Paper 242-2017

Random Forests with Approximate Bayesian Model Averaging

Tiny du Toit, North-West University, South Africa; André de Waal, SAS Institute Inc.

ABSTRACT

A random forest is an ensemble of decision trees that often produce more accurate results than a single decision tree. The predictions of the individual trees in the forest are averaged to produce a final prediction. The question now arises whether a better or more accurate final prediction cannot be obtained by a more intelligent use of the trees in the forest. In particular, in the way random forests are currently defined, every tree contributes the same fraction to the final result (e.g. if there are 50 trees, each tree contributes 1/50th to the final result). This ignores model uncertainty as less accurate trees are treated exactly like more accurate trees. Replacing averaging with Bayesian Model Averaging will give better trees the opportunity to contribute more to the final result which may lead to more accurate predictions. However, there are several complications to this approach that have to be resolved, such as the computation of a SBC value for a decision tree. Two novel approaches to solving this problem are presented and the results compared to that obtained with the standard random forest approach.

INTRODUCTION

Random forests (Breiman, 2001; Breiman, 2001b) occupies a leading position amongst ensemble models and have shown to be very successful in data mining and analytics competitions such as KDD Cup (Lichman, 2013) and Kaggle (2016). One of the reasons for its success is that each tree in the forest provides part of the solution which, in the aggregate, produces more accurate results than a single tree.

In the bagging and random forest approaches, multiple decision trees are generated and their predictions are combined into a single prediction. For random forests, the predictions of the individual trees are averaged to obtain a final prediction. All trees are treated equally and each tree makes exactly the same contribution to the final prediction. In this paper we question the supposition as model uncertainty is ignored.

Random forests are successful because the approach is based on the idea that the underlying trees should be different (if the trees were equal only one tree would be needed to represent the forest). This tree mixture is achieved by injecting randomness into the trees (this is explained in more detail in the following section). The resulting trees are diverse (by design) with varying levels of predictive power.

A goodness-of-fit statistic such as misclassification rate or average squared error may be used to judge the quality of each tree. Should the more predictive/accurate trees not carry more weight towards the final prediction? If the answer is affirmative, a second question needs to be answered: how should the trees be aggregated/amalgamated to get the best result?

In the rest of the paper a method of intelligent tree combination/aggregation, based on the theory of Bayesian Model Averaging, is proposed. Forests in SAS® Enterprise Miner is described in Section 2. The theory of Bayesian Model Averaging is explained in Section 3. For the theory of Bayesian Model Averaging to be applicable to decision trees, each tree's SBC value needs to be approximated. Neural networks are used to approximate the decision trees and this is explained in Section 4. A new weighting scheme is introduced in Section 5. Directly computing the degrees of freedom of a tree is reviewed in Section 6. The paper ends with a discussion and conclusions.

FORESTS IN SAS ENTERPRISE MINER

A random forest is an ensemble of decision trees. Multiple decision trees are constructed, each tree based on a random sample of observations from the training data set. The trees are then combined into a final model. For an interval target, the predictions of the individual trees in the forest are averaged. For a categorical target, the posterior probabilities are also averaged over the trees. A second step usually involves some kind of majority voting to predict the target category.

In SAS Enterprise Miner, the HPFOREST procedure (De Ville and Neville, 2013) takes a random sample (without replacement) of the observations in the training data set. This is done for each tree in the forest.

When each node in a tree is constructed, a subset of inputs is selected at random from the set of available inputs. Only the variable with the highest correlation with the target is then selected from this subset and used to split the node. Because many decision trees are grown, the expectation is that the better variables are more likely to be selected and that random errors introduced from overfitting will cancel when the predictions are averaged.

Our first attempt at Bayesian Model Averaging centered on the use of the HP Forest node in SAS Enterprise Miner (see Figure 1).

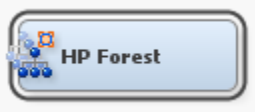

Figure 1. HP Forest node

However, because the node is “locked down”, the user is unable to get access to the individual trees in the forest. Furthermore, the bagged sample (training data set) as well as the out-of-bag sample (testing data set) constructed by the HP Forest node are inaccessible. It is therefore impossible to use the output of one HP Forest node (in its current state) to implement our new approach. After some experimentation we decided on a different strategy.

The data set that is analyzed in this paper is the HMEQ data set. The data set contains 13 variables with loan default status (BAD) as the dependent variable and 12 independent variables, e.g. years at current job (YOJ), number of derogatory reports (Derogatories), number of delinquent trade lines (Delinquencies), etc. The data partition node was used to partition the raw data set into a training data set containing 80% of the data and a validation data set containing 20% of the data (see Figure 2). As the HMEQ data set is relatively small (only 5960 observations), and the training data set is again going to be divided into bagged and out-of-bag samples, the training data set was kept as large as possible without compromising the various model building steps that will follow.

Figure 2. Data partitioning of the raw data set

N different trees are constructed using *N* Decision Tree nodes. The trees are then aggregated as needed. But, the trees have to be different (as would have been the case if the HP Forest node was used). The solution is to use the HP Forest node for variable selection.

N HP Forest nodes (with different seeds) are used to construct *N* different forests, each containing a single tree. As each forest is built using different bagged (0.8) and out-of-bag samples (0.2), the trees in the forests should be very different from each other. Also to restrict the number of variables, the maximum depth of the trees in the forest has been set to three.

Although *N* trees (one from each forest) have been built, the details of the trees are hidden and access to the bagged and out-of-bag samples that were used to construct the trees are unavailable. But, information on the subsets of variables used to construct the forests (and therefore the trees) is accessible. These subsets are now used to construct *N* trees using Decision Tree nodes (see Figure 3). This strategy will force the *N* trees to be different.

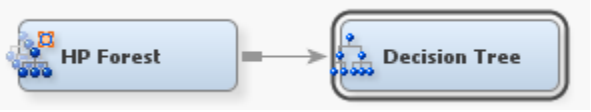

Figure 3. HP Forest node used for variable selection

The HP Forest nodes are basically used as variable selection nodes so that the *N* decision trees that are constructed will be different from each other (simulating the strategy used by the HP Forest node). The trees will most probably not be exactly the same as that constructed by the HP Forest node (because the

order of the splits are unknown and not all variables in the subset are available at each split), but the trees are built on the same subsets of variables used in the forests. The result is *N* different trees that can now be used to compare the different weighting schemes.

It might be tempting to use the *N* trees in the *N* forests to implement our new weighting scheme on. The problem is that each tree was built on a different bagged sample and also has an associated out-of-bag sample. Because the samples are not known, it is impossible to compute the goodness-of-fit statistics for the bagged or out-of-bag samples. The best that can be done is to consider the union of the different bagged and out-of-bag samples, which is the training data set.

As the HMEQ data set is small, partitioning the raw data set into three data sets to obtain a test data set is not practical (this would have been ideal to obtain an independent estimate of the performance of the models). The scoring data set therefore consists of the union of the training and the validation data sets, thus the raw hmeq data set (see Figure 4).

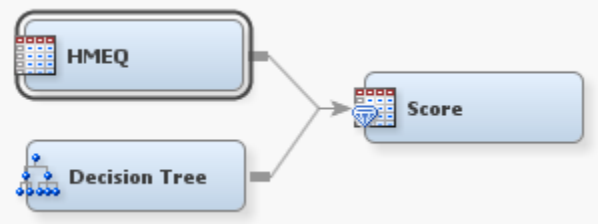

Figure 4. Scoring with a decision tree

This is a compromise and the fit statistics may therefore be optimistic. As only the relative performance of the models are important, all models are treated equally by scoring this data set.

The standard averaging implemented by the HP Forest node is now coded in a SAS Code node (see Figure 5) and the results computed on the scoring (hmeq) data set.

Figure 5. Computing goodness-of-fit measures

In this example, *N=5* trees are constructed. Details of the number of leaves in each tree, the number of variables used in splitting and the depth of each tree are given in Table 1.

Tree	#Leaves	#Variables	Depth
1	10	3	5
2	5	2	4
3	16	4	6
4	14	4	6
5	16	4	6

Table 1. Five Trees

The *c*-statistic for the random forest based on these 5 trees is 88.3015, the misclassification rate is 14.89% and the sum of squared errors (sse) is 646.50. This is our *baseline* model and we will demonstrate in the following sections that a more intelligent amalgamation of the trees in the forest could result in a much better model with higher *c*-statistic, lower misclassification rate and smaller sum of squared errors (an indication of the variance of the errors).

BAYESIAN MODEL AVERAGING

When a single model is selected for predictive modeling, uncertainty about the structure of the model and the variables that must be included are ignored. This leads to uncertainty about the quantities of interest being underestimated (Madigan and Raftery, 1994). Regal and Hook (1991) and Miller (1984) showed in the contexts of regression and contingency tables that this underestimation can be large which can lead to decisions that have too high risk (Hodges, 1987).

In principle, the standard Bayesian formalism (Learner, 1978) provides a universal solution to all these difficulties. Let Δ be the quantity of interest, such as a future observation, a parameter or the utility of a course of action. Given data *D*, the posterior distribution of Δ is

$pr(\Delta|D) = \sum_{k=1}^{K} pr(\Delta|M_k, D)pr(M_k|D)$ (3.1).

The latter expression is the mean of the posterior distributions under each of the models, weighted by their posterior model probabilities. The models that are considered are $M_1, M_2, \dots, M_k$ and

$pr(M_k|D) = \frac{pr(D|M_k)pr(M_k)}{\sum_{l=1}^{K} pr(D|M_l)pr(M_l)}$ (3.2)

where

$pr(D|M_k) = \int pr(D|\theta_k, M_k)pr(\theta_k|M_k)d\theta_k$ (3.3)

is the marginal likelihood of model M_k, θ_k is the parameter vector of M_k, $pr(M_k)$ is the prior probability of M_k, $pr(D|\theta_k, M_k)$ is the likelihood, and $pr(\theta_k|M_k)$ is the prior distrubution of θ_k.

When averaging over all the models, a better predictive ability is obtained compared to using any single model M_j, as measured by a logarithmic scoring rule:

$-E[\log\{\sum_{k=1}^{K} pr(\Delta|M_k, D)pr(M_k|D)\}] \leq -E\left[log\{pr(\Delta|M_j, D)\}\right]$ $(j = 1,2, \dots, K)$

where Δ is the observable to be predicted and the expectation is with respect to

$$\sum_{k=1}^{K} pr(\Delta|M_k)pr(M_k|D).$$

This latter result follows from the nonnegativity of the Kullback-Leibler information divergence.

In practice, the Bayesian model averaging (BMA) approach in general has not been adapted due to a number of challenges involved (Hoeting, Madigan, Raftery and Volinsky, 1999). Firstly, the posterior model probabilities $pr(M_k|D)$ involve the very high dimensional integrals in (3.3) which typically do not exist in closed form. This makes the probabilities hard to compute. Secondly, as the number of models in the sum of (3.1) can be very large, exhaustive summation is rendered infeasible. Thirdly, as it is challenging, little attention has been given to the specification of $pr(M_k)$, the prior distribution over competing models. The problem of managing the summation in (3.1) for a large number of models has been investigated by a number of researchers. Hoeting (n.d.) discussed the historical developments of BMA, provided an additional description of the challenges of carrying out BMA, and considered solutions to these problems for a number of model classes. More recent research in this area are described by Hoeting (n.d).

Lee (1999) and Lee (2006) considered a number of methods for estimating the integral of (3.3) and came to the conclusion that the SBC may be the most reliable way of estimating this quantity. The SBC defined as

$SBC_i = \log\left(\mathcal{L}(\hat{\theta}|y)\right) - K_i \log(n)$

is used. In addition, a noninformative prior is exploited that puts equal mass on each model, i.e. $P(M_i) = P(M_j)$ for all i and j. The SBC approximation to (3.2) then becomes

$pr(M_i|D) \approx \frac{P(D|M_i)}{\sum_j P(D|M_j)} \approx \frac{e^{SBC_i}}{\sum_j e^{SBC_j}}$ (3.4).

The Bayesian approach automatically manages the balance between improving fit and not overfitting, as additional variables that do not sufficiently improve the fit will dilute the posterior, resulting in a lower posterior probability for the model. This approach is conceptually straightforward and has the advantage of being used simultaneously on both the problem of choosing a subset of explanatory variables, and the problem of choosing the architecture for the neural network (Du Toit, 2006).

When the SBC defined as

$$SBC_i = -2\log\left(\mathcal{L}\left(\hat{\theta}\middle|y\right)\right) + K_i\log(n)$$

is used, (3.4) becomes

$$pr(M_i|D) \approx \frac{P(D|M_i)}{\sum_j P\left(D\middle|M_j\right)} \approx \frac{e^{-SBC_i}}{\sum_j e^{-SBC_j}}.$$

APPROXIMATING SBC WITH A NEURAL NETWORK

It is well-known that a decision tree can be used to approximate a neural network. This is usually done to gain some understanding of the neural network, as a neural network can be a black box. The converse, which is using a neural network to approximate a decision tree, is less obvious.

Just as with surrogate models (a surrogate model approximates an inscrutable model's predictions/decisions in order to facilitate interpretation), a neural network may be used to approximate the decision boundaries of the decision tree. As a neural network retains any non-linear relationships that are present in the decision tree, it is a good candidate for approximating a decision tree.

To apply the Bayesian Model Averaging formula of Section 3 to the trees in a forest, each tree's SBC is needed. This is not available as the degrees of freedom *K* for a decision tree is in general undefined. It is furthermore known that objective model selection criteria such as SBC cannot be used to compare models across different modeling techniques. It can only be used as a relative measure ranking models based on the same modeling technique. The expectation now is that the ranking of the decision trees from better to worse will be preserved in the SBC values computed by the neural networks.

Assume there are *N* decision trees in the forest. *N* neural networks are now constructed: each neural network approximating one tree. The number of hidden nodes in each neural network is adjusted to produce a neural network that closely shadows (in ROC curve and misclassification rate) the relevant tree (see Figure 6). As SBC is defined for neural networks in SAS Enterprise Miner, the neural network models' SBCs are now used as proxies for the decision trees' SBCs. Note also that SBC is only computed for the training data set by the Neural Network node, so this is what is used.

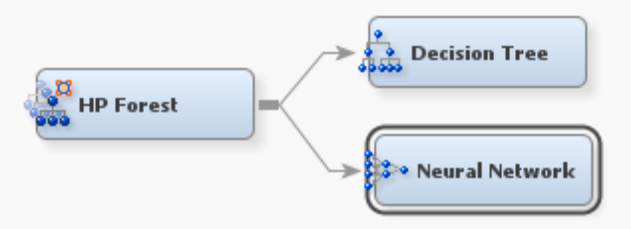

Figure 6. Approximating a decision tree with a neural network

The Neural Network node should be connected in parallel to the Decision Tree node and should have all the variables selected by the HP Forest node as inputs. If the Neural Network node is connected to the Decision Tree node, the Decision Tree node could do additional variable selection which may be undesirable. The training data set is used to construct the decision trees and the neural networks and the validation data set is used to optimize the decision trees as well as for stopped training in the neural networks.

Details of the *N=5* constructed neural networks sorted by SBC are given in Table 2. All neural networks are multilayer perceptrons with one hidden layer and *M* hidden nodes.

Rank	Tree	# Hidden Nodes	SBC
1	2	4	3322
2	5	2	4032
3	3	4	4061
4	4	5	4393
5	1	5	4400

Table 2. MLP architectures

As the SBC values computed by the neural networks for the training data set used in this paper are large (3322 and greater), the base (*e*) used in the Bayesian Model Averaging formula, e.g.

$$\frac{e^{-3322}}{e^{-3322}+e^{-4032}+e^{-4061}+e^{-4393}+e^{-4400}}$$

creates computational difficulties and needs to be adjusted to make the computations viable.

When the base *e* is replaced by base 1, we get averaging:

$$\frac{1^{-3322}}{1^{-3322}+1^{-4032}+1^{-4061}+1^{-4393}+1^{-4400}} = \frac{1}{5}$$

as implemented in the HP Forest node in SAS Enterprise Miner (although SAS Enterprise Miner most definitely did not use the above formula to arrive at 1/5).

We therefore need a base greater than 1, but smaller than *e* to make the computations feasible. In this example, the base is adjusted to 1.002 (some experimentation might be needed to find and to adjust the base used in the formula so that reasonable weights are produced). The final weight computed for the best model is:

$$\frac{1.002^{-3322}}{1.002^{-3322}+1.002^{-4032}+\cdots+1.002^{-4400}} = 0.5867$$

Table 3 ranks the five models from good to bad giving their SBC values (smaller is better) as well as final weight contribution to the forest.

Rank	SBC	Weight
1	3322	0.5867
2	4032	0.1420
3	4061	0.1340
4	4393	0.0690
5	4400	0.0680

Table 3. Bayesian Model Averaging

The *c*-statistic for this model is 89.3474, the misclassification rate is 13.37% and the sse is 565.61. This gives an improvement of more than 1.18% in the *c*-statistic over the standard method used to construct the forest. The misclassification rate is reduced by 10.2% and the sse decreased by 12.5%. Because we do not have the degrees of freedom for the decision tree, we cannot compute the error variance (as is usually done for linear regression), but sse gives a good indication that the size of the errors decreased (the computed probabilities are more precise).

It is worth noting that the SBC is only used to weigh the contributions of each tree and that the underlying trees in the forest are not modified at all. The trees are only amalgamated in a more intelligent way using the computed weights.

A NEW WEIGHTING SCHEME

Approximating a decision tree with a neural network has its drawbacks: extra computation time is needed to train and adjust the neural network and the approximation may be imprecise. Finding the appropriate base to get a reasonable spread of the final weight might be an issue.

As we are only interested in the relative ordering of the models (from good to bad), sse or validation misclassification rate might also suffice. Table 4 lists the SBC and sse on the training data set for the neural networks as well as the sse on the training data set for the decision trees and the validation misclassification rate also for the decision trees.

Rank	TRAIN SBC NN	TRAIN SSE NN	TRAIN SSE DT	VALID MISC DT
1	3322	992	952	12.98
2	4032	1214	1168	15.74
3	4061	1185	1168	15.74
4	4393	1282	1280	16.41
5	4400	1296	1313	16.75

Table 4. SBC, SSE and MISC

Note how closely the neural network sse approximates the decision tree sse. Except for the tie in the 2nd and 3rd models, the decision tree misclassification rate on the validation data set mimics the ordering of SBC computed with the neural network. A simplification of the whole process is therefore to use the validation misclassification rate computed for each decision tree to rank the models.

But, the formula used to compute the final weights depends on SBC and this is now missing if we omit constructing the neural networks. The following weighting scheme can be used as an approximation to the formula of the previous section, and this only depends on the ordering of the models (as given in Table 4), not the absolute values of the computed SBCs.

If there are *N* trees in the forest, the weight for each tree *i (i = 1, 2, …, N)* should be:

$$\left\{ \frac{2^{i-1}}{\sum_{k=0}^{n-1} 2^k} \right\}$$

For the hmeq data set with five trees in the forest, the weights are

$$\left\{ \frac{1}{31}, \frac{2}{31}, \frac{4}{31}, \frac{8}{31}, \frac{16}{31} \right\}$$

which seems reasonable and not that different from the weight computed with Bayesian Model Averaging. The table in the previous section is therefore updated to (see Table 5):

Rank	VALID MISC DT	Weight
1	12.98	0.5161
2	15.74	0.2580
3	15.74	0.1290
4	16.41	0.0645
5	16.75	0.0322

Table 5. Weights based on VALID MISC of DT

Note how closely these weights resemble the weights computed with SBC (see Table 2). The *c*-statistic for this model is 89.4178, the misclassification rate is 13.65% and the sse is 569.20. This gives a 1.12% improvement in the *c*-statistic, a 8.32% improvement in the misclassification rate and a 11.9% decrease in the sse.

Although not as good as the previous model, it is still a significant improvement over our baseline model. Furthermore, this is an extremely simple computation that would require very little time to compute.

The formula also generalizes to larger *N* as the contributions of the inferior models in the forest tend to approach *0*. This makes intuitive sense as the effect of random errors are mitigated. If the misclassification rate on the validation data set is replaced with misclassification rate on the out-of-bag sample, it would be a simple step to update the HP Forest node with this new result.

APPROXIMATING THE DEGREES OF FREEDOM *K*

The problem when trying to compute SBC values for decision trees (highlighted in Section 2) is that we do not have the degrees of freedom *K* for a decision tree. The AIC and SBC information criteria considers the tradeoff between fit and complexity. The principle is to penalize the fit for the complexity. For a decision tree we need to count the number of independent parameters. In Ritschard and Zighed (2003)

$$K = (r - 1)(c - q)$$

is given as the degrees of freedom for a induced/constructed tree, where q is the number of leaves in the tree, r the number of variables in the tree and c the product of the number of distinct levels for each of the r variables in the tree.

Although the formula for *K* looks simple, for any tree of reasonable complexity with multiple occurrences of the same variable, and with continuous variables added, the formula became increasingly difficult to apply. *K* can also become extremely large for a seemingly simple tree.

However, this approach of directly computing *K* seems promising and will be further investigated in a follow-up paper.

DISCUSSION

The theory of Bayesian Model Averaging is well-developed and provides a coherent mechanism for accounting for model uncertainty. It is therefore surprising that it has not been applied directly to random forests.

Bayesian additive regression (Heranádez, 2016) was an attempt to create a Bayesian version of machine learning tree ensemble methods where decision trees are the base learners. BART-BMA attempted to solve some of the computational issues by incorporating Bayesian model averaging and a greedy search algorithm into a modelling algorithm.

The method proposed in this paper does not attempt to turn an ensemble of decision trees into a statistical model (with corresponding probability estimates and predictions). Furthermore, the base

learners (e.g. decision trees) are only combined in a novel way to produce a more accurate final prediction.

The way the decision trees are combined depends on the ordering of the decision trees from more accurate to less accurate. This was first achieved by building a surrogate neural network model for each tree and using the neural network models' ordering as a proxy for the decision trees' ordering.

The improvement in *c*-statistic, misclassification rate and sse confirmed our supposition that there is a better way of combining trees than the standard averaging used in random forests. The improvement is summarized in Table 6.

Model	***c*-statistic**	**MISC Rate**	**sse**
Random Forest (Ave)	88.30	14.89%	646.50
Random Forest (SBC)	89.34	13.37%	565.61
Random Forest (Scheme)	89.41	13.65%	569.20

Table 6. Results

The Bayesian Model Averaging on surrogate neural networks introduced in this paper elegantly mitigates the reliance on the expectation that random errors introduced from overfitting will cancel when the predictions are averaged. Complex models where overfitting might be an issue are penalized in their SBC values (because of the large degrees of freedom value *K* in the surrogate neural network) with a resulting reduction in weight or contribution to the final model. Smaller errors are introduced into the system than is the case with random forests and it pays off in a better final model with improved fit statistics.

A Bayesian approach for finding CART models was presented in Chipman, George and McCulloch (1998). The approach consists of two basic components: prior specification and stochastic search. The procedure is a sophisticated heuristic for finding good models, rather than a full Bayesian analysis.

In a sense the procedure outlined in this paper is also a sophisticated heuristic that is used to compute the contribution of each tree to the forest, but with full Bayesian Model Averaging implemented on surrogate neural networks instead of the actual decision trees.

CONCLUSIONS

Although only a small change was proposed to the random forest algorithm, the improvements as shown in this paper could be substantial. However, the method depends on computing SBC values for decision trees which is problematic as a decision tree is not regarded as a statistical model.

The way around this problem is to use the SBC computed by a surrogate neural network. This gave an ordering of the models from good to bad. This information was then used to vary the contribution of each decision tree to the final model. Although a smart approximation, it still required a neural network to be built.

In a further simplification, validation misclassification rate was used to rank models and the contribution of each model to the final prediction was computed with a novel weighting scheme. The last results were still substantially better than that of the standard random forest approach, but not as good as when a neural network was used to approximate SBC.

REFERENCES

Breiman, L. “Random Forests.” Statistics Department, University of California Berkeley, CA. 2001. Available at http://leg.ufpr.br/lib/exe/fetch.php/wiki:internas:biblioteca:randomforest.pdf

Breiman, L. 2001b. “Random Forests.” Machine Learning, Vol. 45(1):5-32.

Chipman, H. A., George, E. I. and McCulloch, R. E. 1998. “Bayesian CART model search (with discussion and rejoinder by the authors).” Journal of the American Statistical Association. Vol. 93:936-960.

De Ville, B. and Neville, P. 2013. “Decision Trees for Analytics Using SAS Enterprise Miner.” SAS Press, Cary, USA.

Du Toit, J. V. 2006. “Automated Construction of Generalized Additive Neural Networks for Predictive Data Mining.” PH.D. thesis. School for Computer, Statistical and Mathematical Sciences, North-West University, South Africa.

Heranádez, B. “Bayesian additive regression using Bayesian model averaging.” 2016. Available at http://arxiv.org/abs/1507.00181

Hodges, J. S. 1987. “Uncertainty, policy analysis and statistics.” Statistical Science, Vol. 2(3):259-275.

Hoeting, J. A. “Methodology for Bayesian model averaging: an update.” Colorado State University. n. d. Available at http://www.stat.colostate.edu/~nsu/starmap/jab.ibcbma.pdf

Hoeting, J. A., Madigan, D., Raftery, A. E. and Volinsky, C. T. 1999. “Bayesian model averaging: a tutorial.” Statistical Science, Vol. 14(4):382-417.

Kaggle. 2016. Available at https://www.kaggle.com/

Learner, E. E. 1978. “Specification searches: ad hoc inference with nonexperimental data.” Wiley Series in Probability and Mathematical Statistics, John Wiley and Sons, New York.

Lee, H. K. H. 1999. “Model selection and model averaging for neural networks.” PH.D. thesis. Department of Statistics, Carnegie Mellon University.

Lee, H. K. H. “Model selection for neural network classification.” 2006. Available at http://citeseer.ist.psu.edu/leeoomodel.html

Lichman, M. “Machine Learning Repository.” University of California, Irvine, School of Information and Computer Sciences. 2013. Available at http://archive.ics.uci.edu/ml

Madigan, D. and Raftery, A. E. 1994. “Model selection and accounting for model uncertainty in graphical models using occam’s window.” Journal of the American Statistical Association, Vol. 89(428):1535-1546.

Miller, A. J. 1984. “Selection of subsets of regression variables.” Journal of the Royal Statistical Society, Series A, Vol. 147(3):389-425.

Regal, R. R. and Hook, E. B. 1991. “The effects of model selection on confidence intervals for the size of a closed population.” Statistics in Medicine, Vol. 10:717-721.

Ritschard, G. and Zighed, D. A. 2003. “Goodness-of-fit measures for induction trees.” Foundations of Intelligent Systems, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Vol. 27:57-64.

ACKNOWLEGDEMENTS

The authors wish to thank SAS Institute for providing them with Base SAS and SAS Enterprise Miner software used in computing all the results presented in this paper. This work forms part of the research done at the North-West University within the TELKOM CoE research program, funded by TELKOM, GRINTEK TELECOM and THRIP.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Tiny du Toit
Tiny.DuToit@nwu.ac.za