A Hoeffding Decision Tree Based Approach for Soil Classification

Dakota Soares, M. Ali Akber Dewan*, Fuhua Lin

School of Computing and Information Systems, Faculty of Science and Technology
Athabasca University, Edmonton, Canada
dsoares1@athabasca.edu, adewan@athabascau.ca, oscarl@athabascau.ca

Abstract. Soil classification is an important but challenging problem for the research community. As such, current solutions for classifying soil for a wide variety of reasons are out of the reach of hobbyists and the small research firms. This research study focuses on comparing various machine learning algorithms on a custom database generated from Canadian System for Soil Classification (CSSC) attributes to reveal a solution for identifying a soil Pedon. Discussion centres around acquainting the user with soil terminology, current solutions to the problem of soil classification, and the proposed solution. A database using these attributes was constructed, and six algorithms were analyzed using validation, test case, and 70-30 split testing via WEKA. Among the comparing algorithms, the Hoeffding decision tree was found to perform best, and it was subsequently used in developing a simple prototype using Java Graphical User Interface (GUI). Finally, the Hoeffding decision tree was compared to the other algorithms that were used to see why it was more accurate than its competitors.

Keywords: Soil classification, artificial intelligence, machine learning, Hoeffding decision tree.

1 Introduction

Soils come in a diverse range of types and automatic classification of soil type is still challenging. The goal of this paper is to present a simplified soil classification scheme utilizing machine learning models that allow researchers and enthusiasts to ascertain the great group and order of a soil, based on several of its observable characteristics. Soil types differ from region to region, and even soil types themselves are guaranteed to have a range of properties depending on external factors that influence the soil's composition (e.g., uneven weathering, climate, etc.). This makes the classification task difficult. In the context of this discussion, soil is defined naturally occurring unconsolidated mineral or organic material at least 10 cm thick that occurs at the earth's surface and can support plant growth [1]. It does not include the bedrock or underlying lithospheric structure under the soil matrix itself.

Despite the inherent complications when classifying soil, there are some universal guidelines that can assist in classification efforts. A square metre section of soil is termed a pedon, and every soil pedon has one or more soil horizons, labeled with the A, B, and/or C suffix as shown in Figure 1. The "A" horizons refer to topsoil, "B"

* Corresponding author

horizons are regolith, and “C” horizons are saprolites. Of course, there are some complications to this general arrangement. Soils with excessive organic matter have an “O” as their first horizon suffix. In addition, prefixes can be added onto the horizon letter to describe the soil horizon more accurately. For example, an Ae horizon is an A horizon characterized by clay, Fe, AI, or organic eluviation.

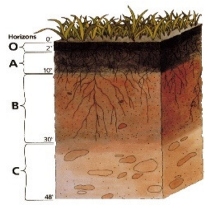


Figure 1: Soil Pedon with O (Organic), A, B, and C Horizons. (NE Soil, N.D.) [2]

Using horizon information, along with other soil properties, Agri-Food Canada began collecting soil data information in the early half of the 20th century. The purpose of classifying soils was (and still is) to provide a framework for formulating hypotheses about soil genesis and the response of soil to management [1]. Originally, each province relied on its own system of soil classification. In the mid 20th century, the National Soil Survey Committee (NSSC) [3] was formed, and standards were universalized. Agri-Food Canada currently defines eleven soil orders spread across the country. There are broken into great groups (number thirty-one in total). Great groups are further broken down into sub-groups.

In this paper, to find a simple and effective solution for soil classification, six mahine learning models were compared, which include J48 decision tree, random forest, Naive Bayes, Bayesian Net, K* (KNN), and Hoeffding tree. Along with the comparisons, a detailed and indepth investigation has been done into why the top performing algorithm appears to be more successful than the others. Finally, a working prototype has been developed and tested with the best performing algorithm using Java Graphical User Interface (GUI) which allows users to select soil properties based on their observations. The program runs the best performing algorithm and attempts to provide the user with the correct great group and soil order based on the data received by it from the user. This provides a simple and effective solution for soil classification for the end users. The prototype can be accessed and downloaded through the link: https://github.com/DakotaCS/CSSC-Application.

2 Background

Currently, industrial soil classification programs offered on the market are virtually inaccessible for small business owners, hobbyists, and small-scale research laboratories due to their price. Many programs offer far too many options, are prohibitively expensive, and/or require specialized equipment or personnel to operate. In addition, the programs are developed primarily in the USA and do not include any CSSC classifications – rendering them unsuitable in a Canadian context.

Compounding the problem is the fact that soil classification itself is a niche market. From research conducted for this report, only three programs were found that classified soil. GAEA's WinSieve Grain Size and Soil Classification Software [4], while suited for geotechnical surveys when it comes to oil, gas, and mining operations, is prohibitively expensive running into thousands of dollars per license. Geosystems Software's CLSuite v.4, a competitor, only includes USDA soil classifications and costs and is comparatively priced. Finally, the US Department of Agriculture's Soil Data Viewer, while free, is woefully outdated. Mobile applications for classifying soils are effectively non-existent.

From a literature standpoint, papers like Robertson [5] and Bhattacharya et al. [6], while providing theoretical groundwork and were helpful in choosing the algorithms and organizing the dataset, were deficient in a couple of areas. From using penetrometers to map stratigraphic profiles (a procedure that is too expensive to be incorporated here), to using machine learning to predict new values (the project aims to classify, not predict), the current academic literature did not appear to assist in finding a solution for the problem posed in any meaningful way.

3 Soil Classification Framework

3.1. Overview

The need for a cost-effective soil classification system that deals with CSSC classifications should be apparent at this point. The solution proposed here includes a program that is easy to use, portable, and gives the user detailed feedback about the classification based on their choice of properties/attributes (see Figure 2). The program is meant to obtain a preliminary classification that can be verified later through official means (i.e., geotechnical survey completed by a licensed engineer). Having this knowledge available earlier in any planning process is beneficial and is where the program can best be used.

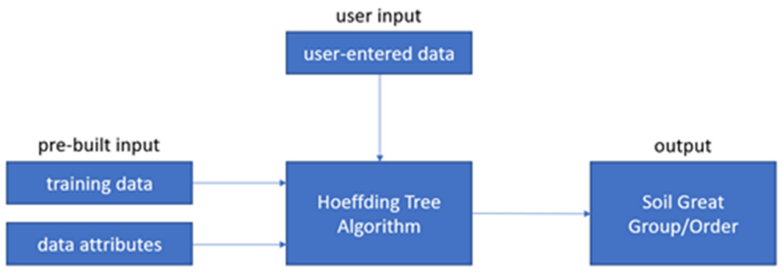


Figure 2: Program Input/Output Sequence

The experimental program collects data from the user, then runs a machine learning algorithm using training data and the user's input. The result is a soil classification at the great group level. The soil order, great group, accuracy of the prediction, and other relevant information about that soil will be displayed to the user. The program is written in the Java programming language utilizing the Waikato Environment for Knowledge

Analysis (WEKA) API. To allow for future portability, Java was preferred as it is easily portable to the Android OS and is better in terms of performance in this context then languages like Python. Training data is obtained from a WEKA compatible *.arff database file. User input is collected via a series of textboxes (for numeric data) and combo boxes (for nominal data).

3.2. Machine Learning Models for Soil Classification

In terms of algorithms used in the classification scheme itself, six were chosen. The first, and most obvious choice, was the J48, C4.5 Decision tree algorithm [7]. Considered the workhorse of the data mining world, this algorithm is built off Quinlin's ID3 tree concept [8] using the concept of information entropy. In a nutshell, the algorithm picks the attribute that is the most effective at splitting (the split is based on information gain) the sample sets into subsets, based on class enrichment.

The second algorithm was the Random Forest [9]. In this scheme, multiple decision trees work together (termed an ensemble). Every tree makes a class prediction, and the Random Forest algorithm selects the class that has been outputted the most from the set of decision trees. Because each tree runs its own prediction, it is insulated from any other tree, reducing the likelihood of error.

The third algorithm used is the Naïve Bayes algorithm [10]. The assumption is made in this scheme that the presence of a particular feature in a class is unrelated to the presence of any other feature. The classifier is based on Bayes' theorem. (Bayes' theorem gives the probability of an event based on conditions that may have caused said event). The algorithm works well on large datasets. Naive bayes was chosen because the dataset is larger, some attributes are nominal (text), and the problem has multiple classes – all of which fit well within the scope of the Naïve Bayes classification scheme.

The fourth algorithm is the Bayesian network classifier [11]. As Mihaljevic, Bielza, and Larranaga [12] state that a Bayesian network classifier is simply a Bayesian network applied to classification, that is, the prediction of the probability $P(C|X)$ of some discrete (class) variable C given some features X. In other words, the Bayesian network classifier is a Bayesian network that is used when predicting a discrete class variable. It assigns predictor variables in a set to the most probable class. It utilizes a directed acyclic graph with each variable at a node encoding conditional independencies.

The K* algorithm [13] is the fifth algorithm that was used on the dataset. Also referred to as the KNN (K-Nearest Neighbours algorithm), it is a simple machine learning, non-parametric supervised classifier. The classifier is based on the idea of similarity (proximity). It uses points on a graph to calculate its classifications. Worth noting here is that the K* algorithm used for this project relied on the entropy-based distance function.

The final classifier used on the dataset was the Hoeffding tree (proposed by Hulten et al. [14]). An incremental anytime decision tree, the Hoeffding tree is used for large amounts of data. The tree grows incrementally based off the theoretical guaranteed Hoeffding bound. If sufficient statistical evidence exists, a decision at that point is made and the tree is split (hence the term optimal splitting). If the model and its training instances is big enough, the Hoeffding tree becomes indistinguishable from non-incremental learners. Hoeffding trees are advantageous due to the fact that they are highly accurate on small samples, incremental, and never scan the same data twice.

4 Experiments and Analyses

4.1. Dataset

Having a good dataset is crucial to ensure the machine learning algorithm is trained as accurately as possible. In addition, a second requirement is that any piece of data in the dataset must be easily obtained through observation or cursory analysis. For example, while determining the grain size of a soil horizon would lead to an accurate classification, the tools needed to get a precise laboratory reading are expensive and complicated, rendering this as a metric that could be left out. However, the colour of a Pedon is something that is easily observable and would work well in determining at least a subset of potential soil classifications. The dataset itself consists of 142 lines of classes and attributes. The dataset was created exclusively for this project from data obtained in the official Canadian System for Soil Classification manual, 3rd Edition. The dataset contains 123 classes. Inspiration was gained from Michalski & Chilausky's expert system database for soybean disease diagnosis [15].

Continuing the example above, while the general colour of a Pedon is helpful, it is not as helpful as grain size. Thus, other metrics need to be accumulated to differentiate soil classifications that are close in nature. For example, a soil labeled "brown" could be either a gray-brown Luvisol, or a Podzolic Folisol. In-depth analysis of the CSSC manual (3rd edition) revealed 12 attributes that could easily be observed or calculated using rudimentary instruments as listed in Table 1. The twelve attributes above allow each soil classification to be unique. While several classifications come close to one another, each does have at least one unique attribute which will allow the algorithm to classify the user's data with a high degree of accuracy. As demonstrated later, the choice of these twelve resulted in a high accuracy rate during classification.

Table 1: Dataset attributes

Attributes	**Type**	**Notes**	**Expected Values**
Biome	Nominal	There are 8 biomes.	decidious-forest, coniferous-forest, hot-desert, cold-desert, wetland, plains, pond, lacustrine
Colour	Nominal	There are 12 possible colour combinations. The colour is defined as the general colour of the entire Pedon.	none, gray, brown, dark-brown, brown-orange, black, gray-brown, yellow, yellow-black, yellow-brown, gray-black, red-brown
Predominant Horizon	Nominal	There are 13 options. The predominant horizon is defined as the thickness horizon in the Pedon.	none, Om, O, Of, B, Ae, Bf, Ah, Bh, Bhf, Btg, Ah-Ap, any

Special characteristic s	Nominal	There are 9 special characteristics.	none, fibric, cryoturbated, organic, gleyed, mineral, warm-soil, clay-enriched, slickenslide
Soil forming process	Nominal	There are 7 soil forming processes.	eluvial, alluvial, eolian, volcanic, glacial, climatic, biogenic
Acidity	Numeric (pH scale)	Acidity uses the pH scale which runs from 0 (base) through 7 (neutral) to 14 (acidity). Thus, decimal numbers between 0-14 are acceptable.	0 – 14
Moisture content	Numeric	Represents the water capacity in a soil by inches/foot in depth. Represented on a decimal scale from 0.25-2.50.	0.00 – 2.50
Consistency	Nominal	Consistency is defined as how smooth or rough the soil feels – this can be deducted simply by handling a sample. There are 4 values for this attribute.	fine, semi-fine, semi-coarse, coarse
Porosity	Numeric (percentage)	Porosity is the ratio of the volume of pores to the volume of bulk material. This is expressed as an integer representing a percentage.	0 – 100
Thickness	Numeric (mm)	The thickness in this context represents the thickness of the predominant horizon. It is represented in centemeters and is a decimal value.	0 – 100
Organic content	Nominal	Because organic soils are so widespread, this attribute was broken out of the special characteristics and soils can have more detail – soil can be not organic, organic, semidecomposed, or decomposed.	no, yes-decomposed, yes-semi-decomposed, yes
Carbonate	Nominal	Carbonate soils have minerals that fizz in hydrocloric acid. This value may be true or false (inputted as yes/no)	no, yes

Information above only pertains to the attribute list. The training data not only lists the optimal values of each attribute, but also values that differ slightly from the optimal soil classification. For example, the organic order has the Mesisol, Humisol, and Fibrisol great groups. These are both dark, oxygen-rich, peat-like soils that can be easily misclassified by the user. To obtain an accurate classification, the machine learning algorithm must be given a subset of possible value for each classification. Sample permutations arising from the Fibrisolic great group of the Organic order are shown below. Note that the attributes in green are the correct attributes for the Fibrisolic great group. The other permutations are possible values that the user could incorrectly enter. To correctly classify the great group, the dataset needs to ensure that a variety of potential cases are considered. These have been added to the dataset, extending the optimal set of eleven great group training sets to over seventy variations (eleven are the correct attribute lists, and the remainder are variations to the correct lists). Figure 3 presents the scenario where the user can potentially misclassify the pedon they are observing. For the sake of brevity, the dataset was allowed to have, at most, three permutations per great group to constrain the dataset and thus prevent it from logarithmically increasing.

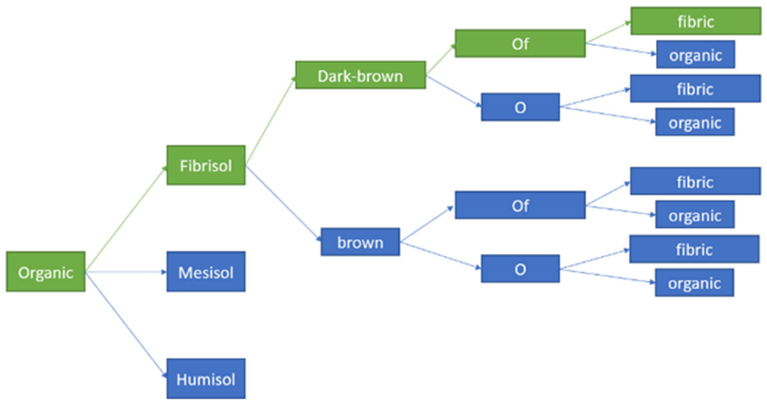


Figure 3: Possible permutations from the fibrisolic great group (partial tree)

4.2. Results and Discussions

Using WEKA, six algorithms were tested against the training data and compared to one another. Three tests were run on each classifier per algorithm, for a total of eighteen tests. The tests vary in scope, allowing for an accurate picture of how the algorithm will perform. In the first round of testing, the percentage split option was utilized and the training data was split into a test subset and training subset. In this round, 70% of the data was used as training data, and 30% used as test data. The second set of tests utilized k-fold cross-validation with $k = 10$. This method partitions the training dataset into k subsets. One subset is analyzed, and the results are validated on the other subsets (validation sets). To ensure maximum accuracy, tests were completed using $k = 10$ rounds of cross-validation. The final round of tests involved using the training dataset as the test dataset. In this scenario, the classifier would use the training set to train itself, then use it again as a test data set. The accuracy percentage for these final tests should be quite high if the dataset is built well. All the six algorithms were run utilizing the three testing methods outlined above and the results are shown in Table 2 in the form of weighted averages.

Table 2: Test results using true positive, precision, and recall

	True Positive			Precision			Recall		
	70-30 split	10-fold validation	Test data as training data	70-30 split	10-fold validation	Test data as training data	70-30 split	10-fold validation	Test data as training data
J48 C4.5 Decision Tree	62.16	78.86	91.05	0.607	0.819	0.966	0.622	0.797	0.935
Random Forest	72.27	82.11	95.12	0.833	0.850	0.959	0.757	0.846	0.951
Naive Bayes	78.38	74.79	93.49	0.890	0.729	0.961	0.784	0.764	0.927
Bayesian Network	62.16	79.67	95.12	0.813	0.799	0.975	0.622	0.780	0.951
K* (KNN)	72.97	86.99	95.12	0.828	0.866	0.975	0.730	0.862	0.951
Hoeffding Tree	86.48	85.37	95.12	0.924	0.855	0.975	0.865	0.854	0.951

A high TP-rate, high recall, and low standard error made the Hoeffding tree the algorithm of choice. While other algorithms (i.e., K*) performed better at certain tests, on the whole, the Hoeffding tree maintained stable results and high accuracy across all tests. The test results varied as shown above – however a cursory glance at the graph shows the Hoeffding tree beats the other decision trees by a fair margin and completely outperforms the Bayesian classifiers.

5 Prototype Development of Soil Classification System

5.1. Rationale for Choosing Hoeffding Tree Algorithm

As we mention earlier, a simple prototype implementation of the soil classification system has been done based on the CSSC standard in this research study and can be downloaded from the following link: https://github.com/DakotaCS/CSSC-Application.

Among the six algorithms compared, Hoeffding Tree algorithm has been choosen for the protype implemention based on our observations and the test results. The Hoeffding Tree algorithm is an incremental, anytime decision tree, utilizing a Hoeffding bound, that is useful when analyzing large amounts of data. In other words, the algorithm is made up of the Hoeffding tree algorithm (HTA) and Hoeffding bound (HB). To outline the rationale behind choosing the algorithm, a more in-depth explanation of the algorithm is required.

The tree is built by initializing a root leaf (this leaf and all subsequent child leaves hold statistics of attribute values). The tree is grown through recursion – replacing leaves with decision nodes. A split check is initiated if a new sample is pushed down the tree and the values in the sample warrant a split after being heuristically evaluated through the information gain split or Gini split strategies. The pure very fast decision tree (VFDT) uses information gain as the splitting strategy, and this strategy was used here. The Hoeffding bound is used to choose a split attribute at a decision node. The induction phase of the tree splits the tree branches, checks tree nodes, and breaks ties. In this study, we emphirically set the Hoeffding tie threshold into 0.05.

In the prediction phase, the Hoeffding tree may use the majority class, Naive Bayes, or Naïve Bayes adaptive strategy. Fong and Yang [16] argue that a Naive Bayes is better to maximize posterior probabilities given Bayes' rule. For this project, the newer Naïve Bayes adaptive strategy was utilized. In this case, leaves store an estimation of the current error. The weight of each node in the voting process is proportional to the square of the inverse of the error [17].

The rationale for choosing this algorithm utilizing the parameters discussed is two-fold. First, the algorithm performed well on the various tests (see Section 4). Secondly, the algorithm's Naive Bayes adaptive strategy allowed it to accurately apply Bayes' theorem with strong independence. Tests utilizing the majority class strategy failed to achieve desirable results and hence the Naive Bayes adaptive was chosen.

A final point worth noting is the reason behind why the algorithm worked better then the other algorithms on this dataset. Based on the research conducted, we put forward a few hypotheses directed individually at each competitor algorithm, demonstrating why it was less accurate, and in this case, why the Hoeffding algorithm is more accurate.

When it comes to the J48 algorithm, the Hoeffding bound allows for optimal splitting of the tree – the J48 algorithm does not utilize this method. Using the default WEKA J48 C4.5 algorithm, the info gain splitting mechanism is used (which is the same as the Hoeffding tree). However, the J48 algorithm does not combine any Naive Bayes prediction. This makes the J48 algorithm in this case more prone to over fitting. Another aspect worth considering here is that the zero values in the dataset will result in empty branches – artificially widening the tree [18].

The Naive Bayes Adaptive leaf predictor in the Hoeffding tree avoids the sparse data problem. This issue plagues the Random Forest algorithm. The CSSC dataset presented in this project includes many instances that have attributes initialized at zero. Based on the tests, this issue is acute in the 70-30 split test.

In terms of the Naive Bayes algorithm, it works well as a leaf predictor within the Hoeffding tree but is sub-optimal when run by itself. Some of the attributes in the CSSC dataset are not independent of one another, and some attributes are more important than others (i.e., the colour of a soil should be more indicative when predicting its soil great group then the attribute regarding the thickness of the predominant horizon). Despite these minor issues, the algorithm performed quite well. It's biggest drawback is that it is naïve, and is generally not correct in real-world situations. As the intention was to construct a simple implementation of this project, the algorithm would not perform as well as the Hoeffding tree in real-world scenarios.

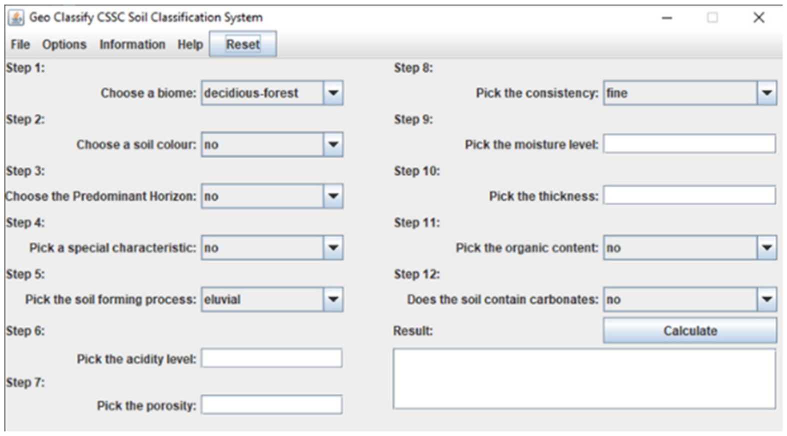
The Bayesian network (Bayes Net) algorithm did poorly across all tests. This failure can be attribute to the fact that each variable in the dataset is not conditionally independent from its non-descendants, given its parents. The final test revealed this fact when the algorithm broke down and could not handle the scenario where thirty percent of the training data instances were used as test data in any meaningful way. The Hoeffding tree is not based on conditional independence – the backbone of the Bayesian network algorithm and naturally performed better.

Finally, the Hoeffding tree in this case is more robust when handling large amounts of data, as well as outliers and missing values. The K* (KNN) algorithm handles neither well. In addition, it does not work well with high dimension data. While in this set, $p < N$ (where p is the total number of dimensions and N is the instances in the test data), compared to other WEKA datasets, the dimension is quite high. These issues (especially the latter) seemed to impact the K* algorithm's performance. The Hoeffding algorithm itself is not perfect. However, based on the analysis above, it appears to be the one best suited for the task.

5.2. Prototype Testing

The dataset and algorithm were implemented as a Java application as part of this resarch study. The program is straightforward and directs the user to answer a series of questions. From this data, the algorithm is used to calculate the great group, and subsequence soil order using the attributes provided. Figure 4 shows the user interface of the implemented prototype. The protype utilizes the WEKA API (specifically the weka.classifiers.trees.*). Training data via a database formatted in Attribute-Relation File Format (ARFF) is used to populate the Hoeffding tree. The test data consists of the user's answers to the questions posed in the main GUI.

There were a couple issues that were found in the prototype application. There appeared to be lower accuracy during classification in the prototype program versus the database testing that was done using the WEKA.



Step 1: Choose a biome: decidious-forest
Step 2: Choose a soil colour: no
Step 3: Choose the Predominant Horizon: no
Step 4: Pick a special characteristic: no
Step 5: Pick the soil forming process: eluvial
Step 6: Pick the acidity level:
Step 7: Pick the porosity:
Step 8: Pick the consistency: fine
Step 9: Pick the moisture level:
Step 10: Pick the thickness:
Step 11: Pick the organic content: no
Step 12: Does the soil contain carbonates: no
Result:

Figure 4: User interface of the prototype

The tests run in the prototype application used a subset of the soil classification database that had only base soil Pedon classes in the database. The reason behind this was to ensure the class permuations (see Figure 3 regarding how these permutations were derived from the base classes) were not unwittingly entered by the user (the chances of this happening were around seven percent). This would have created a successful result without utilizing the power of the Hoeffding algorithm.

Secondly, some variables influenced the results more then others in ways that were unexpected. For example, if the acidity was set to seven, changing this to eight would produce a different result (with the condition no other variables were changed). However, changing the porosity from twenty-five percent to seventy-five percent (again assuming no other variables were changed) would not affect the result. Some test cases and the system generated results are shown in Table 3.

Table 3: Some test scenario and the system generated results

Test Cases	User Submitted Values for the Attributes (see Table 1)	System Generated Classification Results	Intended Result	Success/Fail
1	Decidious-forest, gray, Ah, warm-soil, eluvial, 7, 50, fine, 1.25, 0, no, no	Gray-brown	Gray-brown	Success
2	Wetland, dark-brown, 0, no, biogenic, 10.75, 50, coarse, 2, 30, yes, no	Fibrisol	Fibrisol	Success
3	Lacustrine, brown, Ah, clay-enriched, 10.75, 50, fine, 2, 30, no, no	Humic_3	Humic_3	Success

4	Plains, dark-brown, Ae, slickenslide, glacial, 10, 0.5, semi-coarse, 2.25, 30, no, yes	Brown	Solod	Fail
5	Plains, gray, Ae, slickenslide, glacial, 10, 0.5, semi-coarse, 2.25, 0, no, yes	Vertic	Vertic	Success
6	Hot-desert, yellow, Ah, warm-soil, alluvial, 10, 50, fine, 1, 0, no, no	Regosol	Regosol	Success
7	Coniferous-forest, red-brown, Bh, organic, volvanic, 10, 50, coarse, 1, 30, yes, no	Ferro-humic	Ferro-humic	Success
8	Pond, gray, no, organic, biogenic, 7, 50, coarse, 1, 30, yes-decomposed, no	Humic_1	Gleyol	Fail
9	Cold-desert, yellow, any, cryotur, alluvial, 7, 20, semi-coarse, 1, 70, no, no	Turbic	Turbic	Success
10	Plains, black, Ah, no, eolian, 3, 10, coarse, 1, 40, yes, no	Black	Black	Success

6 Conclusion

The classification of a soil has many benefits to both large and small businesses that span across many industries – from agriculture to resource extraction. For example, proper identification allows for more informed decision-making in regard to construction – what kinds of structures may be built, what urban density the land can sustain, how surface runoff will behave, etc. Unfortunately, the prohibitive price of equipment, and the specialized personnel needed to operate the equipment and software programs often leave companies with a large bill and on top of that a result that may not always be favourable. A limited number of expensive programs exist (such as GAEA's WinSieve) but are not conducive for many businesses that would like a quick and accurate classification before expending more resources on further soil analysis depending on the initial result.

The solution developed in this paper was multi-faceted. First, the properties of various soil types were utilized in order to classify the great groups of every soil in the CSSC. The resulting dataset was fed into six algorithms in order to determine which gave the most accurate result. The six algorithms used in the testing included the J48 C4.5, Random Forest, Bayesian Net, Naïve Bayes, K* (KNN), and Hoeffding Tree algorithms. Every algorithm was tested three times using 10-fold cross validation, 70-30 split, and training set – for a total of eighteen tests. The Hoeffding tree algorithm was the algorithm that appeared to perform best in terms of TP-value, precision, and recall. The Hoeffding tree algorithm was implemented into a prototype to demonstrate how a potential application that solves the problem might be realized. The algorithm was paired with the base data set as training data and user-entered data to output a result.

The topic chosen for this paper is broad, and future research could be conducted on a few areas. For example, other algorithms could be tested, which could have the potential to increase the accuracy above and beyond the approximately eighty-percent threshold. In addition, the properties (attributes) chosen for classifying each soil could be refined and new attributes included. An interesting exercise here would be to prescribe each attribute with a weight (i.e., acidity could be more heavily weighted then colour). This could produce more accurate results. Finally, in terms of implementation, the Java language makes it easy to transform the program into an Android® application that could be used on a mobile device – making it even more convenient to classify soils quickly and accurately under the Canadian soil classification scheme.

References

[1] H. B. Stonehouse, The Canadian system of soil classification, Canada: Third Edition, Agriculture and Agri-Food Canada Publication, 1998.

[2] "N.A. (N.D.) Soil Horizons. NE Soil," [Online]. Available: http://nesoil.com/properties/horizons/sld001.htm.

[3] "Canadian society of soil science," Soils of Canada, 2020. [Online]. Available: soilsofcanada.ca.

[4] "GAEA," [Online]. Available: https://www.gaea.ca/proddetail.php?prod=3110. [Accessed 21 January 2022].

[5] P. Robertson, "Soil classification using the cone penetration test," *Canadian Geotechnical Journal,* vol. 27, no. 1, pp. 151-158, 1990.

[6] B. Bhattacharya and D. P. Solomatine, "Machine learning in soil classification," *Neural Networks,* vol. 19, no. 2, pp. 186-195, 2006.

[7] "Class J48," [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/J48.html. [Accessed 21 January 2022].

[8] S. L. Salzberg , "C4.5: Programs for machine learning by J. Ross Quinlan," *Machine Learning,* vol. 16, pp. 235-240, 1993.

[9] "Class RandomForest," [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html. [Accessed 21 January 2022].

[10] "Class NaiveBayes," [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/bayes/NaiveBayes.html. [Accessed 21 January 2022].

[11] "Class BayesNet," [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/bayes/BayesNet.html. [Accessed 21 January 2022].

[12] B. Mihaljevic, C. Bielza and P. Larranaga, "Learning Bayesian network classifiers," 2021. [Online]. Available: https://cran.r-project.org/web/packages/bnclassify/vignettes/overview.pdf. [Accessed 21 January 2022].

[13] "Class KStar," [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/lazy/KStar.html. [Accessed 02 02 2022].

[14] G. Hulten, L. Spencer and P. Domingos, "Mining time-changing data streams," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021.

[15] R. S. Michalski and R. L. Chilausky , "Learning by Being Told and," *Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis,* vol. 4, no. 2, 1980.

[16] H. Yang and S. Fong, "OVFDT with functional tree leaf - majority class, naive Bayes and adaptive hybrid integrations," in *International Conference on Data Mining and Intelligent Information Technology Applications*, Macao, China, 2011.

[17] "Classifiers," [Online]. Available: https://moa.cms.waikato.ac.nz/details/classification/classifiers-2/. [Accessed 02 02 2022].

[18] N. Saravanan and V. Gayathri, "Performance and classification evaluation of J48 algorithm and kendall's based J48 algorithm," *International Journal of Computer Trends and Technology,* vol. 59, no. 2, pp. 73-80, 2017.