. Feuston. "Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling." *J. Chem. Inf. Comput. Sci.*, vol. 43, pp. 1947-1958, 2003.

[16] Bryan Lemon. "The Effect of Locality Based Learning on Software Defect Prediction." Western Virginia University, 2010.

[17] Celine Vens & Fabrizio Costa, "Random Forest Based Feature Induction." Data Mining (ICDM), 2011 IEEE 11th International Conference, Vancouver, British Columbia, Canada, pp. 744-753, 2011.

[18] S. T. Drummond, K. A. Sudduth, A. Joshi, S. J. Birrell, & N. R. Kitchen. “Statistical and Neural Methods for site-specific Yield Prediction.” Transactions of the ASAE, vol. 46, pp. 5-14, 2003.

[19] Y. Uno, S. O. Prasher, R. Lacroix, P. K. Goel, Y. Karimi, A. Viau, & R. M. Patel. “Artificial neural networks to predict corn yield from Compact Airborne Spectrographic Imager data,” Computers and Electronics in Agriculture, vol. 47, pp. 159-161, May 2005.

[20] Ludmila Monika Moskal, Kevin P. Price, Mark E. Jakubauskas, & Edward A. Martinko. “Comparison of Hyperspectral AVIRIS and Landsat TM Imagery for estimating burn site pine seedling regeneration densities in the central plateau of Yellowstone National Park.” Proceedings of the 3rd International Forestry and Agricultural Remote Sensing Conference and Exhibition, Denver, CO, 2001.

APPENDIX. PROGRAM OUTPUT

The table below shows the output from the program when a scene downloaded from http://landsat.usgs.gov and file containing points with known yield data (obtained from American Crystal Sugar Cooperative) was supplied to the built tool. The yield data and the landsat image were recorded/captured at the same time.

Scene ID	LT50290272010217EDC00
Date Acquired	5th August, 2010
Date of Yield Recording	August, 2010
Image width	8281
Image Height	7491
Number of bands	7
Crop	Sugar beet
Number of Instances	1326
Program Output (Random Forest,10-fold cross-validation)	Random forest of 10 trees, each constructed while considering 4 random features. Out of bag error: 0.8015 Correctly Classified Instances 332 25.0377 % Incorrectly Classified Instances 994 74.9623 % Kappa statistic 0.0231 Mean absolute error 0.1193 Root mean squared error 0.2674
Program Output (J48,10-fold cross-validation)	Correctly Classified Instances 319 24.0573 % Incorrectly Classified Instances 1007 75.9427 % Kappa statistic 0.0169 Mean absolute error 0.1192 Root mean squared error 0.3018

Scene ID	LT50290272010217EDC00
Program Output (Random Forest, training data set)	Random forest of 10 trees, each constructed while considering 4 random features. Out of bag error: 0.7873 Correctly Classified Instances 1307 98.5671 % Incorrectly Classified Instances 19 1.4329 % Kappa statistic 0.9816 Mean absolute error 0.0303 Root mean squared error 0.0849
Program Output (J48, training data set)	Correctly Classified Instances 957 72.1719 % Incorrectly Classified Instances 369 27.8281 % Kappa statistic 0.6378 Mean absolute error 0.0575 Root mean squared error 0.1695
Kappa statistic	A measure that takes into account the agreement occurring by chance
Mean Absolute error	Average of the absolute errors
Root mean squared error	Average of squares of differences between actual and predicted values