Papers

Merits of interactive decision tree building — Part 2: How to do it

Received (in revised form): 11th September, 2007

Bas van den Berg

is Principal Consultant at the marketing intelligence department of VODW Marketing (www.vodw.com). His core business is helping companies make their marketing activities more efficient and effective based on facts. His fields of interest span: predictive modelling, lifetime value management and retention.

Tom Breur

runs consulting firm XLNT Consulting (www.xlntconsulting.com) dedicated to helping companies make more money with their data. His fields of interest span: data mining, analytics, data quality, IT governance, data warehousing and business models.

Keywords *decision tree, data mining, targeting, direct marketing, model, monitoring*

Abstract In the previous paper, the authors explained why there is a tendency to embed data mining solutions in end-to-end software solutions. The advantages of integrated data mining solutions lie in making the process less people dependent, but the disadvantages are that learning from the mining process is hampered. The topic of the previous paper was *why* to build data mining models interactively. In this paper, the authors will explain *how* to build decision trees interactively. In this paper, we will demonstrate how interactive model building generates more knowledge on customer behaviour and on the structure of the data. The authors present guidelines for interactive tree building. These guidelines demonstrate how knowledge on when and how the model will be deployed can be taken into account to optimise the model. Furthermore, they illustrate how the context of the business problem that is being addressed with data mining can and should be taken into account when developing models.

Journal of Targeting, Measurement and Analysis for Marketing (2007) **15,** 201–209. doi:10.1057/palgrave.jt.5750054

INTRODUCTION

In the previous paper, the authors have made a case for interactive model building. Many commercial products for analytical CRM offer integrated solutions for model building that have largely automated the process to create data mining models.

Although the authors acknowledge the advantages that automatic model building offers, in most practical settings we feel that organisational learning is hampered by automatic model building. It has been our experience that automatically built models are of lower quality for two main reasons. First, automatically built models are less adapted to the business problem. Secondly, automatically built models have lower predictive accuracy. But what is worse, knowledge from model building no longer serves to improve the data sources and meta data knowledge that are urgently needed to keep improving on predictions in the mid to long term. These reasons together have led the authors to generally prefer interactive model building.

In the previous paper, we outlined *why* interactive model building leads to superior results; in this paper, we will provide practical guidelines on *how* to build tree models interactively.

Correspondence: Tom Breur, XLNT Consulting, Langestraat 8-03, Tilburg 5038 SE, The Netherlands.
Tel: +31 6 463 468 75;
E-mail: tombreur@xlntconsulting.com

AUTOMATIC VERSUS INTERACTIVE MODEL BUILDING

The choice between automatic versus interactive model building can apply across many data mining algorithms. In essence, interactive model building is characterised by the miner overriding statistical parameters on the basis of previous experience, and thus inputting domain knowledge in the model-building process. These approaches are sometimes labelled 'model engineering'.

As an additional output from interactive model building, the miner gains additional insight into the data, and the business phenomenon one is modelling. Although the principles of interactive model building can pertain to many algorithms, in this paper the authors have chosen to provide guidelines specifically for decision tree building.

PRACTICAL GUIDELINES TO INTERACTIVE TREE BUILDING

How to decide on a split

Interactive tree building consists of carefully selecting and oftentimes moulding individual splits. Each splitting point in the tree is handled manually, one by one. In this section, we will describe practical guidelines that the authors have found useful for this process of interactive tree building. With some people, data mining has, or used to have, a poor reputation. Data mining was often associated with looking for patterns in the data in an exhaustive way that would then bring up spurious results. To deal with such risks, the mining set is split, at random, into three parts: the training, test and evaluation set. The training set is used to discover the splitting points in the data set. The test set is used to see how reliable the chosen splitting points are. The evaluation set is tucked away and never touched until one has decided on the final model. Since the actual performance of the final model as a whole is never estimated until it is completed, and then held against the evaluation set, these findings are expected to hold up in the population at large from which the evaluation set was drawn. Clearly, this hinges on the assumption that the mining set is representative of the population from which it is drawn.

We emphasise here that the success of interactive model building that allows the miner, at times, to override statistical evidence in the training data ultimately depends on the *experience* of the miner. Mining experience and insight into prevailing business constraints are the keys to success. A less experienced miner should be more cautious and conservative in moulding splits at will. Experience and judgment underpin these choices, in particular experience from evaluating previously deployed models. Without sufficient experience, one risks making poor choices and winding up with an interactively built model that performs worse in the real world than one that would have been automatically built.

This holds in particular when the size of the mining set is relatively small. In such cases, the miner is often asked to simply make the best of it, which could imply that there are insufficient data to set aside an evaluation set. Or sometimes even a testing set, which we label an emergency scenario, not to be recommended to the faint at heart. One then relies entirely on experience and domain knowledge of the miner.

Minimising the loss function

When deciding on a split, the very first thing to look at is the value of the loss function (eg Chi-square, Gini index, Entropy, etc). The split with the lowest value is the first one to consider. It will be clear that the split with the lowest value for the loss function is the one that would have been chosen in an automatic tree-building mode. The magnitude of the loss function for consecutive splitting options should be assessed first. But besides the loss function there are additional factors to consider.

Reliable splits

A split should be reliable. By this we mean that the effect of the explanatory variable on the target variable should be approximately equal between the train and the test set. If this is not the case, the miner is at serious risk of overfitting the data. The effect in the train set will then not generalise to other data sets, and such splits should be pruned.

In the undesirable case when there are not enough data to put aside a separate test and evaluation set (we labelled this an 'emergency scenario'), the following rule of thumb may be applied: the direction of the effect (increase/decrease) on the target variable should be the same *throughout the tree*. This rule of thumb precludes certain effects to be included (eg parabolic functions). Experience has shown that these kinds of effects risk overfitting the most.

Searching for robust effects

A split should be robust. By this we mean that the effect of a variable should hold over time. This way, the model can be reused for subsequent campaigns with the same product. It will be clear that robustness cannot be determined from the mining set itself. Previous experience with monitoring models is the guide here. Robustness is not the same as reliability of a split. It is possible that on a given moment in time (eg during a stock recession), a certain split is reliable, but only for a short period of time. The robustness of a split is frequently characterised by the following two rules of thumb:

1. Monotonic effects

Effects should preferably be monotonically increasing or decreasing. A monotonic effect shows a steady increase or decrease in values of the target variable between adjacent leafs. In other words, for higher values of the predictor or splitting variable, the response variable shows a steady increase or decrease. Only in rare cases do robust effects come in the form of a quadratic function. In our experience, monotonic effects have proven to be the most robust. One reason for this is that if the apex of a hyperbole should move along the *x*-axis over time, this would result in a reversal of the effect as specified by the model. Clearly, this will cause the model to make very poor predictions. Besides this monotonic increase there may also be saturation effects. In these cases, after a monotonic rise a flattening out or even a decrease may set in (quadratic function). The explanation may be that for very high values of the explanatory variable, a substitute product for the one offered may become more appealing.

For information on balances, and moving averages (as derived for instance from transaction data), the effect should always show that an increase in the time window that is used to calculate averages should be associated with a monotonous increase or decrease in response. In the case of an increase in the number of transactions (an indication of activity), the response probability for an offer to take up electronic banking will rise. When more transactions in the previous month are associated with a higher response probability, one typically expects this same direction of effect for the number of transactions in the past 3, 6 and 12 months. Of course, the strength of the association can differ.

2 Meaningful relations between variables

The effects in the tree should be plausible to domain experts. For instance, in the case of a savings product, which variables show up frequently in predictive models for savings? What direction of effect do these variables tend to show when predicting acquisition of savings products? Experience within a given domain is paramount for determining which effects are plausible. Clearly, there is a tension here between finding obvious and trivial results that are of little value to the business, and peculiar but potentially interesting relations in the data. No textbook on data mining has been able to formulate general rules, yet, to single out such 'interesting' results. This might just be another domain where human expertise is not likely to be replaced by artificial intelligence in the foreseeable future.

Another way to use domain knowledge would be as input for deriving new variables. If, for instance, geographic location turns out to play an important role when predicting sales of mortgages, then maybe inferred information on how often people change address within each region proves to hold predictive power. In many cases, such variables first need to be 'constructed' ('feature extraction'), derived from indirectly existing information.

Rules of thumb to assess plausibility of effects:

— When ownership of a certain product typically means that the chance of uptake

of the target variable increases, one should consider whether this product appeals to a complementary need. For example, in the case of electronic banking services one could imagine that customers who own stock deposits (more active traders) would be more inclined to use this service. Reverse effects may exist, too. As an example, having a loan usually decreases the chance of response to acquire a savings product, because these two products seem to fulfil contradictory needs.

— A general rule when judging the plausibility of effects is that domain experts should be able to provide some heuristic explanation of a discovered effect. This goes in particular for 'unexpected' results. Be weary of the fact, though, that there is often an explanation available for almost *any* effect, and sometimes even for the same effect in both directions (!).

Number of respondents in a leaf

A split should contain sufficient respondents in each leaf. We consider five respondents the absolute minimum, 10 respondents as reasonable and more than 20 a 'perfect' situation by our standards. Bear in mind that for most statistical procedures there is no 'minimum lower bound' for the loss function to be calculated. Representativeness considerations, however, typically come into play at the tails of distributions. Clearly, when very few cases are involved, deciding on the exact cut-off between categories (the split point between two adjacent leafs) is subject to sample fluctuations (due to very low density of state space). Therefore, the authors prefer to choose a split with a somewhat less favourable value for the statistical loss function to establish more reliable values for the splitting point.

As an aside, if one of the classes holds few respondents, the Chi-square test as used in CHAID tools does not perform well (other loss functions operate differently). The asymptotic limit of the Chi-square test does not hold any longer. This may lead to erroneous use of test statistics, because candidate splits are typically ordered on the basis of their *p*-value. The magnitude of the effect is overestimated in such cases and may cause erroneous choice of splitting variables. To our knowledge, no decision tree tool at the moment has 'the minimum number of respondents per class' as a default setting for building trees. This is a pity, as this would eliminate these types of error and possible overfitting of data. Currently, we are limited to the default setting 'minimum number of records per leaf', which is taken regardless of the distribution of the target variable. As described in section 'Reliable splits', splits in areas of state space with an extremely low density tend to be unreliable, and hence often show unacceptable differences between the train and the test set.

Applicable cut-off point at deployment

When deciding on a split, one should take into account what percentage of the population will be targeted. A split only matters to the target group when one or more of the leafs fall within the group that will be targeted, and others outside. Hence, the child leafs from a split should end up to the left and right of the cut-off point. In such cases, the response percentage in the targeted group goes up and the response percentage in the deselected group goes down. When all leafs of a split fall within the targeted group or outside it, the split is of no value (at least not at *this* particular targeting depth).

How to deal with 'clusters' in the data

It is useful to balance the leaf sizes of a split. This applies in particular nearer the root node of the tree. When 50 per cent of the population will be targeted, it is much more important to 'disentangle' high up in the tree than if only 10 per cent of the population will be targeted. In the latter case, it would be more important to identify small sub-groups with a high response percentage. If you start looking for small groups when the target group will be 50 per cent of the population, then chances are that the model will display excellent lift for the first 10–20 per cent, and a poor lift for the rest. Experience has shown

that selecting small groups with a high response percentage early on in the tree (near the root node) tends to leave large groups later on in the tree that cannot effectively be split any more. Clearly, this does no good for the lift of the best 50 per cent. In such cases, it is sometimes better to choose some splits high in the tree that tend to break up the population, even if these appear to have less of an impact (locally!) on minimising the loss function.

Rearranging a split

When a split with the lowest value on the loss function does not match the previous criteria, one can begin by rearranging[1] this split. As an example we will demonstrate how to create a monotonic split (Figure 1).

The result of this split is a branch with five leafs. The fourth and fifth leafs are quite small, however, and what is worse, the response frequency in the fourth leaf is higher than the fifth leaf. We can 'fix' this by merging the fourth and fifth leaf, and then let the tool recalculate the value of the loss function. It will then necessarily be lower than it was before. The result is shown in Figure 2.

What happens here is that we are willingly giving up some lift (in the training set!) at this particular split. By merging some leafs, we obtain a larger portion of the records arriving at this node. Therefore, we now have more 'options' for the candidate splitting variables at the *next* level. One very useful side effect of this 'evening out' of effects is that the interpretation of the tree becomes much easier. Another side effect of rearranging the split in this way is that the statistical power is higher when response frequencies of groups are compared during evaluation after the campaign has been rolled out.

Determining the 'stop' criterion

If the rearranging procedure we described in section 'Rearranging a split' fails to create a satisfactory split for the variable with the best value on the loss function, the split with the second best value on the loss function will be considered. When this also does not lead to satisfactory results, then the next variable will be chosen, until eventually there are no more variables left to split a particular leaf. At that point, the tree simply stops growing. When all the splits in the tree are chosen in this manner, the model-building process stops by itself. The resulting tree then need not be tested. Basically, this has already been done, split by split. What remains to be done is to determine the expected predictive accuracy in the evaluation set.

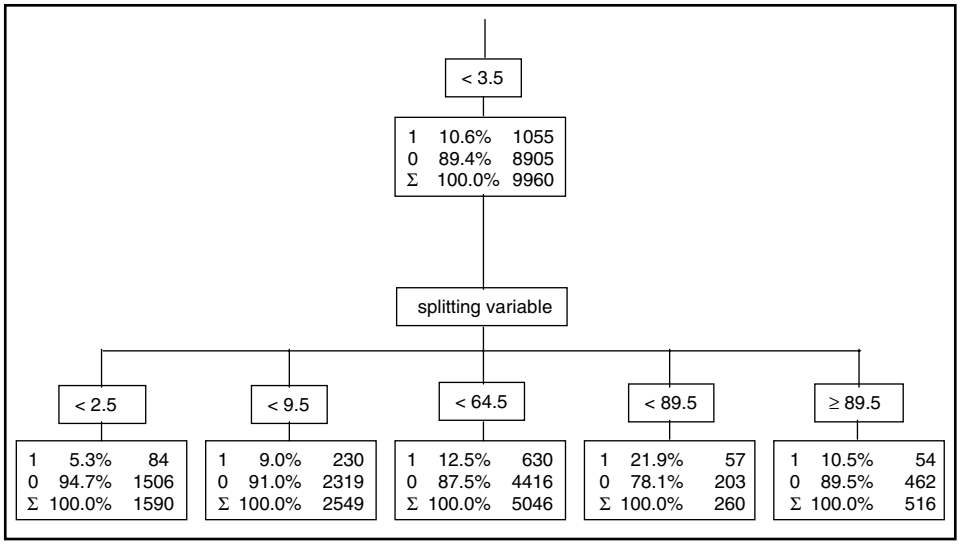

Figure 1: Example of a split as it initially appears in the tree

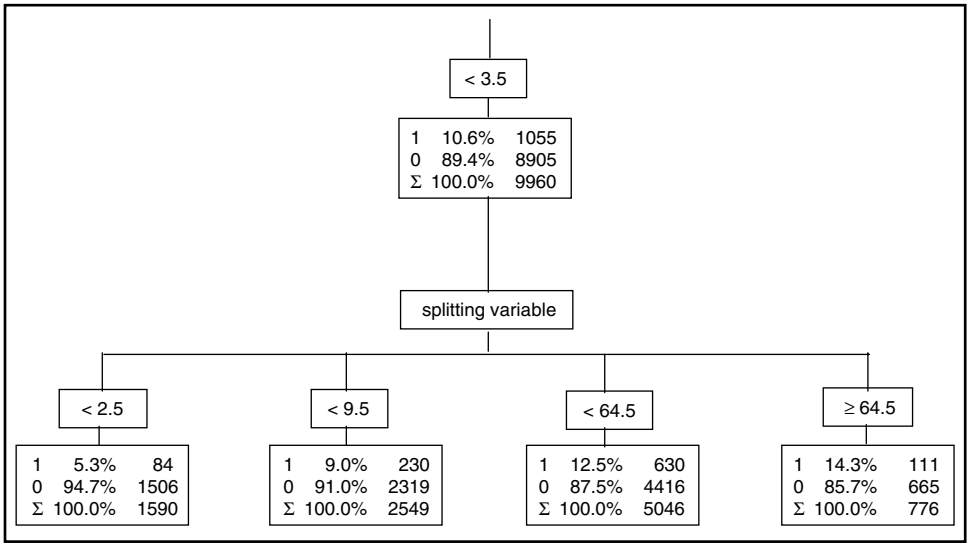

Figure 2: Result after rearranging the split

When a tree is built in automatic mode, the preset stop criterion will determine when the tree stops growing. In these cases, you may also 'test' the tree, but that process works quite differently. When you test an automatically built tree, you obtain response percentages for the chosen tree on the test set. If there appear large differences between response percentages in the train and test set, a variable can be removed, and basically the tree needs to be rebuilt from the ground up (or that sub branch pruned). Then you test again, repeating the same procedure. Note that in this automatic procedure, one might need to remove variables that can be usefully applied (when the splits are locally moulded) in a manual procedure. The function of testing in an automatic procedure works quite differently, and is much less effective.

Default settings

What are the consequences of this interactive process for the default settings? How should they be set, and how do they play a role in the process? The most important default settings:

(1) *Minimum number of observations in a leaf*: 20. Like we mentioned before, a split should contain a sufficient number of respondents in all leafs. At least ten respondents is desirable. Respondents and nonrespondents together should show at least 20 observations in each leaf.

(2) *Number of observations needed for a split search*: 50. To wind up with two leafs, each with at least 20 observations (minimal leaf size), one needs more than 40 observations. Because a split into two groups of exactly 20 each would be rare, we set this number at 50. Fifty is not a practical number, however, for the entire interactive process, in particular higher up near the root of the tree. Preferably, this setting should change as one moves through the layers of the tree: higher numbers close to the root node, and smaller numbers down below. To our knowledge, current commercial tools require that this setting is fixed throughout the tree.

(3) *Maximum number of classes from a split*: 5. Binary splits tend to grow more elaborate trees. We prefer a split on more categories: 5. This number will cause the tree to become 'wider' and therefore easier to 'read'. We feel that this allows the miner to input more of his domain knowledge in the tree-building process. Because of the extraneous demands that are set on the splits in the interactive process like we describe it, it is desirable not to choose this maximum number of classes too large. Otherwise, almost every split needs a lot of rework to make it match our criteria, and

the more the categories from a split, the more cumbersome this process becomes.

(4) *Maximum depth of the tree*: 10. In the aforementioned, we noted how the interactive tree-building process stops 'by itself'. We choose the value 10 just to set the minimum number high enough. Hence, a 'maximum depth of the tree' is not actually necessary. This is why we set this maximum to ten, a level that is rarely attained. Trees that are deeper than this become very difficult to comprehend.

For the above-mentioned default settings, in particular the first and third are used to determine the best split on the basis of the loss function. The second and fourth criteria are deliberately chosen wide enough to allow splitting of leafs until one runs out of observations or splitting variables.

Making model predictions

After the final model has been chosen, this model now needs to be applied on the third portion of the initial mining data: the evaluation set. This portion of the data has so far never been touched or considered in the model-building process. After the model has been scored on the evaluation set, the miner can give reliable, unbiased estimates of the actual response prediction. Given these response prognoses, a rational allocation decision for the marketing budget can be made, based on ROI calculations per segment. Only segments with a positive Net Present Value should be contacted.

Rearranging segments for better model monitoring

In our previous paper, we briefly touched upon a procedure we have used to enhance model monitoring. When there are many small segments, this makes it difficult to come up with statistically powerful and reliable evaluations after the campaign has been rolled out. This is because the evaluation ought to be performed on a random group, and commercial pressure will be to keep this as small as possible because the expected response in this random group will be lower than the target group. More about this in another paper by Breur.[2]

The procedure we use for decision trees (or other segmented modelling techniques) works as follows: a new data set is created where segment number as derived from the first stage model is appended to every record, alongside the response variable. Then a *new* tree model is built, with *only* the segment number as a predictor for the target variable. What is accomplished by this two-stage approach is that nonadjacent leafs from the tree will be 'merged' in the final tree on the basis of statistical criteria that will determine whether their response frequencies are sufficiently similar. The outcome from this second stage model is a model with fewer, but larger segments. This second stage merging of tree segments serves to facilitate model monitoring and enhances statistical power when the model needs to be evaluated after the campaign.

Checking of the segment distributions at scoring time

The final model will be deployed on the run-time population. Every customer in the population will be attributed to a model segment, and a corresponding response prognosis. On the basis of these scores and a chosen threshold, the size of the target group is determined.

After the population has been scored, it is good practice to check the distribution of the segments to see whether the relative size of the segments at scoring time (still) matches the expected distribution on the basis of the mining set. If the miner finds unacceptable differences in the segment distributions here, there might be two plausible explanations:

— Either the population has begun 'drifting', meaning that the relationships that were found at the time the mining set was assembled no longer hold in the current population at large,
— Or there might be a data quality issue, in which case this needs to be researched and corrected.

In the former case, the business is faced with a quandary: apply a sub-optimal model and face disappointing response rates, or perform rework and scrap to come up with an entirely new

model, if this is at all possible and feasible within the given timelines. It is much better to have this difficult discussion *before* the campaign, rather than afterwards. If it turned out to be a data quality issue, fix the problem, deploy the model again and roll out.

SUMMARY

The advantages of interactive decision tree building that we see are:

— The use and generation of domain knowledge is stimulated. On the input side, domain knowledge can be used as an input for the model-building process. On the output side, new knowledge on customer behaviour and the data is gained. It is in particular this 'output knowledge', the knowledge gained from model building, that holds the most promise in deriving new variables ('feature extraction'). These newly derived variables are quintessential in working towards more accurate predictive models in the mid to long term.
— Interactively built trees are easier to interpret, and therefore they are more insightful. The substantive interpretation of a data mining model is of paramount importance in getting support for its use, and in creating useful new insights that can move the business forward.
— Interactively built trees are easier to test when deployed. When carefully matched to the business situation, they will have been designed with monitoring in mind. Therefore, testing how the model keeps performing when it is being reused should be straightforward, and statistically powerful.
— The chosen splits are better tuned to the business problem, and the model meets the requirements of the data mining problem better. Considerations like at what targeting depth the model should perform best have been taken into account when the model was built. The trade-off between short-term maximum lift and long-term performance of the model has been considered. As such, the 'right' model for this business problem has been generated.
— Possible errors in the model-building process are detected earlier on. Databases are known to contain errors, anomalies and sampling variations. An adept miner will be aware of this, and will minimise the chances that 'a' particular data set pulls such tricks on him. Let the data speak, yes, but do not let them fool you.
— The predictive quality of the model/scoring code is better. This does not necessarily translate into higher prognoses. It may also be that prognoses are somewhat lower, but remain valid for a longer period of time.

These advantages of interactive tree building, in our mind, generally outweigh the advantages of the automatic model-building process.

Oftentimes, the splits that are chosen on the grounds mentioned above are not the splits with the lowest value in terms of the loss function that should be minimised. But the loss function is only *one* possible criterion to evaluate, and as such only operates on this particular split. Keep in mind that there is never 'one right model', but instead there are always several possible models. By merely choosing a different split variable at the first level, one can return a tree with completely different variables. Most often, the predictions from these two trees (apparently completely different models) are nonetheless very highly correlated.

What counts is how successful the *entire* tree is in minimising the loss function, and how accurate this tree will turn out to be *over its entire lifetime*. By their nature, decision tree algorithms are extremely 'short-sighted': they only consider statistics on the particular split at hand. The criteria mentioned above take other factors into account besides the loss function, like looking at the entire tree and taking business constraints (eg targeting depth, model monitoring) into consideration when deciding on 'the best tree'.

Reference and Note

1 Rearranging here refers to changing the distribution among leafs.
2 Breur, T. (2007) 'How to evaluate campaign response — The relative contribution of data mining models and marketing execution', *Journal of Targeting, Measurement and Analysis for Marketing*, Vol. 15, No. 2, pp. 103–112.

Further Reading

Mittchell, T. (1997) 'Machine Learning', McGraw Hill, New York, NY.
Dhar, V. and Stein, R. (1997) 'Seven Methods for Transforming Corporate Data into Business Intelligence', Prentice-Hall, Upper Saddle River, NJ.
Witten, I. H. and Frank, E. (2000) 'Data Mining', Morgan Kaufman, San Francisco, CA.
Pyle, D. (2003) 'Business Modelling and Data Mining', Morgan Kaufman, San Francisco, CA.
Abelson, R. P. (1995) 'Statistics as Principled Argument', Lawrence Erlbaum Associates, Hillsdale, NJ.
Hogg, R. V. and Tanis, E. A. (1997) 'Probability and Statistical Inference', Prentice-Hall, Upper Saddle River, NJ.
Hastie, T., Tibshirani, R. and Friedman, J. (2001) 'The Elements of Statistical Learning', Springer-Verlag, New York, NY.
Sheskin, D. J. (1997) 'Handbook of Parametric and Non-Parametric Statistical Procedures', CRC Press, Boca Raton, FL.
Kremer, J. (1992) 'The Complete Direct Marketing Sourcebook', John Wiley & Sons, New York, NY.
Davies, J. M. (1992) 'The Essential Guide to Database Marketing', McGraw-Hill, New York, NY.