THE COMPUTERTECH
(An International Peer Review Journal)
VOLUME 5; ISSUE 2 (JULY-DEC); (2019)
WEBSITE: THE COMPUTERTECH

Deploying Breiman's Random Forest Algorithm in Machine Learning

Divya Sai Jaladi[1*], Sandeep Vutla[2]
[1]Senior Lead Application Developer, SCDMV, 10311 Wilson Boulevard, Blythewood, SC 29016, UNITED STATES
[2]Assistant Vice President, Senior-Data Engineer, Chubb, 202 Halls Mill Rd, Whitehouse Station, NJ 08889, UNITED STATES
*Corresponding Author: divyasaij26@gmail.com

Abstract
This paper presents the implementation and evaluation of Leo Breiman's Random Forest algorithm within the Weka data mining environment, with a particular focus on addressing imbalanced datasets that hinder machine learning performance in domains such as fraud detection and rare disease diagnosis. The study enhances Weka's Random Forest functionality by integrating variable importance calculations, allowing for the identification of key predictive attributes. The implementation involves modifications to existing Weka classes and introduces new extensions to support out-of-bag (OOB) error estimation and variable permutation testing. Verification of the implementation is conducted by comparing results from Weka with those obtained from Breiman's original Fortran-based code, demonstrating comparable variable importance behavior. To simplify the use of Breiman's code, data converters and a Java-based GUI were developed. Additional improvements, such as transitioning Breiman's static Fortran code to dynamic array handling in Fortran 90, further enhance usability. This work contributes to more robust model evaluation and attribute selection in Random Forests, offering insights for future development and practical application in machine learning research and software tools.

Keywords: Machine Learning, Algorithm, Data, Training, Accuracy

Introduction

A traditional machine student is created by gathering tests of information to address the whole populace. This informational index is typically partitioned into at least two datasets. Part of the dataset set is ordinarily used for building up the machine student, and the excess information is used for assessment. Regularly this informational collection is imbalanced; the data comprises just a minuscule minority of the word. Imbalanced machine students will, in general, perform ineffectively with the arrangement of misrepresentation discovery, network interruption, uncommon infection diagnosing, and so on. This is because of imbalanced examining while building up the machine student. During the testing stage, these unusual cases are concealed during the preparation stage and are generally misclassified. Leo Breiman, an analyst from the University of California at Berkeley, built up an AI calculation to improve assorted information utilizing random testing and characteristics determination. This task included the execution of Breiman's arbitrary timberland calculation into Weka. Weka is an information mining programming being developed by The University of Waikato. Numerous highlights of the rough woodland measure still can't seem to be carried out into this product [1].

Background

The irregular backwoods machine student is a meta-student, comprising numerous individual students (trees). The arbitrary backwoods utilize various odd tree groupings to votes on a general order for the given arrangement of info sources. Overall in every individual machine, student vote is given equivalent weight. In Breiman's later work, this calculation was adjusted to perform both un-weighted a lot casting a ballot. The woodland picks the individual arrangement that contains the most votes [2]. Figure 1. the following is a visual portrayal of the un-weighted arbitrary woods calculation.

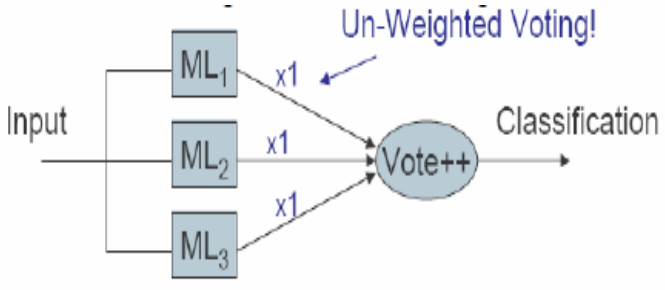

Figure 1. Meta Learners [3]

Singular random tree machine students are filled in an accompanying way:

1. An informational collection [inbag] is framed by inspecting with substitution individuals from the preparation set; this procedure is regularly alluded to as "bootstrapping." The quantity of models in the [inbag] informational collection is equivalent to that of the preparation informative collection. This new informative collection may contain copy models from the preparation set. Utilizing the bootstrapping procedure, typically, 33% of the trial set information is absent in the [inbag]. This avoided over information is known as the with regards to sack information [oob].

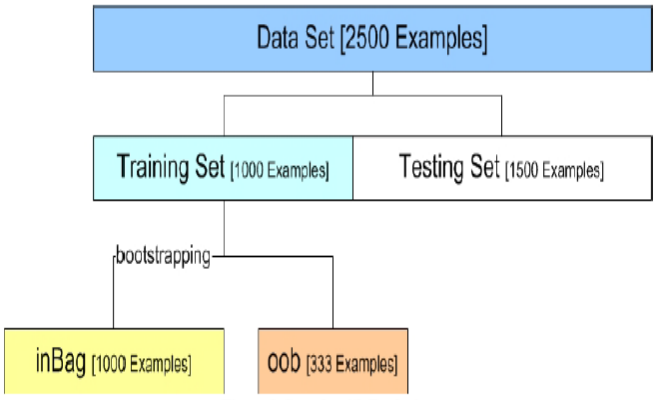

Figure 2. Sample with Replacing

2. An arbitrary number of traits are picked for each tree. These ascribe structure the hubs and leaves utilizing standard tree building calculations.
3. Each tree is developed to the furthest reaches conceivable without pruning.

This interaction is rehashed to build up numerous individual arbitrary trees students. After the tree's advancement, the out-of-sack models test the person's trees just as the whole timberland. The normal misclassification over all trees is known as the out-of-sack mistake gauge. This blunder gauge helps foresight the machine student's presentation without including the test set model. This data could be discovered to decide loads of the individual trees characterization in the weighted irregular backwoods student [3].

Variable Importance

A significant element of Breiman's calculation is the variable significance count. This calculation examines each property and uncovers the significance of the trait in foreseeing the arbitrary backwoods machine student's proper characterization. The client, at that point, could sift through superfluous ascribes, which would save time during information gathering and trial run time. This calculation initially processes immaculate proper check, the quantity of correct grouping utilizing the out-of-pack information as to its test set. The estimations of the ascribe are then haphazardly permuted in the out-of-pack models. This new informational collection is then tried for correct characterization. This number over all trees in the timberland is the crude significance score for the specific characteristic. This calculation is clarified in more subtleties in the execution area.

$$raw\_importance_{variable[m]} = \frac{untouched\_count - variable_m count}{number\_of\_trees}$$

Eqn. 1: raw importance calculation

A low crude significance score, a score almost zero, demonstrates a helpless connection between a given characteristic and suitable arrangement. A positive significance score indicates that the given variable is significant for proper order

Implementation of Variable Importance into Weka

The execution of the variable significance required the alteration of weka classifiers meta Bagging java and weka classifiers trees Random Forest java. Two new classes were made called Bagging Ext and Random Forest Ext. Singular trees are made with the call to the Weka classifiers meta. Bagging. Build Classifier (Instances information) technique, where Instances information is the preparation dataset. This technique constructs the individual trees and stores them in the item Classifiers [j], where j is the tree's record. This technique likewise holds the out-of-pack models in the variety of occurrences oobData [j]. Every individual tree has its arrangement of out-of-pack models, which can be utilized by record j [4].

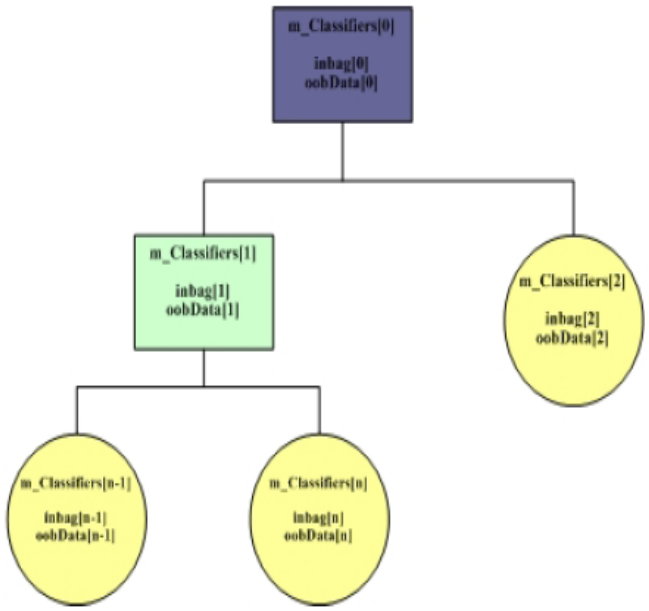

Figure 3: Tree Objects

The capacity of the out-of-pack models is a significant advance for figuring variable significance. Variable significance is acted in Weka.classifiers.meta.Bagging.calc Importance(). The initial phase in computation the variable significance is to get a right order check of the whole woods utilizing out-sack models as the test sets. This worth is put away in the variable correctSum. After calculating the correctSum, each character in the dataset is made with a permuted incentive for the given property. For instance: The traditional Weather. arff dataset has four info ascribes (viewpoint, temperature, moistness, and breezy), and one yields quality play. Hub access the timberland (m_Classifier[j]) has the out-of-pack dataset in Table1 segment 1. Viewpoint would have the outcome situated in segment 2 of Table1. This property is permuted all through all hubs in the whole woodland [5].

Table 1.

Original Out-of-bag Instance	Modified Out-of-Bag Instance
@data	@data
sunny,hot,high,FALSE,no	overcast,hot,high,FALSE,no
rainy,mild,high,FALSE,yes	sunny,mild,high,FALSE,yes
rainy,cool,normal,TRUE,no	sunny,cool,normal,TRUE,no
sunny,cool,normal,FALSE,yes	rainy,cool,normal,FALSE,yes
sunny,mild,normal,TRUE,yes	sunny,mild,normal,TRUE,yes

If the property is apparent, the worth is arbitrarily picked from the characterized choice of qualities in the arff document. If the character is numeric or genuine, the price is chosen utilizing the yield from the irregular number generator. This recently altered model is then executed down the tree for characterization. The altered out-of-pack right arrangement include is put away in the variable raw score. On the off chance that the variable is significant, the correct arrangement tally ought to fluctuate from the first check, correction. This method is rehashed for each trait in the preparation set.

Using Variable Importance in Weka

The recently added highlights of the Random Forest can be accessed by choosing the RandomForestExt Classifier from the Weka Explorer. The outcomes from the variable significance figuring are added to the classifier yield [6].

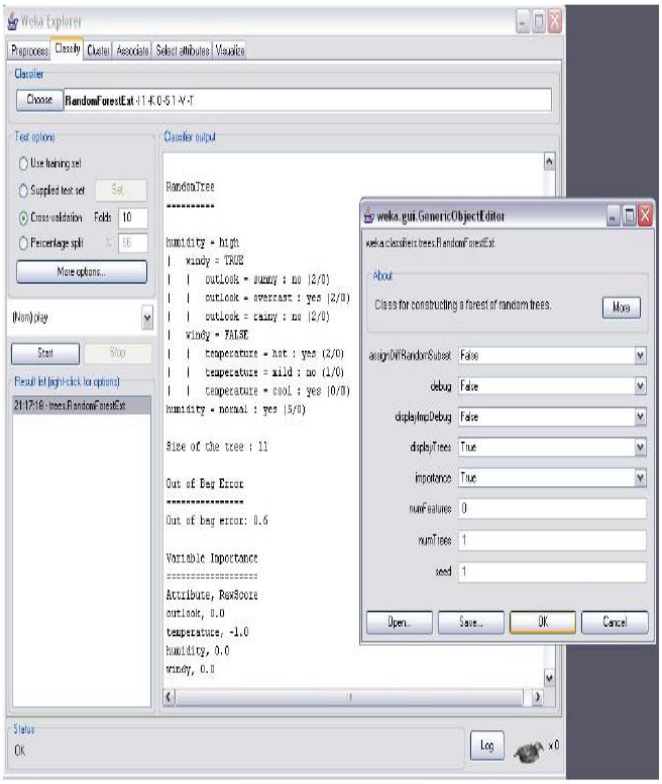

Figure 3. Weka Screenshot

Figure 3. shows the screen capture for the expanded rendition of the irregular timberland calculation. The all-encompassing variant is executed like the first form with a couple of alterations in the boundaries. The recently added boundaries comprise of displayImpDebug, display trees, and significance.

- displayImpDebug, Controls whether to show variable significance troubleshooting data. Setting this worth True will show data like the things in Table1. (Note: Increase required memory)

- display trees, Controls whether to show the individual trees

- significance, Controls whether to compute and show variable significant

Verification of Variable Importance

Approval of the variable significance execution was performed by looking at the aftereffects of the Weka's yield to the outcomes from Breiman's irregular backwoods paper of 2001. Breiman's article included two model counts of variable significance. In the primary model, varying importance was figured utilizing the Diabetes information test, with 1000 trees. Figure 4. shows the yield from Breiman's paper, and Figure 5. shows the output from Weka. There is comparable conduct in the

variable significance. In the two figures, the following property is essentially more imperative to the proper characterization [7].

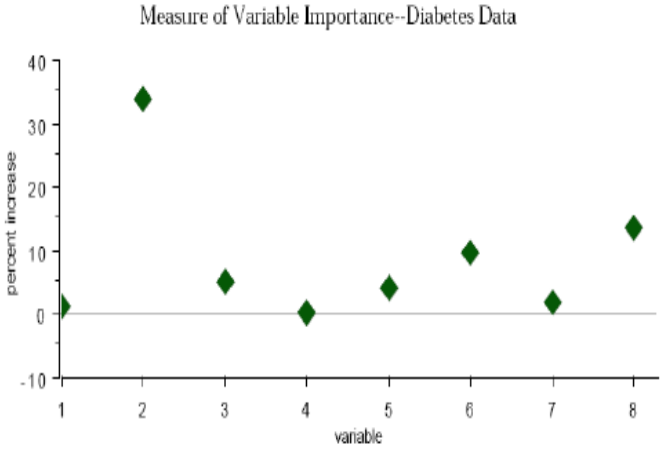

Figure 4. Breiman's Diabetes Output [1]

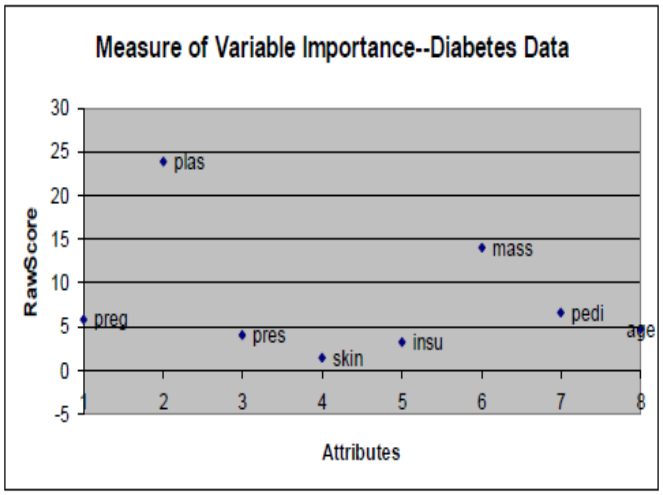

Figure 5. Weka's Diabetes Output

The analyses were continued utilizing the Votes dataset. Figure 6. shows Breiman's yield, and Figure 7. shows Weka's results for this investigation. Again the significance of the factors in both models has similar patterns [8].

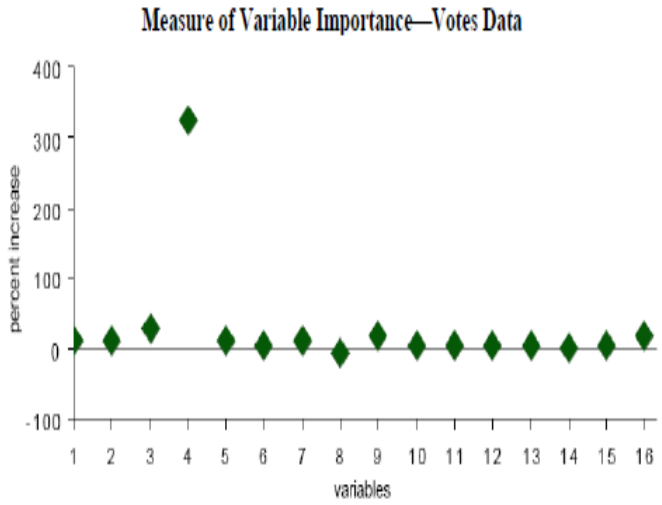

Figure 6. Breiman's Votes output [1]

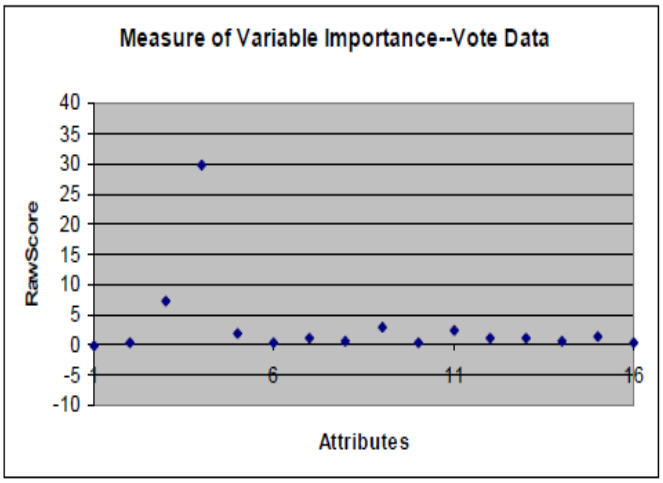

Figure 7. Weka's Votes output

Although Weka and Breiman's outcomes are the same, the product's approval now is relatively feeble. One potential issue is that the diagrams look at variable significance raw score (see Eqn 1.) to percent increment. Although there might be some caring a connection between the two, these correlations are not ideal. Another potential issue is the absence of check models.

Random Forest Software Improvements

To appropriately confirm the new Weka's execution, the classifier yields ought to be contrasted with Breiman's code's work. The total irregular backwoods calculation is carried out in open source Fortran 77 code. To execute his code, one should complex code the classifier boundaries like many trees, informational index names, etc. At that point, the client should assemble the code utilizing an outdated compiler and arrange the informational indexes to such an extent that it very well may be perused by the program. Another weakness to his code is that Fortran 77 performs static memory allotment, which means memory for the program is saved toward execution. This benefit for speed yet could cause issues with huge informational collections on PCs with limited irregular access-

memory. Fortran 77 likewise needs highlights, for example, order line contentions and pointers. To make Breiman's work easier to use, changes/add-on were explored [9].

To change over informational index from and to Weka's configuration (arff expansion) to Breiman's arrangement, a java application was created call arffConverter. This program can be executed physically with the accompanying order.

java –cp RandomForest.jar arffConverter *[input file] [output file] [mode]*

[input file], the absolute or relative path to the input file
[output file], the absolute or relative path to the where the new file will be stored
[mode], an integer that is either a one or zero.

If the mode is zero, the [input file] is changed over from Breiman's arrangement to Weka's document design. If the method is one, the [input file] is changed over from Weka's configuration to Breiman's document design.

Breiman's record design contains just mathematical information. Strings are given numbers names starting with the file of 1 and increased once for each string occurrences, and spaces supplant the commas. Table 2. shows an illustration of this transformation.

Notes: Breiman's code doesn't fall except for properties estimations of anything. An answer is to fudge all estimates of zero by some modest number (ex. 0 becomes 0.001). This fudging isn't executed in this product and should be done physically. Most datasets needn't bother with this change.

Weka's arff Format	Breiman's Format
sunny,85,85,FALSE,no	1 85 85 1 1
sunny,80,90,TRUE,no	1 80 90 2 1
overcast,83,86,FALSE,yes	2 83 86 1 2
rainy,70,96,FALSE,yes	3 70 96 1 2
rainy,68,80,FALSE,yes	3 68 80 1 2

Table 2. Sample Conversion

The following improvement includes adjustment of the Fortran 77 code. Rather than the client's hard-coding boundaries and afterward arranging the program, it is ideal to pass the program's borders. Fortran 77 has next to no capacity to play out this errand; since all exhibit sizes are static and are decided when accumulating. One arrangement is to instate the clusters to a highly enormous size. Albeit this is a legitimate arrangement, it is very memory concentrated. The best performance is to utilize dynamic exhibits where the cluster's size can be set during runtime. This component was made accessible beginning with Fortran 90. Fortran 90 is practically in reverse viable with Fortran 77 aside from a couple of minor language structure changes. Just as the adjustment from static to dynamic exhibits, these designing changes were made to Breiman's code and incorporated as random_foreset_win32.exe and random_forest_linux.bin. This executable peruses in the investigation from params.dat. Table 3. shows the design record for the Weather dataset. More insights regarding the design can be found in Breiman's Manual.

4 14 2 1 0 0 1	Line 1: Describe data
4 1 10 10 1 0 0 0 4351	Line 2: Set run parameters
0 0 0 0	Line 3: Set importance options
0 5	Line 4: Set proximity computations
0 0 0	Line 5: Set options based on proximities
-999 0 0	Line 6: Replace missing values
0	Line 7: Visualization
0 0 0 0	Line 8: Saving a forest
0 0 0 0	Line 9: Running new data down a saved forest
dataset.train dataset.test	Line 10: datasets files

Table 3. Configuration File

To execute this program, make a book document named params.dat containing the legitimate design information. Make dataset by hand or utilizing the AIFF converter. At that point, run the proper random_forest executable at the order brief. The yield is composed of the standard output. The program likewise made a few documents that save various boundaries. The program will produce a runtime blunder if the documents as of now exist. These documents should be erased before re-execution.

The RandomForest java application was created to mechanize the above interaction's entirety and give a GUI to the arbitrary backwoods calculation. Figure 8. shows a screen capture of the application.

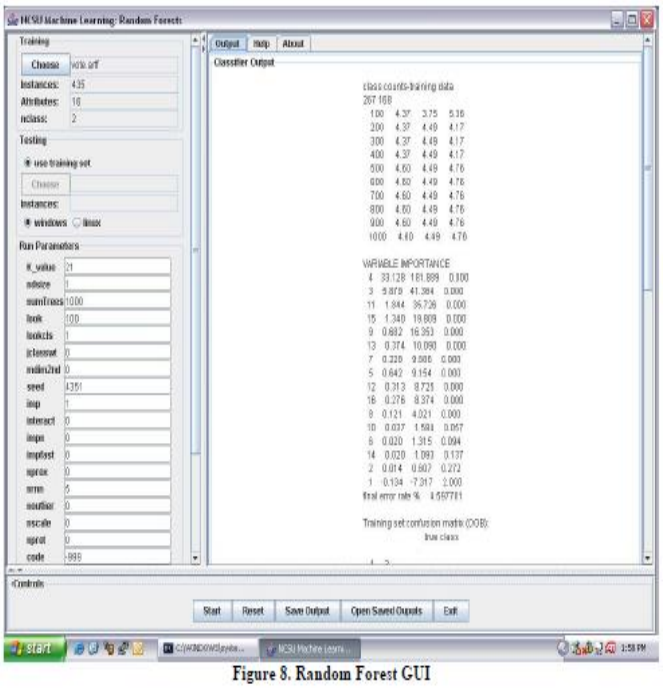

Figure 8. Random Forest GUI

The accompanying order is utilized to begin the application.

java -Xmx256m -jar RandomForest.jar

Little charms were made to RAFT, an irregular backwoods representation device, for simpler activity. The accompanying boundaries should be set in the arbitrary woodland application:

impn=1, imp=1, nscale=3 and nprox=1

To begin, RAFT utilizes the accompanying order:

java -mx300m -cp visad.jar;ij.jar;raft.jar;. Raft –norescale

Improved Verification

These new devices were, at that point, use to rehash the confirmation tests. Table 4. looks at the variable significance from Weka and Breiman. Albeit the crude score changes, the expectation of the two most significant ascribes is something very similar.

Attributes	Weka RawScore	Breiman RawScore
preg	5.877	2.103
plas	23.835	8.122
pres	4.037	0.286
skin	1.497	0.367
insu	3.182	0.457
mass	14.067	3.617
pedi	6.641	0.859
age	4.675	2.674
oob error%	30.69	23.697918

Table 4. Verification Diabetes Dataset

This investigation was rehashed for the Votes dataset. The outcome from this test confirms the correct execution of Weka's code. In Table 5. Weka's yield and Breiman's yield are practically the same.

Attributes	Weka's RawScore	Breiman's RawScore
handicapped-infants	-0.19	-0.134
water-project-cost-sharing	0.135	0.014
adoption-of-the-budget-resolution	7.281	5.879
physician-fee-freeze	29.706	33.128
el-salvador-aid	1.921	0.642
religious-groups-in-schools	0.146	0.02
anti-satellite-test-ban	1.011	0.228
aid-to-nicaraguan-contras	0.486	0.121
mx-missile	2.871	0.682
immigration	0.136	0.037
synfuels-corporation-cutback	2.232	1.844
education-spending	1.126	0.313
superfund-right-to-sue	0.902	0.374
crime	0.611	0.02
duty-free-exports	1.407	1.34
export-administration-act-south-africa	0.137	0.276
oob error%	6.52	4.5977

Table 5. Verification Votes Dataset

Understanding the Variable Importance

This segment inspects the variable significance results from the last part in more detail. Figure 9. shows a visual portrayal of the varying significance of the Diabetes dataset. The second and the six ascribes have higher importance than additional credits.

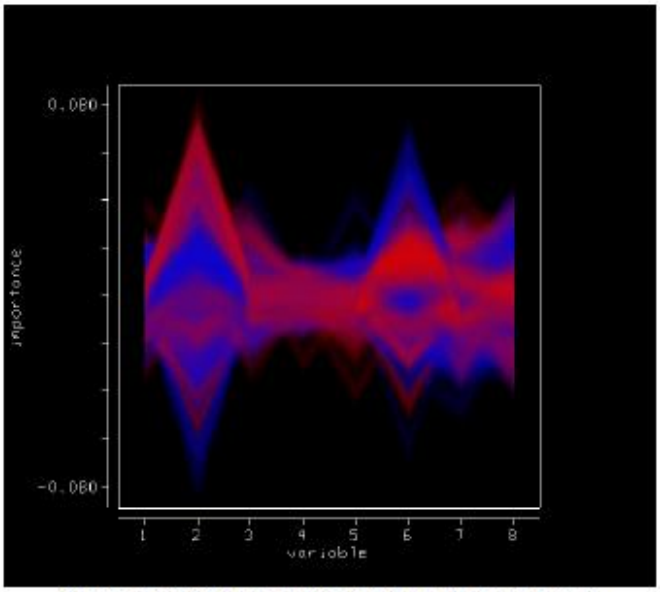

Figure 9. RAFT generated results from Diabetes dataset

By Utilizing 1000 trees, the out-of-pack mistake is 24.61%. I needed to perceive what the two most significant factors mean for the out-of-pack blunder [10-16]. The examination was re-run utilizing just the two most important factors, plans and mass. The mistaken characterization expanded somewhat to 27.99%. The analysis was re-run using any remaining credits aside from projects and assembly. In this examination, I was anticipating an enormous expansion in a wrong grouping. Albeit the blunder rates increment, it was a lot more modest than expected. These analyses were rehashed utilizing more miniature trees to check whether the patterns remain constants. The outcomes from these examinations are in Table 6. This data could help create future students for this specific dataset. One could accumulate and prepare the students utilizing just the plans and mass credits and get comparative outcomes with every one of the ascribes.

Trial	Correctly Classified Instances	Incorrectly Classified Instances
1000 Trees		
all attributes	75.39%	24.61%
plas,mass	72.01%	27.99%
all others	66.15%	33.85%
100 Trees		
all attributes	75.99%	24.09%
plas,mass	72.01%	27.99%
all others	66.28%	33.72%
10 Trees		
all attributes	73.96%	26.04%
plas,mass	71.48%	28.52%
all others	66.54%	33.46%

Table 6. Classification Error –Diabetes dataset

Conclusion

The underlying objective of this task was to carry out Breiman's arbitrary timberland calculation into Weka ultimately. The RandomForest java application permits full admittance to Breiman's analysis and is viable with Weka's datasets. The source code and the executables for this venture can be gotten at the connections recorded underneath.

Random Forest GUI

http://s112088960.onlinehome.us/ml2005/Zip%20of%20OTHER%20ournal%20Paper%20Files%20NOT%20JOURNAL%20PAPER/RandomForest.zip

Weka with Variable Importance Add-ons

http://s112088960.onlinehome.us/ml2005/Zip%20of%20OTHER%20Journal%20Paper%20Files%20NOT%20JOURNAL%20PAPER/Weka_Distro_FredLivingston.zip

RAFT Visualization Tools

http://s112088960.onlinehome.us/ml2005/Zip%20of%20OTHER%20Journal%20Paper%20Files%20NOT%20JOURNAL%20PAPER/RAFT_Distro_Livingston.zip

Future Work

The following is a rundown of potential assignments that could be discovered useful:

- Implementing significantly more highlights into Weka and checking those highlights.
- Modify code to manage non-ostensible classifier ascribes like reals and numerics

Interfacing RAFT into Weka.

References

[1] Pasham, S.D. (2017) AI-Driven Cloud Cost Optimization for Small and Medium Enterprises (SMEs). The Computertech. 1-24.

[2] Pasham, S.D. (2018) Dynamic Resource Provisioning in Cloud Environments Using Predictive Analytics. The Computertech. 1-28.

[3] Pasham, S.D. (2019) Energy-Efficient Task Scheduling in Distributed Edge Networks Using Reinforcement Learning. The Computertech. 1-23..

[4] Sai, K.M.V., M. Ramineni, M.V. Chowdary, and L. Deepthi. *Data Hiding Scheme in Quad Channel Images using Square Block Algorithm*. in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2018. IEEE..

[5] Alam, K., Mostakim, M. A., & Khan, M. S. I. (2017). Design and Optimization of MicroSolar Grid for Off-Grid Rural Communities. Distributed Learning and Broad Applications in Scientific Research, 3.

[6] Agarwal, A. V., & Kumar, S. (2017, November). Unsupervised data responsive based monitoring of fields. In 2017 International Conference on Inventive Computing and Informatics (ICICI) (pp. 184-188). IEEE.

[7] Agarwal, A. V., Verma, N., Saha, S., & Kumar, S. (2018). Dynamic Detection and Prevention of Denial of Service and Peer Attacks with IPAddress Processing. Recent Findings in Intelligent Computing Techniques: Proceedings of the 5th ICACNI 2017, Volume 1, 707, 139.

[8] Mishra, M. (2017). Reliability-based Life Cycle Management of Corroding Pipelines via Optimization under Uncertainty (Doctoral dissertation).

[9] Agarwal, A. V., Verma, N., & Kumar, S. (2018). Intelligent Decision Making Real-Time Automated System for Toll Payments. In Proceedings of International Conference on Recent Advancement on Computer and Communication: ICRAC 2017 (pp. 223-232). Springer Singapore.

[10] Agarwal, A. V., & Kumar, S. (2017, October). Intelligent multi-level mechanism of secure data handling of vehicular information for post-accident protocols. In 2017 2nd International Conference on Communication and Electronics Systems (ICCES) (pp. 902-906). IEEE.

[11] Malhotra, I., Gopinath, S., Janga, K. C., Greenberg, S., Sharma, S. K., & Tarkovsky, R. (2014). Unpredictable nature of tolvaptan in treatment of hypervolemic hyponatremia: case review on role of vaptans. Case reports in endocrinology, 2014(1), 807054.

[12] Shakibaie-M, B. (2013). Comparison of the effectiveness of two different bone substitute materials for socket preservation after tooth extraction: a controlled clinical study. International Journal of Periodontics & Restorative Dentistry, 33(2).

[13] Gopinath, S., Janga, K. C., Greenberg, S., & Sharma, S. K. (2013). Tolvaptan in the treatment of acute hyponatremia associated with acute kidney injury. Case reports in nephrology, 2013(1), 801575.

[14] Shilpa, Lalitha, Prakash, A., & Rao, S. (2009). BFHI in a tertiary care hospital: Does being Baby friendly affect lactation success?. The Indian Journal of Pediatrics, 76, 655-657.

[15] Singh, V. K., Mishra, A., Gupta, K. K., Misra, R., & Patel, M. L. (2015). Reduction of microalbuminuria in type-2 diabetes mellitus with angiotensin-converting enzyme inhibitor alone and with cilnidipine. Indian Journal of Nephrology, 25(6), 334-339.

[16] Gopinath, S., Giambarberi, L., Patil, S., & Chamberlain, R. S. (2016). Characteristics and survival of patients with eccrine carcinoma: a cohort study. Journal of the American Academy of Dermatology, 75(1), 215-217.