International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-5)

Research Article | **May 2015**

Study and Implementation of Random Forest Algorithm with WEKA Tool

Dr. N. Venkatesan
Associate Professor, Dept. of IT.,
Bharathiyar College of Engg. & Technology
Karikal, India

G. Priya
Assistant Professor, Dept. of CS & CA
Siga College of Mgnt & Computer Science
Villupuram, India

Abstract–

Random forest algorithm in data mining is discussed here with its depth explorations. The result.csv dataset used to implement this algorithm is a sample result dataset of a college, generated in real time through automation software. This research tells about the clear vision of Random forest algorithm with what it is, where we have to use this algorithm and its benefits then algorithm implementation with clear screen shot with explanations. It mainly focuses on the development, applications and the significance of Random forest algorithm.

Keywords— Data Mining; Weka tool; random forest algorithm; classification; Dataset

I. INTRODUCTION

Random Forest (RF) is widely used to be a powerful new approach to data exploration, data analysis and predictive modeling. Random Forest was proposed by Breiman (2001) father of CART at University of Californio, Berkley.The results of random forests constructed from results from individual decision trees out of set of decision trees which are learned independently from a subset of training data. Random Forest has its roots in CART, Learning ensembles, committees of experts and combining models. Random forest offer Data Visualization for High dimensional data (many columns), clustering, outlier and error detection.

A Random Forest is a collection of CART-like trees for growing, combination, testing and Post-processing. One is an ensemble of trees where each tree is growing while training on a sample obtained from the raining set via bagging *without* replacement. This is a known technique from ensemble learning methodology where generalization error is decreased due to combining decisions (or so-called votes) of multiple learners which are usually weak and unstable individually. The second approach is random split selection for a decision tree. This split is chosen randomly from a subset of best splits. Thus, these two ideas led finally to the basis for Random forest. It generally applies two mechanisms: building an ensemble of trees via bagging *with* replacement (bootstrap) and a random selection of features at each tree node. The first one means that any example selected from the training set can be selected again. Each tree is grown using the obtained bootstrap sample. The second mechanism performs random selecting a small fraction of features and further splitting using the best feature from this set. The size of a fraction (i.e. the number of features to select) is fixed within the algorithm execution.

II. RANDOM FOREST CLASSIFIER

A. Variable Importance in RF

Random forests algorithm used to rank the importance of variables in a classification or regression problem in a natural way. The following technique was described in Breiman's paper, is implemented in the R package randomForest. The first step in measuring the variable importance in a data set is to fit a random forest to the data. During the fitting process the out-of-bag error for each data point is recorded and averaged over the forest.To measure the importance of the f-th feature after training, the values of the f-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the f-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences. Features which produce large values for this score are ranked as more important than features which produce small values.

B. Proximity in RF

These are one of the most useful tools in random forests. The proximities originally formed a “MxM” matrix. After a tree is grown, put all of the data, both training and out of bag, down the tree. If cases k and n are in the same terminal node increase their proximity by one. At the end, normalize the proximities by dividing by the number of trees. Users noted that with large data sets, they could not fit an “MxM” matrix into fast memory. A modification reduced the required memory size to MxT where T is the number of trees in the forest. To speed up the computation-intensive scaling and iterative missing value replacement, the user is given the option of retaining only the “MxM” largest proximities to each case. When a test set is present, the proximities of each case in the test set with each case in the training set can also be computed.

C. OOB (Out of Bag Error) in RF

In random forests, no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, as follows: Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree. Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the out-of-bag error estimate. This has proven to be unbiased in many tests.

D. Features of RF

- It is in accuracy among current algorithms.
- It runs effectively on large data bases.
- It can handle many numbers of input variables without variable deletion.
- It gives estimates of what important in variables in the classification.
- It generates an internal estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use of the data .
- Prototypes are computed, that gives information about the relation between variables and classification.
- The capabilities can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable importance.

RF algorithm solves described challenge by minimizing correlation while maintaining strength. It is achieved by injecting the randomness into the training process. Particularly, random selection of features results in diverse learners which are still individually strong due to splitting using the best feature from the random fraction. Another property increasing diversity is that trees are not pruned during growing. Instead, growing is stopped when a leaf size limit is reached.

In fact, RF algorithm has only two main parameters affecting its performance considerably. The number of trees nt in an ensemble to grow and the number of features f to select randomly at each split. Moreover, it is naturally suitable for multiclass classification challenges because it consists of tree classifiers. Unlike many classifiers cause growing number of parameters to tune while combining them as multiple learners in an ensemble in order to deal with multicast classification, RF algorithm retains only two mentioned parameters mainly to consider. The RF algorithm is presented below.

Algorithm 1 Random Forest

Input: dataset *Tree = (a,b)*, number of trees nt, number of random features f
Output: *RF*, a set of grown trees
Initialize *RF*
for *i* = 1 **to** *nt* **do**
    Tree' ← *bootstrap(Tree)*
    Tree ← *trainDT(Tree', f)*
    add *Tree* to *RF*
end for

Algorithm 1 shows the technique of building an ensemble of decision trees using bagging. The function *trainDT(Tree',f)* performs training of a decision tree on a bootstrap sample *Tree'* selecting *f* features randomly at each split. The process of training a decision tree needs clarification and described in Algorithm 2.

Algorithm 2 Decision Tree

Input: a sample *Tree = (a,b)*, number of random features f, a leaf size limit *lsize*
Output: *Tree*, a trained decision tree Initialize *Tree, fnum* as a total number of features, *tnum* as a predefined number of thresholds
function *trainDT(Tree, f)*
if *sizeof(Tree)* <= *lsize* **then**
    label ← *indexof(max* in *histogram(Tree))*
else
    fraction ← *random(f* from *fnum)*
    thresholds ← *random(tnum)*
    (f, t) ← *max* of split function in *fraction* and *thresholds*
    (leftT, rightT) ← *split(T, f , t)*

add left child ← *trainDT(leftT, f)*
add right child ← *trainDT(rightT, f)*
end if
end function

III. IMPLEMENTATION

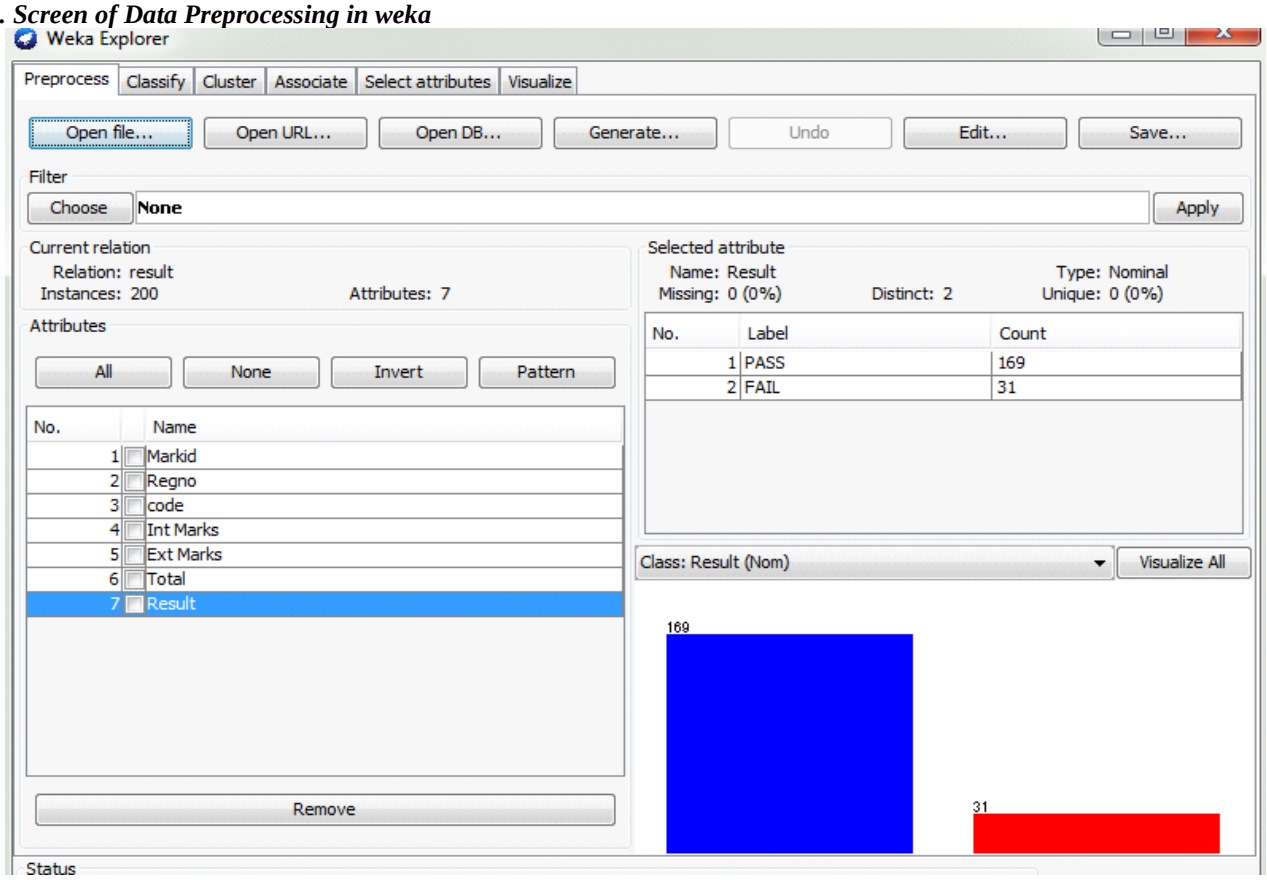
This algorithm is implemented through weka with the following screen shot and its output. Weka was developed at the University of Waikato in New Zealand; the name stands for *Waikato Environment for Knowledge Analysis*. It provides a uniform interface to many different learning algorithms, along with methods for pre- and postprocessing and for evaluating the result of learning schemes on any given dataset.

A. Dataset Result.csv

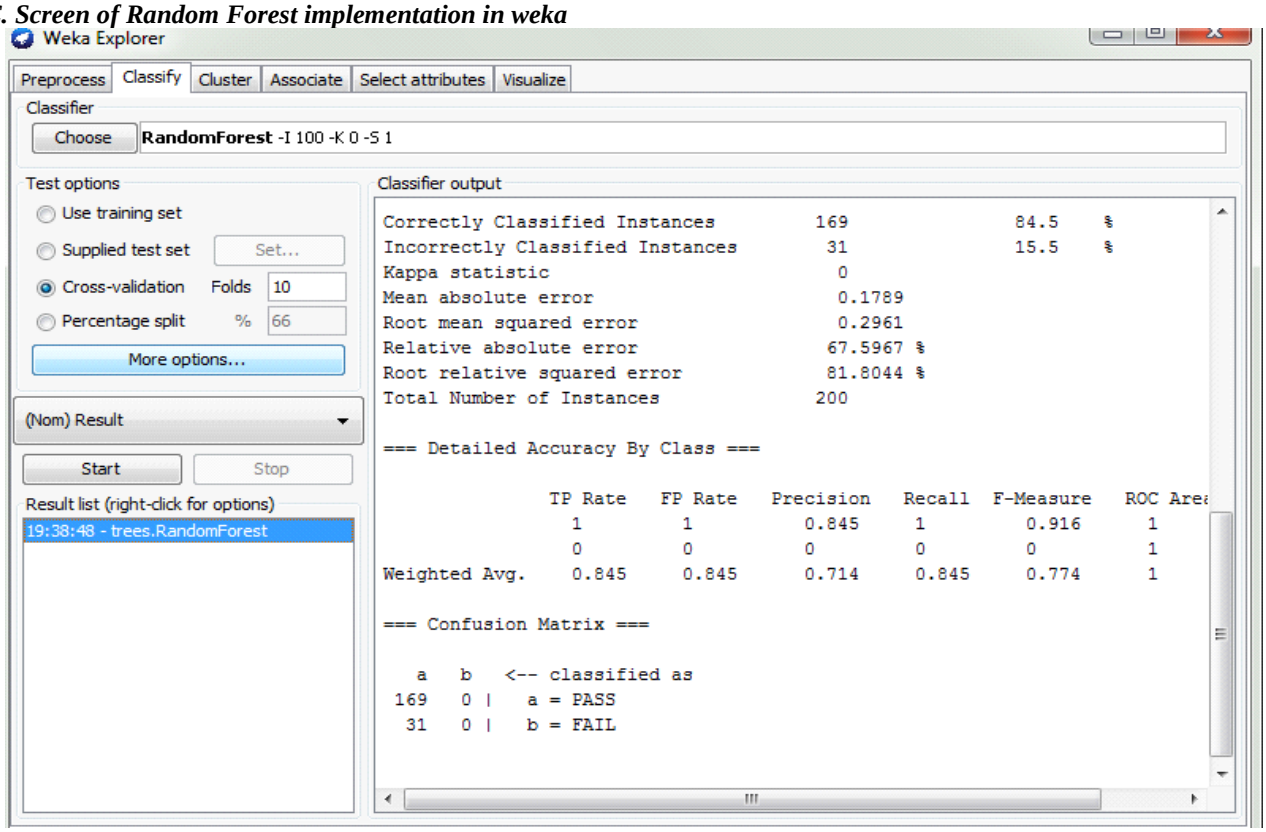
Markid	Regno	code	Int Marks	Ext Marks	Total	Result
M1	10UCCA006	UCCM601	24	51	75	PASS
M2	10UCCA006	UCCM602	25	39	64	PASS
M3	10UCCA006	UCCM603	25	43	68	PASS
M4	10UCCA006	UCCM604	25	53	78	PASS
M5	10UCCA006	UCCM606	21	49	70	PASS
M6	10UCCA006	UCCM607	-	88	88	PASS
M7	10UCCA006	UCCO601	24	46	70	PASS
M8	10UCCA006	UCCR605	60	39	99	PASS
M9	10UCCA006	USKS601	84	-	84	PASS
M10	10UCCA021	UCCM601	20	43	63	PASS
M11	10UCCA021	UCCM602	19	37	56	PASS
M12	10UCCA021	UCCM603	19	34	53	PASS
M13	10UCCA021	UCCM604	20	36	56	PASS
M14	10UCCA021	UCCM606	17	39	56	PASS
M15	10UCCA021	UCCM607	-	76	76	PASS
M16	10UCCA021	UCCO601	18	37	55	PASS
M17	10UCCA021	UCCR605	60	38	98	PASS
M18	10UCCA021	USKS601	77	-	77	PASS
M19	12UMAT046	UMAM402	15	13	28	FAIL
M20	12UMAT058	UMAM402	15	14	29	FAIL
M21	12UMAT071	UPHA402	18	23	41	FAIL
M22	12UPHY014	UCHA401	19	8	27	FAIL
M23	12UPHY015	UCHA401	19	6	25	FAIL
M24	12UPHY021	UCHA401	20	14	34	FAIL
M25	12UTAM001	UBCE403	15	24	39	FAIL
M26	12UTAM003	usks401	29	-	29	FAIL
M27	12UTAM012	UBCE403	13	18	31	FAIL
M28	12UTAM013	UBCE403	12	11	23	FAIL
M29	12UTAM013	UTAM401	22	17	39	FAIL

Total No of Instances taken : 200
Pass : 169
Fail : 31
Pass or Fail is determined by Int Marks and Ext Marks attributes.

B. Screen of Data Preprocessing in weka



C. Screen of Random Forest implementation in weka



Classifier Output

=== Run information ===

Scheme:weka.classifiers.trees.RandomForest -I 100 -K 0 -S 1
Relation: result

Instances: 200
Attributes: 7
Markid
Regno
code
Int Marks
Ext Marks
Total
Result
Test mode:10-fold cross-validation

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

Random forest of 100 trees, each constructed while considering 3 random features.
Out of bag error: 0.15
Time taken to build model: 0.13 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	169	84.5 %
Incorrectly Classified Instances	31	15.5 %
Kappa statistic	0	
Mean absolute error	0.1789	
Root mean squared error	0.2961	
Relative absolute error	67.5967 %	
Root relative squared error	81.8044 %	
Total Number of Instances	200	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	1	0.845	1	0.916	1	PASS
	0	0	0	0	0	1	FAIL
Weighted Avg.	0.845	0.845	0.714	0.845	0.774	1	

=== Confusion Matrix ===

a	b	<-- classified as
169	0	a = PASS
31	0	b = FAIL

IV. CONCLUSION AND FUTURE WORK

This paper was implemented with an ordinary small sample dataset with 200 instances. This would be a very small work of a research. This work is extended to World Wide Web dataset for the implementation of web mining. Web spam detection is a very big problem of the today's internet world. This analysis work will be extended to the application of content spam features.

ACKNOWLEDGMENT

This work was supported by my Ph.D. guide and colleague friends who gave me best support to fulfill this paper successfully.

REFERENCES

[1] L. Breiman, L. Random Forests, Machine Learning, vol. 45 pp. 5–32, 2001.
[2] Robnik-Sikonja, M. Improving to Machine Learning, MIT Press, Cambridge, Massachusetts, 2nd Edition
[3] Witten, Ian and Frank, Eibe. Data Mining, Practical Machine Learning Tools and Techniques
[4] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations, Volume 11 Issue 1 (2009).
[5] R. Nicole, "Title of paper with only first word capitalized," J. Name For. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

BIOGRAPHIES

Dr. N.Venkatesan is working as an Associate Professor, Information Technology Department, Bharathiyar college of Engineering and Technology, Karaikal. He completed Ph.D in Computer Science at MK University.. He has been member of ISTE. He has 18 Years of teaching and 2 years of research experience. He published 50 papers in National and International journals and conferences. He has authored a book ***Data Mining and Warehousing***. His areas of interest are Data Mining and Network Security. He is the editorial board member of several international journals and conferences.

Mrs. G. Priya, Completed her M.C.A., M.Phil., degree in Madurai Kamaraj University, Madurai. She has 7 years of Teaching experience. She is at present working as Assistant Professor, Dept. of CS & CA in Siga College of Management and Computer Science, Villupuram. Her research area is Web mining, Data Structures and Algorithms.

