Improved Customer Choice Predictions using Ensemble Methods

Michiel C. van Wezel [*] Rob Potharst

Faculty of Economics, Econometric Institute, Erasmus University
P.O. Box 1738, 3000 DR, Rotterdam, The Netherlands

Econometric Institute Report EI 2005-08

March 15, 2005

Abstract

In this paper various ensemble learning methods from machine learning and statistics are considered and applied to the customer choice modeling problem. The application of ensemble learning usually improves the prediction quality of flexible models like decision trees and thus leads to improved predictions. We give experimental results for two real-life marketing datasets using decision trees, ensemble versions of decision trees and the logistic regression model, which is a standard approach for this problem. The ensemble models are found to improve upon individual decision trees and outperform logistic regression.

Next, an additive decomposition of the prediction error of a model, the bias/variance decomposition, is considered. A model with a high bias lacks the flexibility to fit the data well. A high variance indicates that a model is instable with respect to different datasets. Decision trees have a high variance component and a low bias component in the prediction error, whereas logistic regression has a high bias component and a low variance component. It is shown that ensemble methods aim at minimizing the variance component in the prediction error while leaving the bias component unaltered. Bias/variance decompositions for all models for both customer choice datasets are given to illustrate these concepts.

Keywords: Bagging, Bias/Variance Decomposition, Boosting, Brand Choice, CART, Choice Models, Data Mining, Ensembles.

1 Introduction

Understanding, modeling and predicting customer choices has always been an important branch of marketing research. The 2004 Marketing Science conference in Rotterdam featured 23 presentations where choice models where an important or the main topic, indicating that customer choice modeling is still an active research field. Many textbooks on marketing models devote one or more chapters on modeling individual's choice probability (see, e.g., [12, 28]). Customer choice models are valuable assets for marketing managers, who can use them to assess how manipulations of marketing variables as price and amount of advertising will influence market shares and thus perform 'what-if' simulations. The widespread academic interest and practical applicability of these models justifies the search for ever more accurate and better performing models.

In recent years there has been a growing interest in the machine learning and statistics communities in methods for combining multiple predictions. The most widely used and studied methods are bagging [4] and boosting [13, 7, 25]. These so-called ensemble learning methods, also known as committee methods, enable the user to get more accurate predictions than the predictions generated by individual models or experts on average.

[*]Corresponding author. Phone: +31 10 4081341. Fax: +31 10 4089167. Email: mvanwezel@few.eur.nl

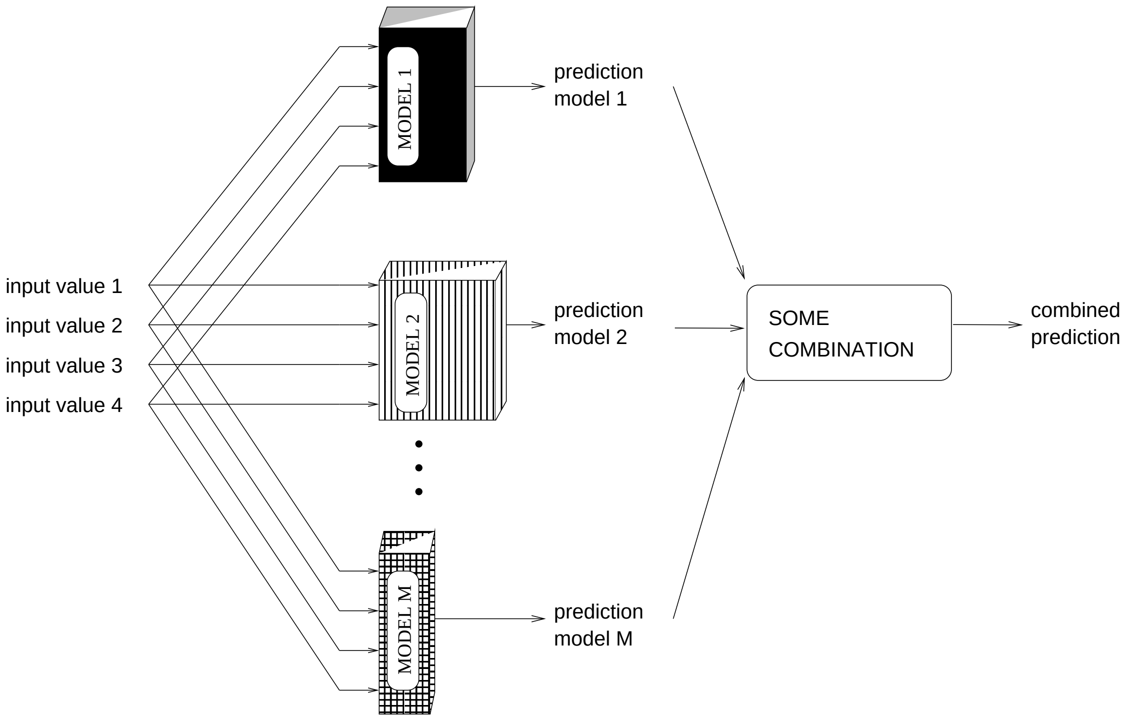

Figure 1: Schematic representation of an ensemble.

When multiple models make a prediction for the same phenomenon, there is often some amount of ambiguity in these predictions: They differ over the models. This is the case for human experts, e.g., the members of the jury in a figure skating contest that 'predict' the final score of a participant, as well as for statistical models, e.g., multiple models for credit scoring fitted on different data. In ensemble learning, the individual predictions are combined to arrive at a combined prediction. A schematic representation of an ensemble is shown in Figure 1.

Empirical results show that these combined predictions are often more accurate than their 'individual' counterparts. This is illustrated in Figure 2. This graph shows the percentage of correctly classified samples for a credit-scoring dataset taken from the Internet [22], where the problem was to predict at the time of application for a loan, whether the applicant would repay the loan or not. Just by combining multiple models for this case, the error rate drops from 32% to 25%. It will be explained below how the ensemble used in this example was generated.

To our knowledge there have been very few applications of ensemble learning within marketing. In a paper by Hu [24] ensemble learning was applied to the modeling of customer responses. In this article, customer choice data are analyzed by an ensemble of neural networks: a set of individual neural networks are 'trained' to model what type of long distance communication – postcard or telephone call – is chosen by a household on the basis of situational and demographic variables. Subsequently, the predictions of these individual models, called the level 0 models, are combined using a second model – the level 1 model. This method is referred to as 'stacking' [37].

Stacking, the method applied in [24], is just one of a number of possible ensemble methods. In this paper we extend the work in [24] in two ways. First, we apply a number of other ensemble schemes to the customer choice problem, i.e., boosting and bagging. These ensemble methods are combined with various choices for the base classifier: We use the logistic regression model and CART trees as the base classifiers. Second, these methods are described in detail and it is explained why they work from a statistical point of view. This way, we hope to give the reader understanding of these methods and the necessary confidence for their use.

The organization of this paper is as follows. The next section (Section 2) is devoted to customer

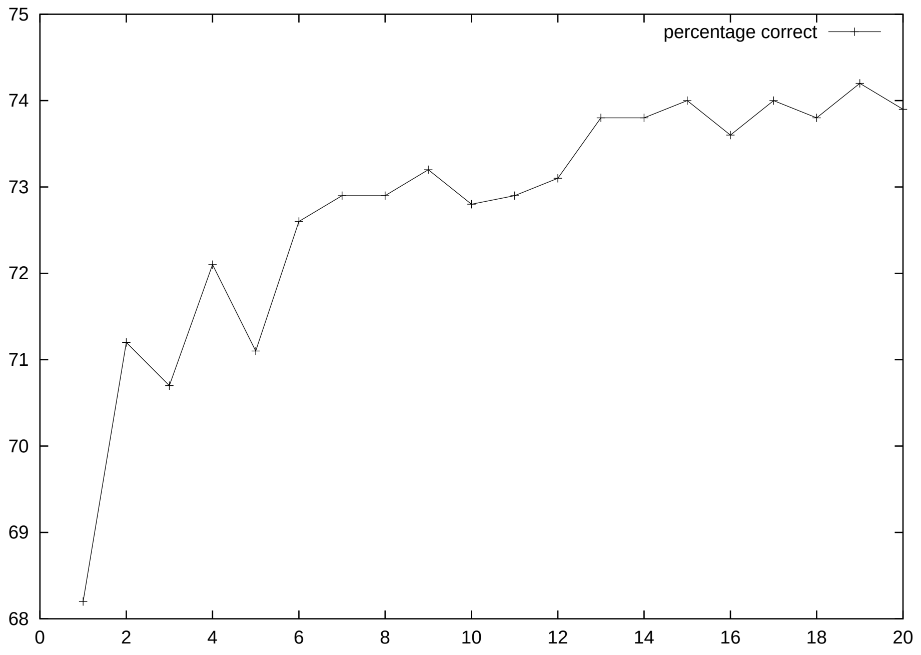

Figure 2: Error improvement when applying ensemble learning for creditscoring. The horizontal axis shows the number of models combined, the vertical axis shows the percentage of correct predictions.

choice modeling, classification and classifiers. It deals with the statistical models we use throughout the paper: the logistic regression model and CART. Section 3 then deals with the ensemble methods we use in our analyses: bagging and boosting. Both the individual classifiers and the ensemble methods are described in a level of detail that hopefully provides enough 'how-to' material to enable the reader to apply them in his own context. Next, Section 4 describes the experiments that were performed and the results that were obtained. Some theory supporting the experimental results in Section 4 is given in Section 5, where the so called 'bias-variance decomposition' of the prediction error is introduced. This decomposition is additive and it is illustrated that ensemble methods can be expected to lower the 'variance' term in the decomposition. A decomposition of the prediction error obtained in the earlier experiments is given. Finally, a summary, a discussion and some conclusions are given in Section 6.

2 Customer choice modeling, classification and classifiers

In customer choice modeling one attempts to create a model that predicts which product (or brand or service) a customer will buy on the basis of a number of features of this customer, the product and the situation in which the purchase occurs. Stated in a machine learning jargon, one classifies a feature vector $\mathbf{x}$ into one of a number of disjoint classes. The customer choice model can thus be seen as a classifier, where the feature vector describes the customer, the product and the situation, and the classes are the possible brands, products or services, e.g., 'buys Coca-Cola', 'buys Pepsi', 'buys dr. Pepper', etc.. In this section and Section 3, we will use the terms 'classifiers' and 'classes' instead of 'choice models' and 'brands'. For a general text on classification we refer to [11].

Given a dataset with N pairwise observations of feature vectors and corresponding classes, the quality of a classification model can be measured using the zero-one loss function:

$$\sum_{n=1}^{N} I(\text{predicted class}, \text{actual class}), \tag{1}$$

with $I(a,b) = 1$ if $a \neq b, 0$ otherwise. This error function is particularly useful if the classifier generates 'crisp' classifications.

Instead of giving crisp classification, some types of classifiers give class membership probabilities. Suppose that we have K classes in total, denoted $C_1, \ldots C_K$. Then we can express the unconditional occurrence of a class C_i by a K-dimensional vector $\mathbf{z} = (z_1, \ldots, z_K)^T$ in which only one element (z_i) is 1, the rest is 0. Regarding $\mathbf{z}$ as a random variable, the appropriate distribution is the multinomial distribution with parameters $(n = 1, \mathbf{p})$ [1]

$$P(\mathbf{z}) = \prod_{k=1}^{K} p_k^{z_k}, \tag{2}$$

where the vector $\mathbf{p} = (p_1, \ldots, p_K)^T$ contains the class probabilities, so $p_k = P(z_k = 1)$. Obviously, $\sum p_k = 1$. As an example, suppose that $K = 3$, $\mathbf{p} = (0.1, 0.5, 0.4)^T$, then the probability of $z = (1, 0, 0)^T$ is 0.1.

Given this probability model we can easily compute the likelihood $\mathcal{L}(D)$ for a set of observations $D = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$:

$$\mathcal{L}(D) = \prod_{n=1}^{N} \prod_{k=1}^{K} p_k^{z_{nk}}, \tag{3}$$

and its logarithm

$$\mathcal{LL}(D) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log p_k. \tag{4}$$

In the example, the likelihood of $D = \{(0,1,0)^T, (1,0,0)^T, (1,0,0)^T, (0,1,0)^T\}$ is 0.0025.

Now, suppose that we are given a data set $T = \{(\mathbf{x}_1, \mathbf{z}_1), \ldots, (\mathbf{x}_N, \mathbf{z}_N)\}$ consisting of feature vectors $\mathbf{x}$ paired with class indicator vectors $\mathbf{z}$. Suppose that want to model the probability that feature vector $\mathbf{x}$ belongs to class C_k. (In choice model terms this is the probability that C_k is the chosen brand for feature vector $\mathbf{x}$ describing customer, brand and situation.) This can be done using a function $\mathbf{f}(\mathbf{x}|\theta) = (f_1(\mathbf{x}|\theta), \ldots f_K(\mathbf{x}|\theta))^T$. The k-th component of $\mathbf{f}$ models the probability $p_k(\mathbf{x})$ that $\mathbf{x}$ belongs to class C_k, or, equivalent, the conditional probability of C_k given $\mathbf{x}$:

$$f_k(\mathbf{x}|\theta) = p_k(\mathbf{x}) = P(C_k|\mathbf{x}) = P(z_k = 1|\mathbf{x}).$$

$\mathbf{f}$ is parameterized by vector θ. Substituting $\mathbf{f}$ in (4) and negating gives a cost function

$$E(T|\theta) = -\sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log f_k(\mathbf{x}_n|\theta), \tag{5}$$

known as the cross-entropy, multinomial deviance or negative multinomial log-likelihood, that can be minimized as a function of θ. The maximum likelihood estimate for θ is thus

$$\hat{\theta} = \arg\min_{\theta} E(T|\theta). \tag{6}$$

For classifiers that (may) give crisp classifications cost function (5) easily leads to numerical problems: If the model incorrectly assigns probability 0 (or something close to 0) to the target class in one of the test patterns, the term corresponding to this test pattern in the cross-entropy function will be either undefined or cause an overflow. For these classifiers, cost function (1) is more appropriate. Nevertheless we use cost function (5) alongside cost function (1) in our model evaluations throughout this paper whenever possible due to the fact that it is the most predominant cost function in the brand choice literature.

[1] The multinomial distribution with parameters $(n, \mathbf{p})$ is given by

$$P(\mathbf{z}) = C(n; \mathbf{z}) \prod_{k=1}^{K} p_k^{z_k},$$

where $C(n; \mathbf{z}) = n! / \prod_{k=1}^{K} z_k!$ is the multinomial coefficient, which reduces to 1 for $n = 1$.

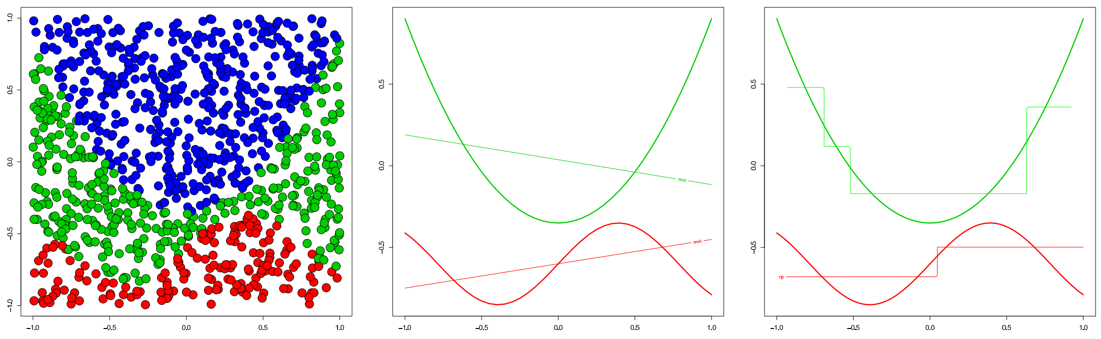

Figure 3: Classification of a CART model and a logistic regression model. Total dataset (l) and decision boundaries implemented by logistic regression (c) and CART (r). The implemented decision boundaries are shown as thin lines, the true boundaries as solid lines.

2.1 Logistic regression

It would be nice if **f** had some properties that make it possible to interpret the components as probabilities: The components should always sum to 1 and they should always be in the interval [0; 1]. These properties are satisfied when

$$f_k(\mathbf{x}|\theta) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{\sum_{k'}\exp(\mathbf{w}_{k'}^T\mathbf{x})}. \tag{7}$$

Here, θ is partitioned into K parameter vectors $\theta = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K)$, one for each class, or brand. (It is assumed here that x_0 takes a constant value so there is no need for explicit intercepts.) Each parameter vector is used to form a linear combination of the input features. This model is known as logistic regression. The decision boundaries between any two classes a and b in the feature space are linear, because they are separated by the hyper-plane $(\mathbf{w}_a - \mathbf{w}_b)^T\mathbf{x} = 0$, as can be seen as follows:

$$\frac{f_a(\mathbf{x}|\theta)}{f_b(\mathbf{x}|\theta)} = 1 \iff \log\frac{f_a(\mathbf{x}|\theta)}{f_b(\mathbf{x}|\theta)} = \log 1 \iff (\mathbf{w}_a - \mathbf{w}_b)^T\mathbf{x} = 0. \tag{8}$$

In customer choice modeling, related models called the multinomial logit model and the conditional logit model are often used but these models make the often unrealistic ‘independence of irrelevant alternatives’ assumption [29, 12].

2.2 CART

Instead of using a linear combination of features such as in (7) we could also use a function defined on a rectangular partition of the feature space, induced by a so-called decision tree. For instance, if we have two features x_1 and x_2, the classification tree of Figure 4 splits the feature space into five non-overlapping rectangular regions $R_1, \ldots, R_5$. Here, a binary split of the form $x_i \leq c$ splits a region into two subregions, one with $x_i \leq c$ (go left in the tree) and one with $x_i > c$ (go right in the tree). Thus, a classification tree θ splits the feature space recursively by binary splits into a number of rectangular regions $R_1, \ldots, R_m$. If p_{ik} is some estimate of the probability that a data point in region R_i will have class label C_k, then the tree model can be written as

$$f_k(\mathbf{x}|\theta) = \sum_{i=1}^{m} p_{ik}\chi_{R_i}(\mathbf{x}), \tag{9}$$

where $\chi_A(\mathbf{x})$ is the set-indicator function defined by $\chi_A(\mathbf{x}) = 1$ if $\mathbf{x} \in A$ and 0 if not. A well-known algorithm for generating a classification tree on the basis of a data set is CART [8]. A

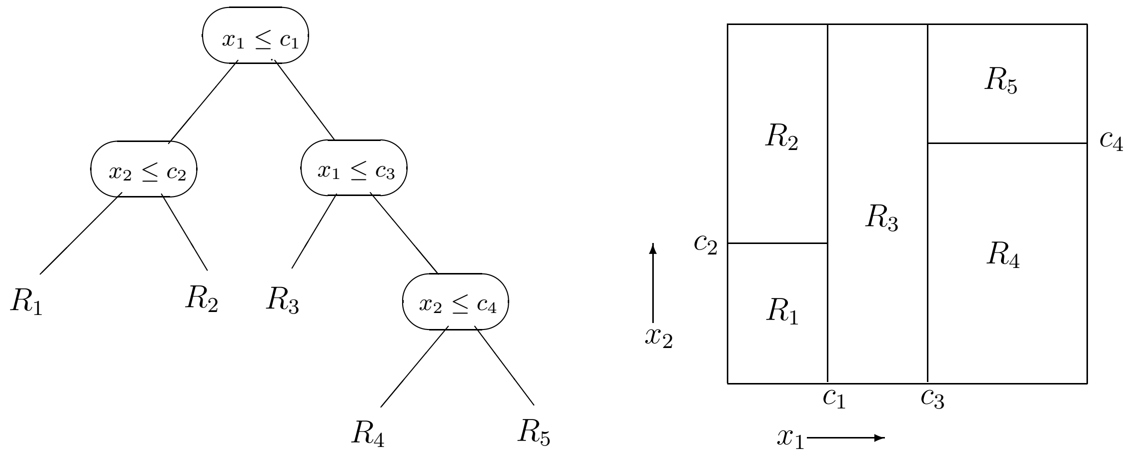

Figure 4: A classification tree (l), and a partitioning of the feature space.

variant of this algorithm produces so-called regression trees, which will be used in this paper in the logitboost algorithm.

An example of a classification problem modeled with logistic regression and a CART decision tree is shown in Figure 3. Here, we see uniformly distributed data in the (x, y) plane belonging to class `red, green` or `blue`. It is clear that the logistic regression model is unable to cope with the nonlinear decision boundaries while the CART model approximates these boundaries reasonably well.

3 Ensemble methods for classification

Although decision trees are good at approximating nonlinear decision boundaries, they are also instable: the implemented decision boundaries depend heavily on the composition of the training set. A small difference in the composition of two training sets may lead to a large difference in implemented functions for these training sets. Instability adds to the error of a model when making predictions on new data: when two models disagree on a prediction, at least one of them must be wrong.

As will become clear in Section 5, it is reasonable to expect that 'averaging' predictions over models fitted on different datasets lowers the prediction error. But how to aggregate over different datasets if the joint distribution $P_{\mathbf{x},\mathbf{z}}$ of $\mathbf{x}$ and $\mathbf{z}$ is known only to mother nature but not to us? In practice, all we have is a single dataset that was sampled from this distribution.

Below we describe two frequently used ensemble methods, bagging and boosting, that generate an approximate aggregated model. We subsequently apply these methods to consumer choice modeling in Section 4.

3.1 Bagging

Bagging, due to Breiman [4], is an acronym for bootstrap aggregating. It uses bootstrapping to create an aggregated model: given a dataset T, a number of so-called bootstrap datasets $T_1, T_2, \ldots$ are created by sampling from T with replacement. The T_t all have the same size as T. $T_1, T_2, \ldots$ are called bootstrap samples from T. A bootstrap sample T_t may contain some patterns (observations, instances, cases) in T multiple times, whereas others are not included. The idea of the bootstrap is that sampling from the actual dataset T is the best possible approximation for sampling from the unknown distribution $P_{\mathbf{x},\mathbf{z}}$. For each bootstrap sample T_t a model $\mathbf{f}_{T_t}$ is trained and these models are 'averaged' to form the aggregated model, also called 'bagged model'. The way in which this averaging is done depends on the loss function that is used. For classification under $0-1$ loss it is done by 'voting' — finding the most frequently predicted class among the individual models. For classification under cross-entropy loss or for regression it is just plain averaging. Section 5 pays more attention to this.

Breiman [4] reports that the classification error is reduced by 20% to 46% for a collection of test problems by using the bagged model instead of the original model.

Wagging [2] is an interesting variant of bagging where, instead of sampling the original training set by means of the bootstrap, new random weights are assigned to the patterns in the original training set in each iteration. For each weighted version of the dataset a separate model is built where the influence of each pattern on the error function is determined by its relative weight: The larger the weight, the larger the influence. This requires that the individual models are able to cope with pattern weights, but for most model types this is not a problem. Based on a statistical argument, Webb [34] suggests to generate these weights using the formula $w = -\log r/1000$ where r is a random integer in the range $1, \ldots, 999$. We use wagging below together with a second ensemble technique, boosting, in the 'MultiBoost' method.

3.2 Boosting

The second ensemble method we consider is called boosting, due to Freund and Schapire [14, 13, 30]. Boosting has received great praise in the machine learning, computer science and statistics communities. The inventors have received the prestigious ACM Gödel prize for it in 2003, and the method was referred to as 'one of the most powerful learning methods introduced in the last ten years' by Hastie, Tibshirani and Friedman in [20]. Boosting is sometimes called arcing in the literature [5, 6].

The general idea of boosting is to create a sequence of models, where each model is trained on a re-weighted version of the original dataset. Each training example in the dataset is assigned a weight and these weights are dynamically adjusted: when the model in a certain iteration makes an error in the classification of the n-th training pattern, the weight associated with this pattern is increased. This causes the model in the next iteration to focus on the patterns that were misclassified earlier. Continuing this way, an ensemble of models is created. The classifications of this ensemble are subsequently combined to form the final classifier. In the original boosting algorithm, AdaBoost, this is done using a weighted majority vote. However Breiman [5] showed experimental results suggesting that the weighted vote at the end of the boosting algorithm is not essential for the success of the boosting algorithm.

The original boosting algorithm, AdaBoost [14, 13], is intended for classifiers with zero-one error loss. In this section, we first describe AdaBoost, and then describe a similar algorithm, LogitBoost, which yields a classifier that gives class probability estimates. LogitBoost is due to Friedman, Hastie and Tibshirani [25]. Finally we consider an algorithm called 'MultiBoost' which combines boosting with bagging. This algorithm is due to Webb [34].

3.2.1 AdaBoost

As stated, AdaBoost [14, 13] is the original boosting algorithm. It is intended for 'crisp' classifiers delivering merely a class prediction rather than class probability estimates, i.e. assigning all the probability to one class. The description below is due to Breiman [6, 7]. The algorithm uses case weights $w_i, i = 1, \ldots, N$.

AdaBoost [14, 13], ARC-FS [6, 7]

1. At the m-th step, using the current weights $w_{i,m}$, sample with replacement from T to get the training set T_m and construct classifier $\mathbf{f}_m$ using T_m.
2. Run T down the classifier $\mathbf{f}_m$ and let $d(i) = 1$ if the i-th case $\mathbf{x}_i$ is classified incorrectly, 0 otherwise.
3. Define
$$\epsilon_m = \sum_i w_{i,m} d(n), \;\; \beta_m = (1 - \epsilon_m)/\epsilon_m,$$
and update the weights
$$w_{i,m+1} = w_{i,m}\beta_m^{d(i)} / \sum_i w_{i,m}\beta_m^{d(i)} \; .$$
4. After M steps, the $\mathbf{f}_1, \ldots, \mathbf{f}_M$ are combined using weighted voting with $\mathbf{f}_m$ having weight $\log(\beta_m)$.

This algorithm has shown to work well on a plethora of problem domains [31, 13, 2, 9].

3.2.2 LogitBoost

Despite the empirical success of AdaBoost, it was poorly understood in terms of traditional statistical models until Friedman, Hastie and Tibshirani showed there is a clear connection to additive logistic regression. In their paper ‘Additive Logistic Regression: a Statistical View of Boosting’ [25] they showed that AdaBoost is essentially a Newton-like minimization procedure for a cost function that is similar (but not identical) to the cross-entropy error function, thus maximizing the log-likelihood. This fact prompted them to derive an algorithm for direct (quasi) Newton-minimization of the cross-entropy. The resulting algorithm was called ‘LogitBoost’ and is given below.

LogitBoost creates the ensemble on the level of the logit transform of the class probabilities, and thus it uses a regression model as the base model. More precisely, the logistic transform is given by

$$p_k(\mathbf{x}) = \frac{e^{G_k(\mathbf{x})}}{\sum_{k=1}^K e^{G_k(\mathbf{x})}}, \tag{10}$$

and its inverse, the *logit* transform, is given by

$$G_k(\mathbf{x}) = \log p_k(\mathbf{x}) - \frac{1}{K}\sum_{j=1}^{K} \log \, . \tag{11}$$

The combined model $G_k(\mathbf{x})$ is formed by combining a number of base models on the level of the logit transform

$$G_k(\mathbf{x}) = \sum_m g_{mk}(\mathbf{x}), \tag{12}$$

hence the name LogitBoost.

```
LogitBoost ($K$-classes) [25]
1. Start with weights $w_{ik} = 1/N, i = 1, \dots, N, k = 1, \dots, K, G_k(\mathbf{x}) = 0$ and $p_k(\mathbf{x}) = 1/K \; \forall k$.
2. Repeat for $m = 1, 2, \dots, M$:
   (a) Repeat for $k = 1, \dots, K$:
       • Compute working responses and weights in the $k$-th class, for $1 \le i \le N$
             $w_{ik} = p_k(\mathbf{x}_i)(1 - p_k(\mathbf{x}_i))$
             $r_{ik} = (z_{ik} - p_k(\mathbf{x}_i))/w_{ik}$
       • Fit the function $g_{mk}(\mathbf{x})$ by a weighted least squares regression of $r_{ik}$ to $\mathbf{x}_i$
         with weights $w_{ik}, 1 \le i \le N$.
   (b) Set $g_{mk}(\mathbf{x}) \leftarrow \frac{K-1}{K}(g_{mk}(\mathbf{x}) - \frac{1}{K}\sum_{k=1}^{K} g_{mk}(\mathbf{x}))$, and $G_k(\mathbf{x}) \leftarrow G_k(\mathbf{x}) + g_{mk}(\mathbf{x}))$.
   (c) Update $p_k(\mathbf{x})$ via (10).
3. Output the classifier $\arg\max_k G_k(\mathbf{x})$.
```

To prevent confusion with earlier classifiers f that estimated class probabilities, we have designated the combined regression model G and the individual models g.

3.2.3 MultiBoost

The MultiBoost method (due to Webb [34]) combines bagging with boosting: boosting is applied to a number of bootstrapped versions of the original dataset. Webb uses wagging instead of ordinary bagging together with the AdaBoost algorithm. In our experimental results below, we combine wagging with the LogitBoost algorithm. The total number M of models in the ensemble (i.e. the sum of all models in the separate 'boosted' committees) is taken as an input parameter to the MultiBoost algorithm. Webb suggests to set the number of boosted committees and the number of models in each committee to $\lceil\sqrt{M}\rceil$ and we follow this approach. We thus use the following procedure:

```
MultiBoost [34]
1. For $m = 1, \dots, \lceil\sqrt{M}\rceil$
   • Create a 'wagged' dataset $T_m$ from the training set $T$,
   • Apply the LogitBoost algorithm to $T_m$ for $\lceil\sqrt{M}\rceil$ iterations. This gives classifier $f^b_m$,
2. Combine the classifiers $f^b_1, \dots, f^b_{\lceil\sqrt{M}\rceil}$ using unweighted voting.
```

It is important to distinguish between the weights used in the wagged dataset and the weights used internally in the LogitBoost algorithm. These weights are be multiplied in the LogitBoost algorithm.

The rationale behind the combination of bagging and boosting is that there is reason to believe that both methods work by a different mechanism: Bagging reduces variance while Boosting reduces bias and variance. (Section 5 will discuss bias and variance in more detail.) Because of the complementary nature of these methods it makes sense to use them combined.

4 Experiments and results

In this section we report on the experiments we performed and the results we obtained.

4.1 Data

We analyzed sales data for two product categories in the well-known ERIM database [18]: ketchup and peanut butter. We used both situational and demographic variables as input (independent) variables, brand name as the target (dependent) variable. Thus, we included the following variables in our datasets:

1. Day of the week,
2. loyalty for all brands,
3. reference price for all brands,
4. prices of all brands,
5. display for all brands,
6. ad for all brands,
7. household income,
8. household size,
9. brand chosen (target or dependent variable).

In total, we use $3 + 5K$ input features (independent variables), where K is the number of brands/products.

Loyalty l_t^{hk} measures the loyalty of a specific household h towards a specific brand k at time t, and thus captures historical purchase behavior. It is computed by exponentially smoothing past purchases y_t^{hk} (1 if brand k is bought, 0 else) for each household h and brand k at purchase occasion t (as proposed in [19]):

$$l_t^{hk} = \alpha l_{t-1}^{hk} + (1-\alpha) y_{t-1}^{hk},$$

where α is a smoothing parameter for which we used the value 0.15.

The reference price [36, 27] represents the expectation household h has of the price of brand k at shopping occasion t. We compute this by exponentially smoothing prices for all brands during previous shopping occasions of household h with smoothing parameter $\alpha = 0.15$.

Following [23], we only incorporated households making at least 10 purchases of the selected brands. The first 4 purchases of each household were used to initialize the reference price- and loyalty variables.

In the case of the ketchup database, we included the three most frequently sold brands of the most frequently bought package size (32000 grams) in our analyses: Heinz ketchup, Hunt's ketchup pls&gls, Del Monte catsup. These three brands have a 55% market share including all other package sizes and a 83% market share within the 32000 grams segment. In the case of the peanut butter data, we used the 6 most frequently sold brands for the most frequently sold package size (18000 grams): Peter Pan crm h, Skippy crm h, Jif crm h, Peter Pan chk h, Skippy chk super h, Jif chk h. These six brands have a 41% market share including all other package sizes and a 67% market share within the 18000 grams segment. Statistics on both datasets are given in Table 1. We have the impression that the resulting data are rather noisy, but we did not attempt to improve data quality. The resulting datasets can be downloaded from `http://www.few.eur.nl/few/people/mvanwezel/ecc.html`.

Peanut Butter (6841 purchases)	
Categorical	
dayofweek	1:761 2:875 3:755 4:745 5:935 6:1200 7:1570
choice.of.brand (target)	B1:2177 B2:1042 B3:1203 B4:1087 B5:742 B6:590
Numerical	
loyaltybrand1	0 0 0.0190 0.3177 0.8530 1
reference.price.brand1	100.4 155.3 167 167.3 175.9 278.2
pricebrand1	95.92 159 169.98 171.17 181 290.00
displaybrand1	0 0 0 0.06841 0 1
adbrand1	0 0 0 0.1156 0 1
loyaltybrand2	0 0 0 0.1518 0.0220 1
reference.price.brand2	109.1 163.1 169.1 172.7 176.7 287.6
pricebrand2	106.3 163.6 171.6 175.6 180.4 296.0
displaybrand2	0 0 0 0.03552 0 1
adbrand2	0 0 0 0.08434 0 1
loyaltybrand3	0 0 0 0.1783 0.1280 1
reference.price.brand3	108.8 167.6 173.5 177 178.8 230.4
pricebrand3	106.3 169 175 180.2 189 266.6
displaybrand3	0 0 0 0.01038 0 1
adbrand3	0 0 0 0.03040 0 1
loyaltybrand4	0 0 0 0.1594 0.1280 1
reference.price.brand4	101.7 158.3 168.7 170.1 177.7 512.7
pricebrand4	98.61 159 173.43 173.42 185 521.00
displaybrand4	0 0 0 0.0671 0 1
adbrand4	0 0 0 0.1126 0 1
loyaltybrand5	0 0 0 0.1066 0 1
reference.price.brand5	108.5 162 169.2 173.2 177.3 580.5
pricebrand5	106.3 163 172 175.9 180.5 590
displaybrand5	0 0 0 0.03567 0 1
adbrand5	0 0 0 0.08376 0 1
loyaltybrand6	0 0 0 0.08627 0 1
reference.price.brand6	108.7 167.7 173.7 178.5 180.4 1008.1
pricebrand6	106.3 169 175 181.7 189 1109.0
displaybrand6	0 0 0 0.01359 0 1
adbrand6	0 0 0 0.03040 0 1
householdincome	1 5 6 6.373 8 14
numberofmembersinhousehold	1 2 4 3.517 4 8
Ketchup (7714 purchases)	
Categorical	
dayofweek	1:891 2:776 3:755 4:906 5:1178 6:1417 7:1791
choice.of.brand (target)	brand1:4765 brand2:1751 brand3:1198
Numerical	
loyaltybrand1	0 0.0220 0.8700 0.6062 1 1
reference.price.brand1	64.84 106.35 115.98 116.24 125.51 186.54
pricebrand1	55.53 101.70 115.56 114.81 119.63 193.80
displaybrand1	0 0 0 0.0757 0 1
adbrand1	0 0 0 0.2633 1 1
loyaltybrand2	0 0 0.0030 0.2367 0.1500 1
reference.price.brand2	66.20 98.88 109.73 114.17 123.77 313.24
pricebrand2	61.33 99 109 114.14 124.18 349.16
displaybrand2	0 0 0 0.0805 0 1
adbrand2	0 0 0 0.09891 0 1
loyaltybrand3	0 0 0 0.1572 0.1280 1
reference.price.brand3	67.99 97.03 109 115.05 133.09 528.53
pricebrand3	50 93.48 109 114.22 130.50 597
displaybrand3	0 0 0 0.04926 0 1
adbrand3	0 0 0 0.0914 0 1
householdincome	0 5 6 6.153 8 14
numberofmembersinhousehold	1 3 4 4.004 5 8

Table 1: Statistics on both datasets. For categorical variables the number of purchases per level is given, for numerical variables, the minimum, 1st quantile, median, mean, 3rd quantile and maximum, in that order.

4.2 Experimental setup

In our experiments we compare the performance on the above datasets for the following classifiers: a logistic regression model (lrg), CART, bagged versions of lrg and CART, boosted CART using LogitBoost, and finally multiboosted CART using wagging and LogitBoost. We used the public domain statistics software package `R` [33] for our experiments. `R` includes a package for CART trees, `rpart` – the ensemble methods were programmed by ourselves.

In our experiments we used 2-fold cross validation to obtain an estimate of the prediction error of the models. For each of the 2 folds we performed 25 runs. In each run, one classifier of each type was fitted using a bootstrapped version of the training data (1/2 of the total dataset). However, the first of these 25 datasets was taken to be the full (non-resampled) version of the training data, i.e., the full dataset excluding one fold.

The model-specific parameter values we used were as follows:

- In the CART-model we used complexity parameter $\texttt{cp} = 0$. This means that no regularization was used in building the CART tree. The `maxdepth` parameter, controlling the maximum tree depth, was set to 6. The `xval` parameter, controlling the number of cross-validation folds *within the training set* used in the pruning phase of the CART algorithm, was set to 10. These parameters were found to produce relatively good trees in a series of earlier experiments.

- In the bagged models, we used 49 bootstrapped replicates of the training part of the original dataset. We also included the full training part of the original dataset, making a total of 50 datasets. The base model parameters were identical to the case in which were they were separately applied.

- In the boosted models we used 50 iterations of the LogitBoost algorithm. The base model parameters were identical to the case were they were separately applied.

- In the experiments with the MultiBoost algorithm, we used 8 wagged datasets to each of which we applied 8 iterations of the LogitBoost algorithm.

4.3 Results

In this section we present the results of our analyses. We primarily look at the prediction errors we obtained. We purposely do not give the training errors for all methods, since a data analyst is primarily interested in out-of-sample performance of the models and a low training error gives no guarantee for a low test error.

Table 2 gives statistics on the cross-entropy and zero-one prediction errors for the ketchup and peanut butter datasets based on 25 runs. In each run, a bootstrap replicate of the dataset was created to build to model, as described in Section 4.2.

We first consider the results for the ketchup data. The best performing model here is the multiboosted CART model, with a an error rate of 21.60%, closely followed by the boosted CART model 21.77%. The bagged CART model also performs well (23.79%). The 'standard model', the logistic regression model, achieves a prediction error of 25.02% and thus the relative performance loss with respect to the multiboosted CART model is approximately 16%.

For the peanut butter data the bagged CART model performs best (43.88%). The multiboosted CART model (44.23%) performs slightly worse. The logistic regression model (45.85%) performs worse, but only by a 4.5% margin for this dataset. It is interesting to note that the application of ensemble learning has brought a substantial improvement for CART: individual CART models have an average prediction error of 53.34%, which is reduced down to 43.88% by bagging – A reduction of approximately 18%.

Note that many of the cross-entropy errors are unspecified (-). This is due to the problem with the cross-entropy error function we mentioned before: If the model incorrectly assigns probability 0 (or something close to 0) to the target class in one of the test patterns, the term corresponding

	lrg	bag lgr	CART	bag CART	boost CART	MultiBoost CART
			Cross entropy prediction errors			
			Ketchup			
mean	4827.94	4818.66	-	4775.07	25399.37	5530.97
sd	23.90	18.54	-	64.09	468.66	123.15
min	4773.73	4771.04	-	4674.08	24587.33	5008.88
max	4893.55	4851.43	-	4923.89	26383.30	5659.41
			Peanut butter			
mean	-	8770.89	-	-	45196.12	10982.26
sd	-	89.46	-	-	595.65	178.91
min	8870.26	8512.31	-	7876.27	43996.25	10382.69
max	-	8909.75	-	-	46174.20	11323.71
			Zero-one loss prediction errors			
			Ketchup			
mean	25.02	25.06	24.57	23.79	21.77	21.60
sd	0.19	0.19	0.58	0.56	0.28	0.28
min	24.63	24.67	23.79	23.14	21.09	20.87
max	25.33	25.49	25.95	25.10	22.21	22.21
			Peanut butter			
mean	45.85	45.82	51.86	43.88	44.92	44.23
sd	0.27	0.26	0.94	0.49	0.38	0.43
min	45.26	45.26	48.79	42.60	44.16	43.34
max	46.35	46.40	53.34	44.76	45.74	44.98

Table 2: Cross-entropy and zero-one error statistics on test data based on 25 runs. The test errors were obtained by 2-fold cross-validation.

to this test pattern in the cross-entropy function will be either undefined or cause an overflow. In these cases, we placed a - symbol in the corresponding table entry.

We tested the statistical significance of these results by means of the non-parametric Wilcoxon signed rank test. The results are shown in Table 3. Note from this table that ensemble learning always yields a significant improvement, except when applied to the logistic regression model. This is not surprising when we realize that ensemble learning creates a linear combination of 'base models'. Since a linear combination of linear models is still linear, ensemble learning is useless for linear base models. This issue will be discussed further in Section 5 below, when we discuss bias and variance.

Besides predictive performance, interpretability of a model is an important feature for the marketeer. The logistic regression model is interpreted in the usual way by studying its parameters. CART yields highly interpretable decision trees. Moreover, several useful tools exist for interpreting a fitted CART model [8]:

Relative importance plots, that visualize how important the various independent variables are relative to one another in predicting the dependent variable and,

partial dependence plots, that visualize the partial dependence of the implemented function on a subset of the independent variables.

Both the logistic regression parameters and two typical CART trees are shown in Appendix B. Unfortunately, our software did not facilitate the creation of relative importance plots and partial dependence plots, so we did not include them in this paper. (The emphasis in this paper is on performance, not interpretability.)

Interpretation of combined CART trees is not as straightforward as for a single CART tree. Luckily, relative importance plots and partial dependence plots are easily obtained for ensembles of CART trees as well by averaging over the individual models. See [16, 20]. Again, the software we used did not support making these graphs, so they are not included in this paper.

Model A	Model B	p	begin 95% ci	sign. at p=0.05?
		Ketchup		
lrg	baglrg	0.7232	-0.1425320	NO
lrg	CART	5.193e-05	0.3888138	YES
lrg	bagCART	1.276e-08	1.179635	YES
lrg	boostCART	7.034e-10	3.111195	YES
lrg	multiboostCART	6.996e-10	3.292745	YES
CART	bagCART	2.604e-05	0.557346	YES
CART	boostCART	7.053e-10	2.501971	YES
CART	multiboostCART	7.015e-10	2.683422	YES
		Peanut Butter		
lrg	baglrg	0.2898	-0.0877093	NO
lrg	CART	1	-6.417144	NO
lrg	bagCART	6.983e-10	1.754168	YES
lrg	boostCART	2.718e-09	0.7601356	YES
lrg	multiboostCART	6.99e-10	1.417870	YES
CART	bagCART	7.053e-10	7.674369	YES
CART	boostCART	7.053e-10	6.63648	YES
CART	multiboostCART	7.059e-10	7.338164	YES

Table 3: Outcomes of Wilcoxon signed rank test for the null hypothesis 'mean error Model A = mean error Model B' versus the alternative hypothesis 'mean error Model A > mean error Model B'. The column 'begin 95% ci' gives the improvement that Model B gives over Model A at the 5% confidence level.

It is interesting to investigate the so-called 'confusion matrices' yielded by the various classifiers in typical runs. These matrices depict the number of instances for each predicted brand / chosen brand combination, see Figure 5.

When we consider the confusion matrices for the peanut butter data we note that in general the within-category confusion among the creamy peanut butters (brands 1, 2 and 3, Peter Pan crm h, Skippy crm h, Jif crm h) and the chucked peanut butters (brands 4, 5, and 6, Peter Pan chk h, Skippy chk super h, Jif chk h) is smaller than the confusion between peanut butters of the two categories. However, the models are also likely to confuse two types of peanut butter from the same manufacturer, e.g., brands 2 (Skippy crm h) and 5 (Skippy chk super h) are more likely to be confused than e.g. brands 2 and 6 (Jif chk h). (We consider each product to be a separate brand here. Alternatively we might call two brands from one manufacturer two variants of a brand.)

Applying ensemble learning almost always improves the prediction error of the base learner, thus raising the elements the diagonal of the confusion matrix. If a diagonal element has increased, the sum of the other elements in a column must be lower since the total sum always equals the number of purchases of the brand represented by the column. One would expect all off diagonal elements to contribute to this reduction, but this not true in the case of the peanut butter data.

For example, when we consider the confusion matrices for CART and the multiboosted version of CART (which reduced the prediction error from 51.9% down to 44.2%) we see that the confusion between two brands from the same manufacturer has almost always increased. More precisely, the probability that the model correctly predicts the brand bought increases, but the probability that the model falsely predicts the other brand from the same manufacturer also increases! This is true for all but one pairs of related brands – Skippy crm h and Skippy chk super h, Jif crm h and Jif chk h, etcetera, except for one pair: Peter Pan crm h and Peter Pan chk h. Figure 6 shows the difference between the confusions of multiboosted CART and CART and thus indicates where the improved predictions (positive values on the diagonal) originate. Admittedly, this effect is less pronounced for the other ensemble methods, but is is still present.

Now, the question comes up for what type of individual purchases ensemble learning offers improvement. It turns out that these are to a large extent the purchases on which the individual CART models disagree. The theory behind this will be given in the next section, but it is instruc-

```
                          "Logistic regression"
     [B1] [B2] [B3]                            [B1] [B2] [B3] [B4] [B5] [B6]
[B1] 4219  618  380                       [B1] 1447  260  238  322   60   47
[B2]  334  944  171                       [B2]  191  476  111   50  127   23
[B3]  212  189  647                       [B3]  207  105  707   29   21   93
                                          [B4]  261   46   39  490  135  103
                                          [B5]   40  136   21  122  351   50
                                          [B6]   31   19   87   74   48  274

                          "Bagged logistic regression"
     [B1] [B2] [B3]                            [B1] [B2] [B3] [B4] [B5] [B6]
[B1] 4218  617  378                       [B1] 1448  264  234  324   61   46
[B2]  336  947  174                       [B2]  191  472  110   50  127   25
[B3]  211  187  646                       [B3]  208  107  714   30   20   91
                                          [B4]  261   44   35  483  134   99
                                          [B5]   40  137   23  125  351   52
                                          [B6]   29   18   87   75   49  277


                          "CART"
     [B1] [B2] [B3]                            [B1] [B2] [B3] [B4] [B5] [B6]
[B1] 4257  614  326                       [B1] 1395  266  275  318   88   71
[B2]  383  999  298                       [B2]  216  443  117   57  101   28
[B3]  125  138  574                       [B3]  218  119  668   56   25   88
                                          [B4]  270   76   60  427  151  107
                                          [B5]   49  115   28  152  330   64
                                          [B6]   29   23   55   77   47  232


                          "Bagged CART"
     [B1] [B2] [B3]                            [B1] [B2] [B3] [B4] [B5] [B6]
[B1] 4345  659  320                       [B1] 1662  301  328  374   92   80
[B2]  291  974  272                       [B2]  140  469   74   42   80   14
[B3]  129  118  606                       [B3]  143  109  722   35   23  105
                                          [B4]  184   32   23  448  128   97
                                          [B5]   32  120   21  141  386   54
                                          [B6]   16   11   35   47   33  240


                          "Boosted CART"
     [B1] [B2] [B3]                            [B1] [B2] [B3] [B4] [B5] [B6]
[B1] 4353  588  299                       [B1] 1567  258  283  363   81   64
[B2]  262 1004  169                       [B2]  144  485  102   51  115   18
[B3]  150  159  730                       [B3]  183  116  708   38   23  102
                                          [B4]  216   58   25  464  132  101
                                          [B5]   40  108   28  118  349   58
                                          [B6]   27   17   57   53   42  247


                          "Multiboosted CART"
     [B1] [B2] [B3]                            [B1] [B2] [B3] [B4] [B5] [B6]
[B1] 4336  558  281                       [B1] 1585  258  276  356   79   65
[B2]  275 1029  178                       [B2]  149  485   94   42  108   20
[B3]  154  164  739                       [B3]  183  109  724   36   22  101
                                          [B4]  206   49   21  458  133   96
                                          [B5]   33  124   29  136  362   65
                                          [B6]   21   17   59   59   38  243
```

Figure 5: Confusion matrices for ketchup data (l) and peanut butter data (r) for single runs of all model types. These matrices show the number of instances for each 'predicted brand' versus 'true brand' combination. The row indices represent the predicted brand, the column indices the true ('target') brand. The entries in the matrices always add up to the number of instances in the dataset. Each instance is present exactly once – it is added when part of the test fold.

```
     [B1] [B2] [B3] [B4] [B5] [B6]
[B1]  190   -8    1   38   -9   -6
[B2]  -67   42  -23  -15    7   -8
[B3]  -35  -10   56  -20   -3   13
[B4]  -64  -27  -39   31  -18  -11
[B5]  -16    9    1  -16   32    1
[B6]   -8   -6    4  -18   -9   11
```

Figure 6: Difference in confusions between CART and multiboosted CART.

	Proportion of CART predictions								
purchase	B1	B2	B3	B4	B5	B6	CART	Bagged CART	True
318	0	0	0.04	0.08	0.88	0	B5	B5	B1
319	0.2	0.24	0	0.28	0.28	0	B4	**B1**	B1
320	0.6	0.16	0.04	0.16	0.04	0	B1	B1	B4
321	0.68	0	0.08	0.2	0.04	0	B4	**B1**	B1
322	0.56	0.12	0	0.24	0.08	0	B2	**B1**	B1
323	0.56	0.16	0.08	0.16	0.04	0	B2	B1	B5
324	0	0	1	0	0	0	B3	B3	B3
325	0	0	1	0	0	0	B3	B3	B3
326	0.16	0	0.84	0	0	0	B3	B3	B3
327	0	0	1	0	0	0	B3	B3	B3
328	0	0	1	0	0	0	B3	B3	B3
329	0	0	1	0	0	0	B3	B3	B3
330	0	0	1	0	0	0	B3	B3	B1
331	0.2	0.08	0.64	0.08	0	0	B2	**B3**	B3
332	0.52	0.04	0.4	0	0	0.04	B6	**B3**	B3
333	0.08	0.04	0.84	0	0	0.04	B1	**B3**	B3
334	0.2	0	0.76	0	0	0.04	B3	B3	B3
335	0	0	1	0	0	0	B3	B3	B3
336	0	0	1	0	0	0	B3	B3	B3
337	0	0	1	0	0	0	B3	B3	B3
338	0	0	0.88	0	0	0.12	B6	**B3**	B3

Table 4: For what kind of purchases does bagged CART improve CART? In the left part of this table we see proportions of predicted brands over 25 individual CART models for a number of instances from the peanut butter dataset. In the first column of the right part of the table we see predictions generated by a single CART model, fitted on a bootstrapped version of the total dataset. In the middle column of the right hand part, we see the predicted brands according to a bagged CART model, and in the final column we see the brand that was purchased in reality (the target brand). For a number of instances, the bagged CART model corrects the error made by the individual CART model. (These predictions are typeset in bold.) These improvements tend to concern instances where there is a fair amount of disagreement among the 25 original CART models, as can be seen from the left hand part of the table. The quintessence here is that the application of ensemble learning reduces the error that results from the between-model ambiguity. We will consider this in more detail in Section 5. Ensemble learning improves CART specifically for purchases the individual CART models disagree on.

tive to consider a number of these purchases as an example. Table 4 shows a part of the peanut butter dataset and the improvements yielded by bagging a CART model. See the caption for an explanation.

5 Why ensemble learning works – Bias and variance

In this section we will attempt to explain why ensemble learning works, using bias/variance decompositions of the prediction error. We will illustrate these concepts using the same sales data we used in the preceding part of this paper. The theory that we summarize here is no new – was developed mainly by Breiman [5] and James [26]. It is the first time though, that this theory is applied to a brand choice problem.

As mentioned in Section 3, decision trees are examples of instable classifiers, where the implemented decision boundaries are highly dependent on the composition of the training set. (Another well-known example of an instable model is a neural network.) This instability contributes to the prediction error of the model. Now consider a linear model like logistic regression. This model is much less flexible than, e.g. a neural network, and consequently its variance is much lower. However, it may lack the flexibility to model the target function correctly. This 'stiffness', also called *bias*, may also contribute to the prediction error of the model.

The total error may thus be decomposed as follows:

$$\text{total error} = \text{irreducible error} + \text{bias} + \text{variance}. \tag{13}$$

The first term on the right hand side, the irreducible error, is due to the noise in the target variable. In classification, this is the Bayes error, in regression it is the intrinsic noise on the target. One can never improve a model by lowering this, but one can attempt to lower bias and variance. However, reducing the bias of a model by adding a degree of freedom usually increases the variance of the model. In minimizing the total error, a tradeoff has to be made between bias and variance.

Bias and variance are well known concepts for squared error loss (see, e.g., [1]), and were first applied to neural networks by Geman et. al. [17]. For other loss functions they have only recently been defined [5, 26, 21], and there still exists some controversy about the most appropriate definition. It is illustrative to consider the bias/variance decomposition for a squared error function before turning to general loss functions.

5.1 Bias and variance for squared error loss

We explain the bias/variance dilemma along the lines of and using the notation of [5]. Suppose a training set $T = \{(\mathbf{x}_n, y_n)|n = 1, \ldots, N\}$ is given where the $\mathbf{x}_n$ are multidimensional input vectors and y_n is the corresponding real valued target. y replaces the indicator vector $\mathbf{z}$ used earlier since squared error loss is usually used with regression. y and $f(\mathbf{x})$ are scalar, but generalization to multiple targets is straightforward. Let $f^*(\mathbf{x})$ denote $E[y|\mathbf{x}]$. The target variable y can be decomposed as

$$y = f^*(\mathbf{x}) + \epsilon,$$

where ϵ is a zero mean noise term. Suppose we want to fit a model f (e.g., a neural net with 3 hidden neurons, regression tree) on T in an attempt to find a relation $f_T(\mathbf{x})$ between target and input. The notation f_T stresses the dependence on the specific dataset T. It is important to distinguish between f, a general model class (e.g., a neural network with 3 hidden neurons), and f_T, a member of this model class fitted on T.

The prediction error of f_T is

$$\text{PE}(f_T) = E_{\mathbf{x},y}[(y - f_T(\mathbf{x}))^2].$$

Averaged over all possible training sets of size N, the prediction error of $f(\mathbf{x})$ is

$$\text{PE}(f) = E_T[\text{PE}(f_T)].$$

Denote $f^A(\mathbf{x}) = E_T[f_T(\mathbf{x})]$ – this is the predictor obtained by averaging predictors over all possible training sets. We can then decompose the prediction error PE(f) as follows

$$\begin{aligned}
\text{PE}(f) &= E_T[E_{\mathbf{x},y}[y - f_T(\mathbf{x})]^2] && (14)\\
&= E_T[E_{\mathbf{x},y}[y - f^A(\mathbf{x}) + f^A(\mathbf{x}) - f_T(\mathbf{x})]^2] && (15)\\
&= E_T[E_{\mathbf{x},y}[\{y - f^A(\mathbf{x})\}^2 + \{f^A(\mathbf{x}) - f_T(\mathbf{x})\}^2 + 2\{y - f^A(\mathbf{x})\}\{f^A(\mathbf{x}) - f_T(\mathbf{x})\}]]. && (16)
\end{aligned}$$

The last term in the square brackets vanishes because $E_T[f_T(\mathbf{x})] = f^A(\mathbf{x})$. The remaining expression is thus

$$\begin{aligned}
\text{PE}(f) &= E_T[E_{\mathbf{x},y}[\{y - f^A(\mathbf{x})\}^2 + \{f^A(\mathbf{x}) - f_T(\mathbf{x})\}^2]] && (17)\\
&= E_{\mathbf{x},y}[\{y - f^A(\mathbf{x})\}^2] + E_{T,\mathbf{x}}[\{f^A(\mathbf{x}) - f_T(\mathbf{x})\}^2] && (18)
\end{aligned}$$

the first term of which can be further decomposed in

$$\begin{aligned} E_{\mathbf{x},y}[\{y - f^A(\mathbf{x})\}^2] &= E_{\mathbf{x}}[\{f^*(\mathbf{x}) - f^A(\mathbf{x}) + \epsilon\}^2] && (19)\\ &= E_{\mathbf{x}}[\{f^*(\mathbf{x}) - f^A(\mathbf{x})\}^2 + \epsilon^2 + 2\epsilon(f^*(\mathbf{x}) - f^A(\mathbf{x}))] && (20)\\ &= E_{\mathbf{x}}[\{f^*(\mathbf{x}) - f^A(\mathbf{x})\}^2] + E[\epsilon^2]. && (21) \end{aligned}$$

If we define the following quantities:

$$\begin{aligned} \text{Bias}(f) &= E_{\mathbf{x}}[\{f^*(\mathbf{x}) - f^A(\mathbf{x})\}^2], && (22)\\ \text{Variance}(f) &= E_{T,\mathbf{x}}[\{f_T(\mathbf{x}) - f^A(\mathbf{x})\}^2], && (23) \end{aligned}$$

the prediction error PE(f) can be decomposed as in (13)

$$\begin{aligned} \text{PE}(f) &= E[\epsilon^2] + E_{\mathbf{x}}[\{f^*(\mathbf{x}) - f^A(\mathbf{x})\}^2] + E_{T,\mathbf{x}}[\{f_T(\mathbf{x}) - f^A(\mathbf{x})\}^2] && (24)\\ &= \text{irreducible error} + \text{Bias}(f) + \text{Variance}(f). && (25) \end{aligned}$$

As an illustration of bias and variance, consider Figure 7. Here, we see an ensemble of five neural networks[2] and five linear models trained on 100 random samples from the function $sin(x)/(1 + x^2) + \epsilon$, where ϵ is a noise term. The regression function $sin(x)/(1 + x^2)$ is shown as the bold red line. The functions implemented by the five neural networks and linear models are shown as the thin green and blue lines. The average models are shown as dotted blue and green lines. It is clear that the linear model exhibits a high bias – the dashed blue line is very distant from the regression – whereas the neural network has a low bias – the dashed green line is almost on top of the target function. On the other hand, the average distance of an individual neural network – a thin green line – to the regression is fairly large due to the variance of the model.

5.2 The bias/variance decomposition for general loss functions

Although the bias/variance decomposition for squared error loss is well known and easily obtained, the approach cannot be directly applied to other loss functions, such as polynomial loss, zero-one loss or cross entropy (5). The latter two are used in customer choice modeling. So, in order to study the bias and variance of customer choice models we need a more general definition of the bias/variance decomposition. Various authors have proposed bias/variance decompositions for other loss functions than squared loss (see e.g., [5, 15, 32, 38, 21, 10, 26]), usually with an emphasis on zero-one error loss.

Recently, James [26] published a review of earlier definitions and proposed a generalization of the bias/variance decomposition for general symmetric loss functions (i.e., where $L(a, b) = L(b, a)$). His generalized definitions satisfy the following desiderata (also given in [38, 21, 26]):

1. For squared error loss the generalized definitions should reduce to the original definitions,
2. The 'variance' must measure the variability of the estimator. Hence it must not depend on the distribution of the target variable but only on the distribution of the estimator itself. Furthermore it must be nonnegative and it must be zero if all estimators are equal.
3. The 'bias' must measure the difference between the 'average model' (f^A above) and the conditional expectation of the target.

James makes a distinction between the bias and variance of a model and their effects on the prediction error, the systematic effect and the variance effect. He proposes an additive decomposition of the prediction error into intrinsic noise, a systematic effect caused by the model's bias, and a variance effect caused by the model's variance.

[2]We use neural networks instead of decision trees here because this example was readily available to us and serves the purpose well.

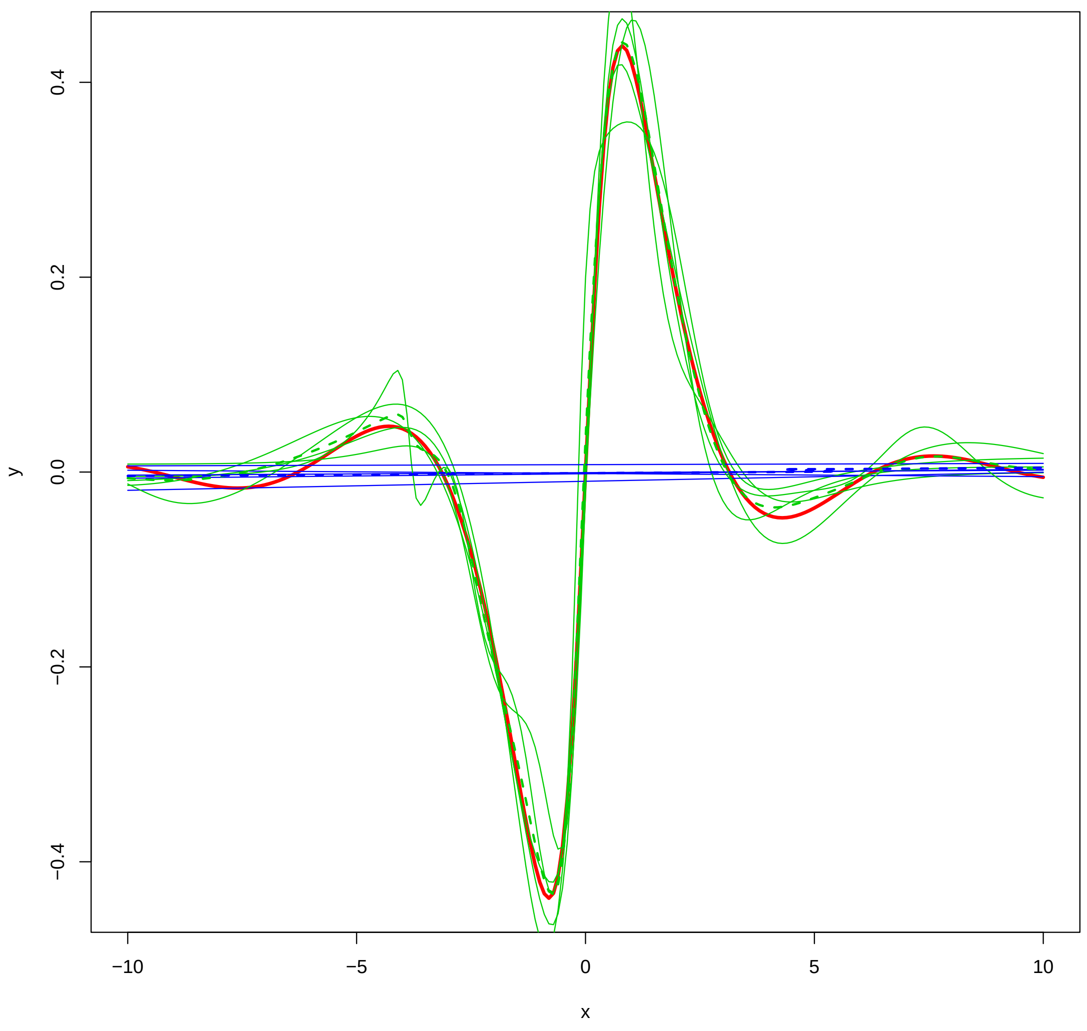

Figure 7: Illustration of bias and variance for in regression problem. See text for more explanation.

Tibshirani [32] earlier proposed an additive decomposition of the prediction error for various loss functions used in classification (i.e., zero-one loss, squared error and cross-entropy) that is identical to the decomposition proposed in [26]. Although James restricts the applicability of his decomposition to symmetric loss functions, Tibshirani shows that the decomposition can also be applied to cross-entropy.

In general terms the definitions of bias and variance and their effects proposed by Tibshirani and James are as follows. Let

$$f^A(\mathbf{x}) = \arg\min_{\mu} E_T[L(f_T(\mathbf{x}), \mu)], \tag{26}$$

denote the average model, and

$$f^*(\mathbf{x}) = \arg\min_{\mu} E_y[L(y, \mu)|\mathbf{x}], \tag{27}$$

denote the optimal model.

The bias and variance of a model are defined as

$$\text{Bias}(f) = E_{\mathbf{x}}[L(f^*(\mathbf{x}), f^A(\mathbf{x})], \tag{28}$$

$$\text{Variance}(f) = E_{\mathbf{x},T}[L(f_T(\mathbf{x}), f^A(\mathbf{x}))]. \tag{29}$$

These definitions provide an additive decomposition for squared error loss, but not for general loss functions. James defines the *systematic effect* SE and the *variance effect* VE of a model f as

$$\text{SE}(f) = E_{y,\mathbf{x}}\left[L(y, f^A(\mathbf{x})) - L(y, f^*(\mathbf{x}))\right], \tag{30}$$

$$\text{VE}(f) = E_{y,\mathbf{x},T}\left[L(y, f_T(\mathbf{x})) - L(y, f^A(\mathbf{x}))\right]. \tag{31}$$

Now, the total prediction error PE(f) of model f can be written

$$\begin{aligned}
\text{PE}(f) &= E_T[\text{PE}(f_T)] && (32)\\
&= E_{T,\mathbf{x},y}[L(f_T(\mathbf{x}), y)] && (33)\\
&= E_{\mathbf{x},y}[L(y, f^*(\mathbf{x}))] + \text{SE}(f) + \text{VE}(f) && (34)\\
&= \text{irreducible error} + \text{Systematic effect}(f) + \text{Variance effect}(f). && (35)
\end{aligned}$$

Some remarks about the systematic and variance effects are useful. For squared error loss the variance and systematic effects are equivalent to the standard definitions of bias and variance given above; The variance effect is 0 for the average model f^A; The bias effect is 0 for the optimal model f^*; For convex loss functions

$$E_{\mathbf{x},y}[L(y, f^A(\mathbf{x}))] \leq E_{T,\mathbf{x},y}[L(y, f_T(\mathbf{x}))] \tag{36}$$

by Jensen's inequality, so the variance effect is always positive. This means that the average model f^A has prediction error lower or equal[3] to the original model f; For non-convex loss functions (such as zero-one loss) the variance effect may be negative and the average model may have higher prediction error than the original model. For more details see [32].

There are a few problems in obtaining the above decomposition in practice. The first problem is that the average model $f^A(\mathbf{x})$ is unknown and has to be estimated somehow. Above, we explained how we estimate the average model by bagging and boosting in order to improve the prediction error.

The second problem is that, for real datasets, the intrinsic noise $E_{\mathbf{x},y}[L(y, f^*(\mathbf{x}))]$ is also unknown,[4] whereas it is required for the decomposition and the calculation of the systematic effect. This inherent noise can be estimated by examining the variability for neighboring input

[3]This does not mean that there is no dataset T' such that $f_{T'}$ has a lower prediction error than f^A.

[4]James [26] shows bias/variance decompositions for a number of synthetic classification datasets where the amount of intrinsic noise is known in order to illustrate his decomposition approach.

vectors, as in [26]. Alternatively, the intrinsic noise can be assumed to be zero, as in [10]. However, this leads to an overestimation of the systematic effect because any variability in the target variable is added to it. In the decompositions described below, we are mainly interested in the variance effect and do not care about overestimating the systematic effect, so we follow the latter approach. Note that when this approach is used and a large systematic effect is found when using a very flexible model, this is an indication that the data are very noisy.

Appendix A gives the bias/variance decompositions for the two classification loss functions that we use in this paper: the cross entropy loss function and the zero-one loss function.

5.3 Bias variance decompositions for customer choice models

We decomposed the prediction errors for the various models that were used in Section 4. In these decompositions, we made the unrealistic assumption that there is no intrinsic noise in the data, i.e. that $E_{\mathbf{x},y}[L(y, f^*(\mathbf{x}))] = 0$. This is no problem, however, since we are primarily interested in the *reduction* of the variance effect and the systematic effect as a result of using ensemble methods. The resulting decompositions are shown in Table 5.

Note that for the zero-one loss function there is no reduction in variance effect by using a bagged logistic regression model instead of a single logistic regression model. As was explained earlier, ensemble learning does not pay off for linear base models because a linear combination of linear models is still linear. Logistic regression thus suffers from a high systematic effect (bias) and a low variance.

For the ensemble versions of the CART model, on the other hand, we clearly see a reduction of the variance effect: it is reduced from 8.62 to 1.57. Also note that the systematic effect of the CART and bagged CART models is considerably lower than the systematic effect of the logistic regression models. Together, logistic regression and CART nicely illustrate the bias/variance tradeoff and how ensemble learning targets the variance component of the prediction error.

We would like to remark that the systematic effect we find is high for all model types. The most likely explanation for this is the inclusion of the intrinsic noise term in the systematic effect. By using better quality data and/or data from other domains, the intrinsic noise component might be reduced substantially. The absolute improvement yielded by ensemble learning would remain equal since it mainly reduces the variance effect. Thus, the relative improvement would increase substantially making ensemble learning look more favorable. Many applications of ensemble learning have been reported in the literature where the performance gain is in the order of 30% to 50% (see, e.g., [7]).

For the cross-entropy error function the decompositions are given for most models, but they cannot be compared with the decomposition for CART because the latter is missing due to numerical instability. Also, the cross-entropy errors for the ensemble CART models are greater than those for logistic regression, although the former models outperform the latter one in terms of zero-one loss. This inconsistency can also be attributed to the numerical instability of the cross-entropy loss function.

6 Summary, discussion and conclusions

In this paper we describe several ensemble learning methods from statistics and machine learning and applied them to customer choice modeling. Ensemble methods combine the predictions of several individual models into one combined prediction. The ensemble techniques we describe are 'bagging', 'boosting' and 'multi-boosting'. It is argued that these methods work best when they are applied to a 'base model' that is instable, i.e., for which the predictions depend heavily on the data it is fitted on or the initial parameter configuration.

Several experiments were carried out using sales data derived from the well known ERIM database, with logistic regression, CART and ensemble versions of these models. The data we used concerned two product categories: ketchup, for which purchase data on three brands was

	lrg	bag lrg	CART	bag CART	boost CART	MultiBoost CART
Cross entropy prediction errors						
	Ketchup					
PE	4827.94	4818.66	-	4775.07	25399.37	5530.97
SE	4773.16	4766.20	-	4673.98	11383.78	4502.64
VE	54.78	52.46	-	101.08	14015.59	1028.33
	Peanut butter					
PE	-	8770.89	-	-	45196.12	10982.26
SE	-	8443.52	-	-	17824.94	8267.72
VE	-	327.37	-	-	27371.18	2714.53
Zero-one loss prediction errors						
	Ketchup					
PE	25.02	25.06	24.57	23.79	21.77	21.60
SE	24.79	24.64	23.50	23.30	20.79	20.87
VE	0.23	0.42	1.07	0.49	0.98	0.73
	Peanut butter					
PE	45.85	45.82	51.86	43.88	44.92	44.23
SE	45.43	45.18	43.24	42.30	42.65	42.29
VE	0.41	0.64	8.62	1.57	2.27	1.95

Table 5: Bias-variance decompositions of the zero-one loss prediction errors reported in Table 2. PE, SE and VE mean prediction error, systematic effect and variance effect respectively.

analyzed, and peanut butter, for which purchase data on six brands was analyzed. CART together with ensemble learning always outperformed the other models.

As an explanation of the improvements, the bias/variance decomposition of the prediction error was considered. This decomposition splits the prediction error in three parts: intrinsic noise, systematic effect and variance effect. The first term is irreducible. The second term represents the part of the error that is due to the inflexibility of the model, and the last term is due to the sensitivity for the 'training data'. Using a more flexible model usually reduces component two, but increases component three. Ensemble models are believed to reduce component three, the variance effect, while leaving the other components unaltered. A bias/variance decomposition of the prediction errors obtained in the experiments was given.

It is useful to remark that the bias/variance decomposition may provide a tool for assessing the correctness of a parametric model. With a parametric model we mean a model that is derived from an underlying theory on a phenomenon. (An example of such a model is the multinomial logit model for customer choice modeling proposed in [29]. This model was designed based on a hypothesized nature of the decision process.) Such models tend to be more 'rigid', with fewer parameters, than general data-mining models such as CART and neural networks. However, if the bias/variance decomposition of such a parametric model reveals that it has a high bias (systematic effect) compared to a more flexible model, this is an indication that the model is not an accurate reflection of reality.

An interesting topic for further research to gain more insight into the problem domain of customer choices, is determining what types of purchases (which values for the marketing variables) lead to high variance among the models. In fact, the definitions of systematic effect and variance effect are easily narrowed to a specific part of the feature space, so this should be easy to implement.

The improvement in the results we obtained in this paper by the application of the ensemble methods is clearly noticeable, but not overwhelming. We feel that this may be due to the high amount of noise in the data we used for our experiments. Many applications of ensemble learning have been reported in the literature where the performance gain is in the order of 30% to 50% (see, e.g., [7]). It is possible that a greater effect of ensemble learning will be observed when using better quality data.

We would like to emphasize that the techniques we outlined in this paper are easy to implement and that their applicability is very broad, i.e. they can be used in many problem domains with many model types. In our opinion the marketing science community could benefit from the acquaintance with these methods.

References

[1] R. Bartoszynski and M. Niewiadomska-Bugaj. *Probability and Statistical Inference.* Wiley, New York, NY, 1996. ISBN 0-471-31073-5.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105–139, 1999.

[3] Y. Bentz and D. Merunka. Neural networks and the multinomial logit for brand choice modelling: a hybrid approach. *Journal of Forecasting*, 19(3):177–200, 2000.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, Berkeley, 1996.

[6] L. Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California, Berkeley, 1997.

[7] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26(3):801–849, 1998.

[8] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth and Brooks/Cole, Monterey, CA, 1984. ISBN 0412048418.

[9] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, Boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[10] P. Domingos. A unified bias-variance decomposition and its applications. In *Proceedings of the 17th International Conference on Machine Learning*, pages 231–238. Morgan Kaufmann, San Francisco, CA, 2000.

[11] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley & Sons, Inc., New York, 2nd edition, 2000. ISBN 0471056693.

[12] Philip Hans Franses and Richard Paap. *Quantitative Models in Marketing Research.* Cambridge University Press, 2001. ISBN 0521801664.

[13] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[14] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[15] J. H. Friedman. On Bias, Variance, 0/1 - loss, and the Curse-of-Dimensionality. *Data mining and Knowledge Discovery*, 1(1):54–77, 1997.

[16] J.H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[17] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[18] Marketing group. University of Chicago. Erim marketing database. Web-Site, 1997. `http://gsbwww.uchicago.edu/research/mkt/Databases/ERIM/ERIM.html`.

[19] P.M. Guadagni and J.D.C. Little. A logit model of brand choice calibrated on scanner data. *Marketing Science*, 2:203–238, 1983.

[20] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer series in statistics. Springer, 2001. ISBN 0387952845.

[21] T. Heskes. Bias/variance decompositions for likelihood-based estimators. *Neural Computation*, 10(6):1425–1433, 1998.

[22] S. Hettich. UCI machine learning repository. Web-Site, 2004. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, School of Information and Computer Science, University of California, Irvine.

[23] H. Hruschka, W. Fettes, M. Probst, and C. Mies. A flexible brand choice model based on neural net methodology - a comparison to the linear utility multinomial logit model and its latent class extension. *OR Spektrum*, 24(2):127–143, May 2002.

[24] M.Y. Hu and C. Tsoukalas. Explaining consumer choice through neural networks: The stacked generalization approach. *European Journal of Operational Research*, 146:650–660, 2003.

[25] R. Tibshirani J. Friedman, T. Hastie. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 28(2):337–407, 2000.

[26] G. M. James. Variance and bias for general loss functions. *Machine Learning*, 51(2):115–135, 2003.

[27] G. Kalyanaram and R.S. Winer. Empirical generalizations from reference price research. *Marketing Science*, 14:161–169, 1995.

[28] Gary L. Lilien and Arvind Rangaswamy. *Marketing Engineering: Computer-Assisted Marketing Analysis and Planning.* Prentice Hall, 2nd edition, 2002. ISBN: 0130355496.

[29] D. McFadden. Conditional logit analysis of qualitative choice behavior. In P. Zarembkla, editor, *Frontiers in econometrics*, pages 105–142. Academic Press, New York, 1973.

[30] R. Schapire. The boosting approach to machine learning: an overview. In *Proceedings of the 2002 MSRI Workshop on Nonlinear Estimation and Classification*, pages 149–173. Springer Verlag, 2003.

[31] H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

[32] R. Tibshirani. Bias, variance, & prediction error for classification rules. Technical report, Department of statistics, University of Toronto, 1996.

[33] W.N. Venables, D.M. Smith, and the R Development Core Team. *An introduction to R – Notes on R: A programming environment for data analysis and graphics.* The R Foundation for Statistical Computing, 1.6.2 edition, 2003. R is free software, available from `http://www.r-project.org/`.

[34] G. I. Webb. MultiBoosting: A technique for combining Boosting and Wagging. *Machine Learning*, 40(2):159–196, 2000.

[35] P.M. West, P.L. Brockett, and L.L. Golden. A comparative analysis of neural networks and statistical methods for predicting consumer choice. *Marketing Science*, 16(4):370–391, 1997.

[36] R.S. Winer. A reference price model of brand choice for frequently purchased products. *Journal of Consumer Research*, 13:250–256, 1986.

[37] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[38] D. Wolpert. On bias plus variance. *Neural Computation*, 9(6):1211–1243, 1997.

A Bias/variance decompositions for classification

In this appendix we give the bias/variance decompositions for two loss functions that are often used inclassification problems: the cross entropy loss function and the zero-one loss function.

A.1 The bias/variance decomposition for cross entropy

The additive decomposition (35) can be applied to the cross-entropy loss function (among others) in a straightforward manner. Recall that the loss function used in consumer response modeling is

$$L(\mathbf{z}, \mathbf{f}(\mathbf{x})) = -\sum_k z_k \log f_k(\mathbf{x}), \tag{37}$$

where the binary vector $\mathbf{z} = (z_1, \ldots, z_K)$ is the target vector and $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))$ are the class probabilities estimated by model $\mathbf{f}$. To stress $\mathbf{f}$'s dependence on training set T we write $\mathbf{f}_T(\mathbf{x}) = (f_{1|T}(\mathbf{x}), \ldots, f_{K|T}(\mathbf{x}))$. Because loss function (37) is convex, Jensen's inequality holds and aggregation can be expected to improve the prediction error.

The unknown average model, in this case given by

$$\mathbf{f}^A(\mathbf{x}) = \arg\min_{\boldsymbol{\mu}} E_T[\sum_k -f_{k|T}(\mathbf{x}) \log(\mu_k)], \tag{38}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\sum \mu_k = 1$. The constraint $\sum \mu_k = 1$ follows from the fact that the μ_k represent probabilities which have to sum to 1. In practice, the average model is estimated by an average over a number (say M) of different datasets $T_1, T_2, \ldots, T_M$ (Bootstrapped or re-weighted dataset in the case of bagging and boosting respectively.) Then, we have

$$\mathbf{f}^A(\mathbf{x}) \approx \arg\min_{\boldsymbol{\mu}} \sum_{t=1}^{M} \sum_{k=1}^{K} -f_{k|T_t}(\mathbf{x}) \log(\mu_k). \tag{39}$$

The Lagrangian for (39) is

$$\mathcal{L}(\boldsymbol{\mu}, \lambda) = \sum_k \log(\mu_k) \sum_t -f_{k|T_t}(\mathbf{x}) + \lambda(\sum_k \mu_k - 1), \tag{40}$$

so the solution lies at

$$\begin{pmatrix} \nabla\boldsymbol{\mu} \\ \nabla\lambda \end{pmatrix} \mathcal{L}(\boldsymbol{\mu}, \lambda) = \begin{pmatrix} -\sum_t f_{1|T_t}(\mathbf{x})/\mu_1 + \lambda \\ -\sum_t f_{2|T_t}(\mathbf{x})/\mu_2 + \lambda \\ \vdots \\ \sum_k \mu_k - 1 \end{pmatrix} = \mathbf{0}, \tag{41}$$

which corresponds to $\mu_k = \sum_t f_{k|T_t}(\mathbf{x})/\lambda$, where $\lambda = M$ the number of terms in the summation $\sum_t$. This can easily be seen by realizing that $\sum\limits_{t=1}^{M} \sum\limits_{k=1}^{K} f_{k|T_t}(\mathbf{x}) = M$, since the probabilities of the individual models sum to 1. Thus,

$$\lambda \sum_k \mu_k = \sum_t \sum_k f_{k|T_t}(\mathbf{x}),$$

which leads to $\lambda = M$. Summarizing, the average model $f_k^A(\mathbf{x})$ is just the average of the individual models $f_{k|T_1}(\mathbf{x}), f_{k|T_2}(\mathbf{x}), \ldots$.

The unknown 'optimal model' is defined as

$$\mathbf{f}^*(\mathbf{x}) = \arg\min_{\boldsymbol{\mu}} E_{\mathbf{z}}[L(\boldsymbol{\mu}, \mathbf{z})|\mathbf{x}]. \tag{42}$$

The error $E_{\mathbf{x},\mathbf{z}}[L(\mathbf{z}, f^*(\mathbf{x}))]$ of this classifier is irreducible and is caused by the 'noise' in the data.

Using these definitions it is straightforward to create a bias/variance decomposition using (30), (31) and (34).

A.2 The bias/variance decomposition for zero-one loss

For the zero-one loss function

$$L(\mathbf{z}, \mathbf{f}(\mathbf{x})) = I(\arg\max_i z_i, \arg\max_j f_j(\mathbf{x})) \tag{43}$$

with $I(a, b) = 1$ if $a \neq b, 0$ otherwise,[5] the average model is

$$\mathbf{f}^A(\mathbf{x}) = \arg\min_{\boldsymbol{\mu}} E_T[L(\boldsymbol{\mu}, \mathbf{f}_T(\mathbf{x}))]. \tag{44}$$

One solution lies at

$$\mu_k = \begin{cases} 1 & \text{if } k = \arg\min_i E_T[I(k_i, \arg\max_j f_{j|T}(\mathbf{x}))], \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

This corresponds to ‘voting’ among the base classifiers.

The unknown ‘optimal model’, or Bayes classifier, is defined as

$$\mathbf{f}^*(\mathbf{x}) = \arg\min_{\boldsymbol{\mu}} E_{\mathbf{z}}[L(\boldsymbol{\mu}, \mathbf{z})|\mathbf{x}]. \tag{46}$$

The error $E_{\mathbf{x},\mathbf{z}}[L(\mathbf{z}, \mathbf{f}^*(\mathbf{x}))]$ of this classifier is irreducible and is called the Bayes error rate.

Using these definitions it is straightforward to create a bias/variance decomposition using (30), (31) and (34).

B Fitted Models

[5]We slightly redefine the $0 - 1$ loss function to work with indicator vectors.

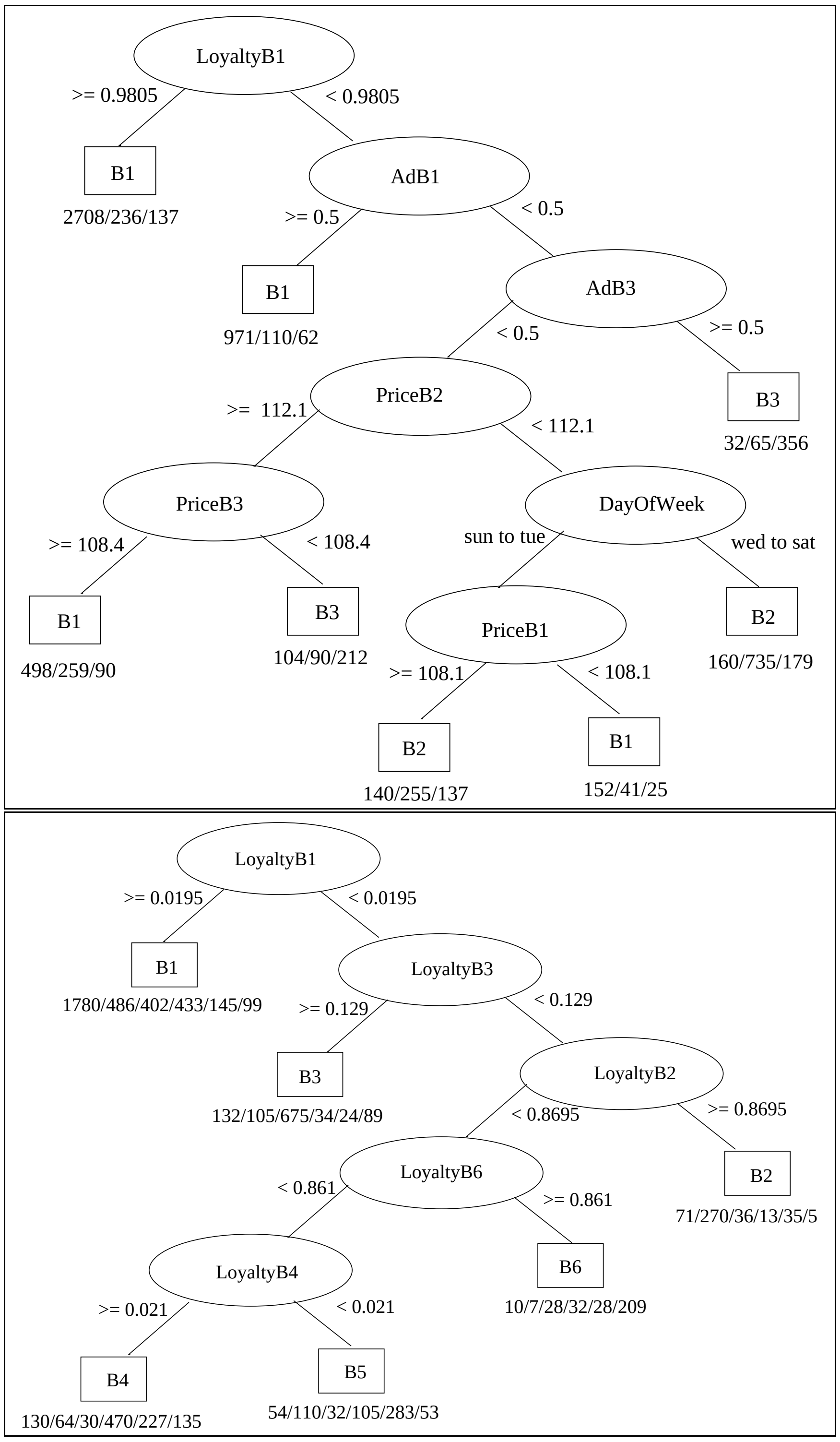

Figure 8: CART models for ketchup (top) and peanut butter (bottom).

	Heinz	Hunts	Del Monte
intercept	86.35	-150.33	63.37
dayofweek = 1	12.50	-21.68	8.97
dayofweek = 2	12.40	-21.82	9.02
dayofweek = 3	12.74	-21.43	9.14
dayofweek = 4	12.84	-21.14	9.39
dayofweek = 5	12.40	-21.36	9.06
dayofweek = 6	12.32	-21.57	8.77
dayofweek = 7	12.38	-21.56	9.04
loyalty brand1	-97.46	170.84	-74.02
reference price brand1	-0.27	-0.28	-0.28
price brand1	-0.15	-0.12	-0.11
display brand1	0.39	-0.15	0.01
ad brand1	1.19	-0.65	-0.87
loyalty brand2	-99.03	171.71	-73.95
reference price brand2	0.36	0.37	0.36
price brand2	0.09	0.06	0.09
display brand2	-0.26	0.56	-0.23
ad brand2	-0.69	0.67	-0.64
loyalty brand3	-99.01	171.19	-72.54
reference price brand3	0.13	0.13	0.13
price brand3	-0.23	-0.22	-0.26
display brand3	0.23	-0.36	0.70
ad brand3	-0.63	-0.39	1.30
household income	0.03	-0.02	0.00
number household members	-0.27	-0.21	-0.23

Table 6: Parameters for logistic regression model.

	B1	B2	B3	B4	B5	B6
intercept	0.33	0.47	-0.35	0.21	0.27	-0.70
dayofweek = 1	-0.28	0.08	-0.41	-0.04	-0.13	0.23
dayofweek = 2	-0.03	0.41	-0.55	0.09	0.23	-0.10
dayofweek = 3	0.19	0.35	-0.42	0.23	0.15	0.40
dayofweek = 4	0.01	0.33	0.02	0.06	0.43	0.43
dayofweek = 5	-0.01	0.10	-0.03	0.15	0.21	0.01
dayofweek = 6	0.07	0.33	-0.19	0.20	0.25	0.20
dayofweek = 7	0.02	0.20	-0.23	0.09	0.10	0.19
loyalty brand1	1.50	0.41	-0.04	0.10	-1.12	-1.48
reference price brand1	-0.08	-0.09	-0.09	-0.09	-0.09	-0.09
price brand1	-0.21	-0.17	-0.18	-0.18	-0.16	-0.17
display brand1	0.36	0.10	0.62	0.30	-0.41	0.36
ad brand1	0.55	0.20	0.46	0.48	-1.90	-0.06
loyalty brand2	0.42	1.75	0.23	-0.70	0.40	-1.08
reference price brand2	-0.28	-0.28	-0.30	-0.28	-0.27	-0.29
price brand2	0.03	0.00	0.04	0.03	0.00	0.04
display brand2	-0.30	0.89	0.05	0.72	0.01	-0.53
ad brand2	-0.61	-0.30	-1.05	0.03	1.24	-0.31
loyalty brand3	-0.05	0.07	2.14	-1.46	-1.67	0.16
reference price brand3	-0.07	-0.07	-0.05	-0.07	-0.08	-0.05
price brand3	0.05	0.05	0.02	0.06	0.06	0.02
display brand3	-1.98	-0.54	-0.57	4.41	-0.72	-0.53
ad brand3	-0.73	-0.31	0.64	-0.81	-0.12	0.98
loyalty brand4	0.15	-0.78	-1.50	1.25	0.44	0.15
reference price brand4	0.28	0.28	0.28	0.28	0.28	0.28
price brand4	0.01	0.01	0.01	-0.02	0.00	0.00
display brand4	0.47	0.06	-0.25	0.53	-0.21	-0.04
ad brand4	0.22	-0.60	-0.54	0.18	1.18	-0.29
loyalty brand5	-1.10	0.29	-1.60	0.39	1.71	-0.17
reference price brand5	0.05	0.04	0.05	0.04	0.05	0.05
price brand5	0.06	0.06	0.06	0.06	0.06	0.06
display brand5	0.35	-0.27	0.05	-0.73	0.69	-0.14
ad brand5	0.07	1.40	0.56	-0.45	-0.14	-0.88
loyalty brand6	-1.41	-1.23	0.41	0.19	0.09	2.13
reference price brand6	-0.02	-0.02	-0.02	-0.02	-0.02	-0.03
price brand6	-0.09	-0.08	-0.10	-0.09	-0.09	-0.08
display brand6	1.31	0.65	1.22	-4.09	-0.40	1.17
ad brand6	0.03	-0.43	0.14	-0.43	0.22	0.09
household income	0.04	0.04	0.01	0.09	0.11	0.12
number household members	-0.27	-0.26	-0.18	-0.30	-0.35	-0.35

Table 7: Parameters for logistic regression model (peanut butter data).