# Demand Estimation with Machine Learning and Model Combination

Patrick Bajari[1], Denis Nekipelov[2], Stephen P. Ryan[3], and Miaoyu Yang[4]

[1]University of Washington and NBER
[2]University of Virginia
[3]University of Texas at Austin and NBER
[4]University of Washington

February 4, 2015

**Abstract**

We survey and apply several techniques from the statistical and computer science literature to the problem of demand estimation. We derive novel asymptotic properties for several of these models. To improve out-of-sample prediction accuracy and obtain parametric rates of convergence, we propose a method of combining the underlying models via linear regression. Our method has several appealing features: it is robust to a large number of potentially-collinear regressors; it scales easily to very large data sets; the machine learning methods combine model selection and estimation; and the method can flexibly approximate arbitrary non-linear functions, even when the set of regressors is high dimensional and we also allow for fixed effects. We illustrate our method using a standard scanner panel data set to estimate promotional lift and find that our estimates are considerably more accurate in out of sample predictions of demand than some commonly used alternatives. While demand estimation is our motivating application, these methods are likely to be useful in other microeconometric problems.

# 1 Introduction

Over the past decade, there has been a high level of interest in modeling consumer behavior in the fields of computer science and statistics. These applications are motivated in part by the availability of large data sets where the demand for SKU's or individual consumers can be observed. These methods are commonly used in industry in retail, health care or on the internet by firms to use data at large scale to make more rational business decisions. In this paper, we compare these methods to standard econometric models that are used by practitioners to study demand. We are motivated by the problem of finding practical tools that would be of use to applied econometricians in estimating demand with large numbers of observations and covariates, such as in a scanner panel data set.

Many economists are unfamiliar with these methods, so we begin by expositing some commonly used techniques from the machine learning literature. We consider 8 different models that can be used for estimating demand for an SKU. The first two models are well known to applied econometricians—the conditional logit and a panel data regression model. We then turn to machine learning methods, all of which differ from standard approaches by combining an element of model selection into the estimation procedure. Several of these models can be seen as variants on regularization schemes, which reduce the number of covariates in a regression which receive non-zero coefficients, such as stepwise regression, forward stagewise regression, LASSO, and support vector machines. We also consider two models based on regression trees, which are flexible methods for approximating arbitrary functions: bagging and random forests. While these models may be unfamiliar to many economists, they are surprisingly simple and are based on underlying methods that will be quite familiar. Also, all of the methods that we use are supported in statistical packages. We perform our computations in the open source software package R. Therefore, application of these methods will not require writing complex code from scratch. However, applied econometricians may have to familiarize themselves with alternative software.

We derive novel results for the asymptotic theory for several of the models above. We show, somewhat unsurprisingly, that many of these models do not have standard asymptotics and converge more slowly than the standard square root rate. Since these models do not have standard normal asymptotics, common methods such as the bootstrap cannot be applied for inference. We also propose using an idea dating back at least to Bates and Granger (1969). We first form prediction for all 8 models in the standard way. We then treat each of these 8 independent predictions as regressors and form a combined model by regressing the dependent variable on to the prediction of each component model. We use a three-way cross

validation to avoid overfitting the models in practice. We split the sample into three disjoint sets; we use the first part to fit all 8 models, we use the second part to fit our regression on the 8 independent model predictions, and we use the final third of the data to test the fit out of sample.

To assess the small-sample properties of each of the machine learning methods and the model combination procedure we run a Monte Carlo. We show that several of the machine learning methods, particularly the random forest, do extremely well in predicting out-of-sample outcomes. In the combined model, the two regression tree-based methods, bagging and random forest, receive nearly fifty percent of weight. The model which receives forty-three percent of the weight in the combined model is the stagewise model. We demonstrate that the predictive accuracy of the combined model can result in dramatic improvements in fit over any of the component models. The Monte Carlo also demonstrates that the method is robust to the inclusion of spurious regressors, does just as well on complex, discontinuous data-generating processes as the simple linear-index models typically used in empirical work, and that the convergence rates of the various component models is typically better than the theoretical convergence rates.

We next apply our method to a canonical demand estimation problem. We use data from IRI Marketing Research via an academic license at the University of Chicago. It contains scanner panel data from grocery stores within one grocery store chain for six years. We used sales data on salty snacks, which is one of the categories provided in the IRI data. The number of observations are 1,510,563, which includes 3,149 unique products.

If we allow for product and store level fixed effects, our model effectively has many thousands of explanatory variables. Therefore, variable selection will be an important problem. If we included all of these variables in a standard regression model, the parameters would be poorly estimated. Also, many of the regressors will be multi-collinear which will make the models predict poorly out of sample.

In our results, we find that the 6 models we use from the statistics and computer science literature predict demand out of sample in standard metrics much more accurately than a panel data or logistic model. We do not claim that these models dominate all methods proposed in the voluminous demand estimation literature. Rather, we claim that as compared to common methods an applied econometrician might use in off the shelf statistical software, these methods are considerably more accurate. Also, the methods that we propose are all available in the well documented, open software package R as well as commercially-available software.

Applied econometricians have sometimes voiced skepticism about Machine Learning models because they do not have a clear interpretation and it is not obvious how to apply them to estimate causal effects. In this paper, we use an idea proposed by Varian (2014) to estimate the marketing lift attributable to promotions in our scanner panel. The idea is similar to the idea of synthetic controls used in Abadie and Gardeazabal (2003). We begin by training our model on the data where there is no promotion. We then hold the parameters of our model fixed and predict for the observations where there is a promotion. We then take the difference between the observed demand and the predicted demand for every observation in our data. By averaging over all observations in our sample, we construct an estimate of the average treatment effect on the treated.

We believe that this approach might be preferable to instrumental variable methods. In practice, it can be difficult to find instruments that are a prior plausible. When they exist, they may be subject to standard critiques such as the instruments may be weak or the identification may only be local.

The logic behind our approach is simply to use lots of data rather than rely on quasi-randomness. As mentioned above, in applied econometrics, there are many forms of data about products that are simply not exploited in empirical studies such as unstructured text or pictures. In some applications, simply using more data and more scalable computations may be a superior strategy to reducing bias in causal lift estimates.

We find quite interesting that a standard panel data model with fixed effects has the "wrong sign" on promotional lift, i.e. promotions decrease demand. By contrast, our models from the statistics and computer science literature have the anticipated sign. We conjecture that this is because they simply use more data and have less bias as suggested by standard omitted variable formulas.

Finally, we can use our model to search for heterogeneity in the treatment effects in an unstructured way. Our model generates a residual for each observation that is treated. We can regress this residual on covariates of interest such as store indicators, brand dummies, hedonic attributes or seasonal factors. Once again, this is a high dimensional regression problem and the estimates would be poorly estimated in a regression framework. We instead propose a method suggested by Belloni et al. (2012) and use a LASSO to select variable and then use standard methods for inference. We believe that this is attractive for applied econometricians since it allows us to learn about heterogeneity in the treatment effect. Also, it could be useful to applied marketers since these variable could be useful in marketing mix models because it allows us to identify a smaller set of variables that predict marketing

return.

The paper is organized as follows: Section 2 introduces the underlying statistical model of demand we study; Section 3 discusses the rates of convergence of our estimators and model combination; Section 4 demonstrates the useful properties of these estimators in a controlled Monte Carlo setting; Section 5 applies the techniques to a scanner data set; and Section 6 concludes.

# 2 Model

Our paper explores using machine learning techniques to help approximate the conditional expectation, $E[Y|X]$; to help motivate this more approximation more concretely, we study a standard model of discrete choice demand.

## 2.1 Components of Demand

Let there be $J$ products, each endowed with observable characteristics $X_j$. Let product $j$ have demand in market $m$ at time $t$ equal to:

$$\ln Q_{jhmt} = f(p_{mt}, a_{mt}, X_{mt}, D_{mt}, \epsilon_{jmt}), \tag{1}$$

where $a$ is a matrix of advertising and promotional measures, $D$ is a vector of demographics, $p$ is a vector of prices, and $\epsilon$ is an idiosyncratic shock.

The above specification is very general. It allows for nesting through the stratification of the error term. Suppose that there are $H$ nests of products; continuing the automobile example, two nests might be entry-level sub-compacts (Ford Fiesta, Toyota Yaris, Mazda 2, Chevrolet Sonic) and luxury performance sedans (BMW M3, Mercedes-Benz AMG C63, Cadillac CTS-V). Nests allow the substitution patterns to vary in a reduced-form way across those different classes of products. One can also extend the model to allow for non-trivial intertemporal shocks, such as seasonality in demand due to environmental conditions or holidays. The specification in Equation 1 is also consistent with models of discrete choice. For example, if the choice problem is discrete, then one obtains quantities by integrating over both the population of consumers and the distribution of errors.

The goal of our exercise is to estimate the relationships between the right-hand side variables and quantities demanded. We explore several approaches to approximating the model in Equation 1. We discuss common approaches such as linear regression and logit

models before describing several machine learning approaches which are less known in the econometrics literature.

### 2.1.1 Linear Regression

A typical approach to estimating demand would be to approximate Equation 1 using a flexible functional form of the following type:

$$\ln Q_{jhmt} = \alpha' p_{mt} + \beta_1' X_{mt} + \beta_2' D_{mt} + \gamma' a_{mt} + \lambda' \mathcal{I}(X_{mt}, D_{mt}, p_{mt}, a_{mt}) + \zeta_{hm} + \eta_{mt} + \epsilon_{jmt}, \quad (2)$$

where $\mathcal{I}$ is an operator which generates interactions between the observables (e.g. interactions of $X$, $p$, and $D$, e.g. high income neighborhoods have higher demand for expensive imported beer). Such a model may have thousands of right-hand side variables; for example, an online retailer such as eBay may offer hundreds of competing products in a category, such as men's dress shirts. The demand for one particular good, say white Brooks Brothers dress shirts, may depend on the prices of the full set of competing products offered. In a more extreme example, as offered in Rajaraman and Ullman (2011), Google estimates the demand for a given webpage by using a model of the network structure of literally billions of other webpages on the right-hand side. Dummy variables on nests are captured by $\zeta_{hm}$. Seasonality is captured by the term $\eta_{mt}$, which varies by time (say, quarters) across markets.

In ordinary least squares (OLS), the parameters of Equation 2, jointly denoted by $\beta$, are typically estimated using the closed-form formula:

$$\beta = (X'X)^{-1}(X'Y), \quad (3)$$

where $X$ is the matrix of right-hand side variables and $Y$ is the vector of outcomes.

We note that the formula requires an inversion of $(X'X)$. This imposes a rank and order condition on the matrix $X$. We highlight this because in many settings, the number of right-hand side variables can easily exceed the number of observations. Even in the simplest univariate model, one can saturate the right-hand side by using a series of basis function of $X$. This restriction requires the econometrician to make choices about which variables to include in the regression. We will return to this below, as some of machine learning methods we discuss below allow the econometrician to skirt the order condition by combining model selection and estimation simultaneously.

### 2.1.2 Logit Models

A large literature on differentiated products has largely focused on logit-type models, where the idiosyncratic error term is assumed to be distributed as a Type I Extreme Value. Under that restriction, market shares are given by:

$$s_{jhmt} = \frac{\exp(\theta' X_{jhmt})}{\sum_{k \in J} \exp(\theta' X_{khmt})}. \tag{4}$$

Starting with Berry et al. (1995), many empirical models of differentiated product demand concentrate on estimating a distribution of unobserved heterogeneity, $F(\theta)$, over a typically low-dimensional $\theta$:

$$s_{jhmt} = \int \frac{\exp(\theta' X_{jhmt})}{\sum_{k \in J} \exp(\theta' X_{khmt})} dF(\theta). \tag{5}$$

Quantities are then computed by multiplying through by market size. An attractive feature of the BLP-style approach is that the method is robust to the inclusion of unobserved heterogeneity and vertical characteristics that are observed to both the firm and consumers. However, this estimator places a significant amount of structure on the demand curve and is computationally burdensome.

### 2.1.3 Stepwise, Stagewise, and Incremental Stagewise Regression

Stepwise regression starts with the intercept as the base model. The algorithm then searches over the set of covariates, selects the one with the highest correlation with the residual, and adds that to the next model. The procedure produces a series of nested models. The procedure runs until no covariates have any a sufficient high correlation with the error term.

Forward stagewise regression is a variant of a variable selection model, with the regression model being built up one variable at a time. At each stage of the process, the econometrician finds the variable with the highest correlation to the residual. This variable is then added to the regression model; the crucial detail here is that the variable is only given an additional $\epsilon$, where $\epsilon$ is an arbitrarily small number. Over iterations, the forward stagewise regression builds up a model of predictors. This is related to the classic forward selection model if $\epsilon$ is set to equal the correlation of the variable with the residual (after standardizing the covariates and demeaning the response variable).

### 2.1.4 Support Vector Machines

Support vector machines are a penalized method of regression, using the following:

$$\min_{\beta} \sum_{i=1}^{n} V(y_i - x_i'\beta) + \frac{\lambda}{2}\|\beta\|, \tag{6}$$

where the penalty function is:

$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon & \text{otherwise.} \end{cases} \tag{7}$$

The tuning parameter, $\epsilon$, controls which errors are included in the regression. Errors of sufficiently small size are treated as zeros. Typically only a partial set of the covariates are assigned a non-zero value in SVM regression.

### 2.1.5 LASSO

Lasso is another penalized regression method. The regression is given by:

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda(t - \sum_{j=1}^{p} |\beta_j|), \tag{8}$$

where $t$ is the tuning parameter governing how strictly additional regressors are penalized. LASSO typically results in a number of covariates being given zero weights.

### 2.1.6 Regression Trees

A regression tree a collection of rules that determine the value of a function. Tree-based methods partition the characteristic space into a series of hyper-cubes, and fits a value to each partition. In a sense, they are a generalization of fixed effects, where the fixed effects value is allowed to depend on the $X$. Trees are characterized by a hierarchical series of nodes, with a decision rule associated at each node. Following ESL, define a pair of half-planes:

$$R_1(j, s) = X|X_j \leq s \text{ and } R_2(j, s) = X|X_j > s, \tag{9}$$

where $j$ indexes the *splitting variable* and $s$ is the *split point*. Starting with the base node at the top of the tree, the rule for that node is formed by the following optimization problem:

$$\min_{j,s} \left[ \min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]. \tag{10}$$

The inner minimization is solved by setting $c$ to be equal to the average of the function in that partition. The key issue here is finding the right split point, $s$. Once the split point has been found, the same procedure is then performed on each resulting partition, resulting in a further partitioning of the space.

In the limit, each value of $x \in X$ is assigned the value $Y = f(X = x)$, which is a perfect reconstruction of the underlying function $f$. In practice, the tree is expanded until the reduction in squared prediction error falls under some threshold. Often, the tree is grown until a specific number of splits are achieved.

The literature has proposed several variations on the regression tree estimator. One is *bagging* (Breiman, 1996), which uses resampling and model combination to obtain a predictor. The idea is to sample the data with replacement $B$ times, train a regression tree on each resampled set of data, and then predict the outcome at each $x$ through a simple average of the prediction under each of the $B$ trees.

A second approach, which we have found to work exceptionally well in practice, are *random forests*, as in Breiman (2001). Random forests expand on the idea of using collections of predictors to make predictions by introducing randomness into the set of variables which are considered at node level for splitting. Before each split, only $m \leq p$ of the explanatory variables are included in the split search. Repeating this across $B$ trees results in a forest of random trees. The regression predictor for the true function is then:

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x). \tag{11}$$

Trees of sufficient size can be unbiased but exhibit high variance, and therefore may benefit from averaging.

# 3 Model Averaging and Model Selection

The problem under consideration has a flavor of model selection problem with a large set of alternative models. The models are defined by the vectors of nonzero coefficients. The study of model selection has a long history. The main concepts of model selection contrast the selection criteria based on the measures of the (penalized) model fit and the criteria based on the model information. Themodel fit criteria include Mallows (1973) for generalized deviation-based goodness of fit, the prediction criterion of Amemiya (1980), and various information criteria, such as Akaike (1973), Schwarz (1978), Hannan and Quinn (1979), and Chow (1981). However, given that these procedures do not account for model complexity, they can lead to a selection of a miss-specified model. Shrinkage techniques have been proposed as a solution to the *pre-test problem*: see Stein (1956), James and Stein (1961),Yancey and Judge (1976) and Judge and Bock (1978), and associated algorithms such as Lasso (Tibshirani, 1996). One criticism of shrinkage is that it is not progressive, in the sense of knowledge accumulating about the process being modelled, because the decision rule need not eliminate variables. Bayesian model averaging (see Hoeting, Madigan, Raftery andVolinsky, 1999, for an overview) is often used to account for model uncertainty, as is the extreme bounds literature of Leamer (1978, 1983, 1985). Bayesian model averaging in its classical form, however, was heavily criticized by McAleer, Pagan and Volker (1985), Breusch (1990) and Hendry and Mizon (1990) among others. Stepwise regression is popular, but is path dependent, is susceptible to negative dependence, and does not have a high success rate of finding the DGP. Berk (1978) demonstrates that applying both forward and backward selection is no guarantee to finding the correct model. Alternative selection procedures exist, such as "optimal regression" in which all subsets of regressors are included (see Coen, Gomme and Kendall (1969) and Box and Newbold (1971)), but that approach is anyway intractable with a large set of potential regressors.

The discussed literature is mainly concerned with selecting a correctly specified model. However, it is not unfamiliar to choose a miss-specified model with much better predictive properties. There could be many other reasons why choosing the model with the correct specification is not the focus of the model selection procedure. Using criteria like the out-of-sample predictive accuracy and treaded-off goodness of fit and the model size is not uncommon. The proposed approach combines the Bayesian model averaging with the resent work on the least absolute shrinkage and selection operator and the Dantzig selector as well as the data mining literature.

We consider the model where the outcome variable $D$ is binary and it is driven by a vector

of regressors $X$. Our goal is to generate an approximation for the conditional expectation $E[D \mid X = x]$.

**Regression tree model** The regression tree model is based on the idea that the conditional expectation of interest can be approximated using the "flexible histogram approach": one would select the bins in the support of $X$ using an algorithmic selection rule and then the expectation will be approximated in each bin as an average of $D$ for all $X$ that fall into the bin. We start with a simple case and consider a situation where $X$ is one-dimensional. In most cases the split decision is based on the minimization of the least squares criterion. In other words, at each step the decision is made as to where to split the support of $X$. Consider the effect of the decision of the first split. At the first split we select a single point in the support of $X$ (we call it $x^{(1)}$) and the approximation for the conditional expectation takes the form

$$\widehat{\theta}^{(1)}(x) = \begin{cases} \theta_1^{(1)}, & \text{if } x < x^{(1)}, \\ \theta_2^{(1)}, & \text{if } x \geq x^{(1)}. \end{cases}$$

We call $\theta^{(1)} = (\theta_1^{(1)}, \theta_2^{(1)})'$. Provided that the decision is informed by the least squares criterion, we can write the sample objective function as

$$\widehat{Q}^{(1)}(x^{(1)}, \theta^{(1)}) = \frac{1}{n} \sum_{i=1}^{n} \left( d_i - \widehat{\theta}^{(1)}(x_i) \right)^2.$$

This objective is minimized with respect to $x^{(1)}$ and $\theta^{(1)}$. Suppose that $\widehat{x}^{(1)}$ is the minimizer of this objective function. Then

$$\widehat{\theta}_1^{(1)} = \frac{\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \leq \widehat{x}^{(1)}\}}{\sum_{i=1}^{n} \mathbf{1}\{x_i \leq \widehat{x}^{(1)}\}}$$

and

$$\widehat{\theta}_2^{(1)} = \frac{\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}}{\sum_{i=1}^{n} \mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}}.$$

Then the objective function can be re-written as

$$\widehat{Q}^{(1)}(x^{(1)}, \theta^{(1)}) = \frac{1}{n} \sum_{i=1}^{n} \left( d_i - \widehat{\theta}_1^{(1)} \right)^2 \mathbf{1}\{x_i \leq \widehat{x}^{(1)}\} + \frac{1}{n} \sum_{i=1}^{n} \left( d_i - \widehat{\theta}_2^{(1)} \right)^2 \mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}.$$

Further transformations yield

$$\widehat{Q}^{(1)}(\widehat{x}^{(1)},\widehat{\theta}^{(1)}) = \frac{1}{n}\sum_{i=1}^{n} d_i - \frac{\left(\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}\right)^2}{n\sum_{i=1}^{n}\mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}} - \frac{\left(\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \leq \widehat{x}^{(1)}\}\right)^2}{n\sum_{i=1}^{n}\mathbf{1}\{x_i \leq \widehat{x}^{(1)}\}}.$$

Now we note that the minimization of the criterion function can be replaced with the maximization of the following new objective function

$$\widehat{G}(\widehat{x}^{(1)}) = \frac{\left(\frac{1}{n}\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}\right)^2}{\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{x_i \geq \widehat{x}^{(1)}\}} + \frac{\left(\frac{1}{n}\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \leq \widehat{x}^{(1)}\}\right)^2}{\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{x_i \leq \widehat{x}^{(1)}\}}$$

with respect to the single parameter $\widehat{x}^{(1)}$. The presence of the indicator function leads the corresponding maximizer to behave similarly to the parameters in the Manski's maximum score model. To formalize the result we impose the following assumption.

**ASSUMPTION 1** *Suppose that*

*(i) Random variable $X$ has a continuously differentiable density. Moreover $E[|X|^{4+\delta}] < \infty$ for some $\delta > 0$*

*(ii) The conditional probability $P(D = 1|X = x)$ is continuous in $X$*

Denote the cdf of random variable $X$ by $F(\cdot)$. Consider the population version of the objective function which can be written as

$$G(x^{(1)}) = P(D = 1|X \leq x^{(1)})^2 F(x^{(1)}) + P(D = 1|X > x^{(1)})^2 (1 - F(x^{(1)})).$$

Now we bound the absolute difference $|\widehat{G}(\widehat{x}) - G(x^{(1)})|$ using the fact that $F(x^{(1)})(1 - F(x^{(1)})) \leq \frac{1}{2}$, $\frac{1}{n}\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \leq \widehat{x}^{(1)}\} \leq 1$ and the properties of empirical and true distribution functions:

$$|\widehat{G}(\widehat{x}) - G(x^{(1)})| \leq \left|\frac{1}{n}\sum_{i=1}^{n} d_i \mathbf{1}\{x_i \leq x^{(1)}\} - P(D = 1|X \leq x^{(1)})F(x^{(1)})\right| + \left|\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{x_i \geq x^{(1)}\} - F(x^{(1)})\right|.$$

As a result, the difference between the empirical and the true objective function is dominated by the sum of the difference between the true and the empirical cdf and the difference between the empirical and true correlation between $D$ and the indicator $\mathbf{1}\{X \leq x^{(1)}\}$. The latter is precisely the Manski maximum score objective function for estimation of the maximum

score model with normalization of the coefficients of the index $(1, X)$ to $(x^{(1)}, 1)$ which is one possible normalization in this model. This allows us to use the results of Kim and Pollard (1990) to establish the following result.

**THEOREM 1** *Suppose that Assumption 1 holds. Suppose that* $0 = argmax_x G(x)$ *(without loss of generality). Then if* $\widehat{x}^{(1)} = argmax_x \widehat{G}(x)$*, then*

$$n^{1/3}\widehat{x}^{(1)} \xrightarrow{d} \widehat{t},$$

*where random variable* $\widehat{t}$ *corresponds to* $\widehat{t} = argmax_t \left\{-t^2 G''(0) + W(t)\right\}$*, where* $W(t)$ *is a Brownian bridge process.*

This shows that the regression tree leads to the non-standard distribution for the cutoff points. At the same time, if we return to the analysis of the consistency of the regression function itself, we note that the cutoff point enters into the formula as a plug-in component and it is averaged. Newey (1994) and Brown and Newey (1996) demonstrated that in the analysis of the plug-in estimators which average over a functional of a non-parametric object, the estimation error of the plug-in nonparametric component only affects the quadratic second-order term. In other words, we can establish that for some $\delta_1$ we can represent

$$|\theta_1^{(1)} - \theta_1^{(1)}| = \left| \frac{\sum_{i=1}^n d_i \mathbf{1}\{x_i \leq x^{(1)}\}}{\sum_{i=1}^n \mathbf{1}\{x_i \leq x^{(1)}\}} - \theta_1^{(1)} \right| + \delta_1 (\widehat{x}^{(1)} - x^{(1)})^2 + o_p((\widehat{x}^{(1)} - x^{(1)})^2).$$

Provided that the standard Lindeberg CLT applies to the first term, we can establish that $\sqrt{n}|\theta_1^{(1)} - \theta_1^{(1)}| = O_p(1)$ if $|\widehat{x}^{(1)} - x^{(1)}| = o_p(n^{-1/4})$. In light of Theorem 2, provided that $|\widehat{x}^{(1)} - x^{(1)}| = O_p(n^{-1/3})$, it immediately follows that $\sqrt{n}|\theta_1^{(1)} - \theta_1^{(1)}| = O_p(1)$ and thus the estimator for the parameters in the regression tree model converges to the true coefficient at the standard parametric convergence rate.

These results allow a simple extension to a multi-dimensional case. In case of multiple regressors $X$, the tree model allows one to select the dimension of the variable $X$. Suppose that the dimension of $X$ is $d$ and consider the case where all components of $X$ are continuous. As it will become clear, the case where some components of $X$ are discrete come as a simple extension. By $X(r)$ we denote $r$-th component of $X$. The algorithm in the multi-dimensional case proceeds as follows: at iteration $k$ we consider each of $r$ dimensions. Suppose that by step $k$, $k_r$ splits were performed along dimension $k$. Then for each dimension $r$ we compute the outcome of $k_r + 1$-st split. The actual split will be performed along the dimension for which the $k_r + 1$-st split results in the largest decrease in the mean-squared error. Note that

if any component that is already discrete does not need splitting, the decision would only be whether including that variable decreases the sum of squared residuals more than splitting along other variables.

As before, without loss of generality we can consider the split decision in the first step. At the first split we select the split dimension $r$ and a single point in the support of $X$ (we call it $x^{(1)}(r)$) and the approximation for the conditional expectation takes the form

$$\widehat{\theta}^{(1)}(x) = \begin{cases} \theta_1^{(1)}, & \text{if } x(r) < x^{(1)}(r), \\ \theta_2^{(1)}, & \text{if } x(r) \geq x^{(1)}(r), \end{cases}$$

if dimension $r$ was chosen for a split. The sum of squared residuals can be again written as

$$\widehat{Q}^{(1)}(x^{(1)}, \theta^{(1)}) = \frac{1}{n} \sum_{i=1}^{n} \left( d_i - \widehat{\theta}^{(1)}(x_i) \right)^2.$$

Then the problem of the first step boils down to finding $d$ numbers $\widehat{x}^{(1)}(1), \ldots, \widehat{x}^{(1)}(d)$ and the dimension $r$ which minimizes

$$\widehat{Q}^{(1)}(\widehat{x}^{(1)}, \widehat{\theta}^{(1)}, r) = \frac{1}{n} \sum_{i=1}^{n} d_i - \frac{\left( \sum_{i=1}^{n} d_i \mathbf{1}\{x_i(r) \geq \widehat{x}^{(1)}(r)\} \right)^2}{n \sum_{i=1}^{n} \mathbf{1}\{x_i(r) \geq \widehat{x}^{(1)}(r)\}} - \frac{\left( \sum_{i=1}^{n} d_i \mathbf{1}\{x_i(r) \leq \widehat{x}^{(1)}(r)\} \right)^2}{n \sum_{i=1}^{n} \mathbf{1}\{x_i(r) \leq \widehat{x}^{(1)}(r)\}}.$$

As in the one-dimensional case, we can reduce the problem to the maximization of the objective function

$$\widehat{G}(\widehat{x}^{(1)}, r) = \frac{\left( \frac{1}{n} \sum_{i=1}^{n} d_i \mathbf{1}\{x_i(r) \geq \widehat{x}^{(1)}(r)\} \right)^2}{\frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{x_i(r) \geq \widehat{x}^{(1)}(r)\}} + \frac{\left( \frac{1}{n} \sum_{i=1}^{n} d_i \mathbf{1}\{x_i(r) \leq \widehat{x}^{(1)}(r)\} \right)^2}{\frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{x_i(r) \leq \widehat{x}^{(1)}(r)\}}$$

with respect to $d$ continuous and one discrete variable. Now we note that along each dimension, under our previous normalization we can apply the result of Kim and Pollard (1990) to get

$$\widehat{G}\left( x^{(1)} + \frac{t}{n^{1/3}}, r \right) \Rightarrow -t^2 G''(0, r) + W_r(t),$$

where $W(t) = (W_1(t), \ldots, W_d(t))$ is the $d$-dimensional Brownian bridge process. This delivers $n^{1/3}$ convergence rate for the split point for each dimension. We note that this rate applies to all split dimensions. Previously we have established that this ensures that the estimated regression $\theta^1(x)$ will converge at parametric $\sqrt{n}$ rate for the given number of splits. Now

since there are $d$ dimensions, we can majorize the error

$$|\theta_1^{(1)} - \theta_1^{(1)}| \leq d \sup_{r=1,\ldots,d} \left| \frac{\sum_{i=1}^n d_i \mathbf{1}\{x_i(r) \leq x^{(1)}(r)\}}{\sum_{i=1}^n \mathbf{1}\{x_i(r) \leq x^{(1)}(r)\}} - \theta_1^{(1)} \right| + \delta_1 \sum_{r=1}^d (\widehat{x}^{(1)}(r) - x^{(1)}(r))^2$$
$$+ o_p(\sum_{r=1}^d (\widehat{x}^{(1)}(r) - x^{(1)}(r))^2) = O_p\left(\frac{1}{\sqrt{n}}\right) + O_p\left(\frac{d}{n^{2/3}}\right).$$

This delivers the same $\sqrt{n}$ convergence rate for the estimated regression.

**Stepwise regression model** Consider the stepwise regression model based on the choice of the subset of regressors $X$, $X^2$, $X^3$, etc. by a successive enlargement of the model. For simiplicity of the exposition, as in the case of the regression tree we will characterize the properties of the one-dimensional regressor. Suppose that we use the AIC-style criterion for the choice of the selection of regressors. In this case the selection criterion can be characterized by

$$\widehat{Q}(\widehat{\beta}^{(p)}, p) = \frac{1}{n} \sum_{i=1}^n \left( d_i - \sum_{k=0}^p \beta_k^{(p)} x_i^p \right)^2 - \lambda\, p,$$

which is minimized with respect to the vector of coefficients at each $p$ and then the decision is maed on the basis of whether the criterion function increases in the transition from $p$ to $p+1$.

Let $x_i^{(p)} = (1, x_i, x_i^2, \ldots, x_i^p)'$. Then the fitted value of the estimated projection of $d$ on the polynomial temrs can be expressed for each $p$ as $\widehat{\beta}^{(p)} = \left( \frac{1}{n} \sum_{i=1}^n x_i^{(p)} x_i^{(p)\prime} \right)^{-1} \frac{1}{n} \sum_{i=1}^n x_i^{(p)} d_i$. Further, we can convert the problem of selection of $p$ into the problem of solving for a continuous parameter $\tau$ such that $p = [n^{1/3}\tau]$ where $[\cdot]$ denotes the integer part of a real number.

Now we consider the objective function

$$\widehat{G}(\tau) = \widehat{Q}(\widehat{\beta}^{([n^{1/3}\tau])}, [n^{1/3}\tau])$$

with the population analog $G(\tau)$.

It turns out that the optimization problem formulated in this form leads to a similar cube-root convergence result for the parameter $\tau$.

**THEOREM 2** *Suppose that Assumption 1 holds. Suppose that* $1 = argmax_{t \geq 0} G(t)$ *(without*

*loss of generality). Then if* $\widehat{\tau} = argmax_{t \geq 0} \widehat{G}(t)$*, then*

$$n^{1/3}(\widehat{\tau} - 1) \xrightarrow{d} \widehat{t},$$

*where random variable* $\widehat{t}$ *corresponds to* $\widehat{t} = argmax_t \left\{ -t^2 G''(1) + W(t) \right\}$*, where* $W(t)$ *is a Brownian bridge process.*

## Random forest model

The random forest model is based on the the aggregation of the ensamble of $B$ trees. We have previously shown that for the distribution of the first split of the regression tree model we can evaluate the variance term as

$$|\widehat{\theta}^{(1)} - \theta^{(1)}| = O_p(1/\sqrt{n}).$$

This result will also apply to any subsequent split. In other words, this means that the variance term of the mean-squared error for the tree of length $k$ is

$$|\widehat{\theta}^{(k)} - \theta^{(k)}| = O_p(1/\sqrt{n}).$$

The bias will be determined by the location of the split points, which we found to have the following structure:

$$|\widehat{x}^{(1)} - x^{(1)}| = O_p(1/\sqrt[3]{n}).$$

The same evaluation with also apply to the $k$-th split. Now looking at the random forest with a fixed number of $B$ trees with a fixed number of splits $k$ yields the overall evaluation for the mean-squared error of $O_p(1/\sqrt[3]{n})$ dominated by the bias of the split points. Note that this is the lower bound for the convergence rate of the random forest. Data-dependent choice of the number of trees and the number of splits may further accelerate the convergence rate (see Scornet (2014)).

## Support vector machines

Note that the SVM estimator for the regression with a fixed margin $\epsilon$ (see Vapnik (1998)) leads to an objective function with the structure

$$nQ(\beta;\epsilon,\lambda)=\sum_{i=1}^{n}(y_i-x_i'\beta-\epsilon)\mathbf{1}\left\{y_i-x_i'\beta-\epsilon\leq 0\right\}+\frac{\lambda}{2}\|\beta\|.$$

Assuming that the model dimension $d$ is finite and fixed and the parameter space is compact, then $\frac{\lambda}{2}\|\beta\|=O(d)$. This means that

$$Q(\beta;\epsilon,\lambda)=\frac{1}{n}\sum_{i=1}^{n}(y_i-x_i'\beta-\epsilon)\mathbf{1}\left\{y_i-x_i'\beta-\epsilon\leq 0\right\}+O(d/n).$$

The first term in this objective function has a structure of the objective function that parallels the Manski's maximum score objective function. This fact was first noticed in Komarova (2013). In Kim and Pollard (1990) it was noticed that such a structure of the objective function leads to the following evaluation

$$|\widehat{\beta}-\beta|=O_p(\frac{1}{\sqrt[3]{n}}).$$

This means that the predictions from the SVM regression have the mean square error with the lower bound $O_p(1/\sqrt[3]{n})$. Making the margin $\epsilon$ and the penalty constant $\lambda$ data-dependent may further accelerate the convergence.

## LASSO

The general theory for LASSO regression has been developed in Belloni et al. (2011). They impose the restriction on the structure of regressors and consider the case where the dimensionality of the model $d$ can grow along with the number of non-zero coefficients. The establish the bound on the mean-squared error of the vector of estimated coefficients of $O_p(\sqrt{s\log d/n})$. This means that with a moderate growth of sparsity $s=O(n^{1/3})$, we obtain the $O_p(1/\sqrt[3]{n})$ mean-squared error for prediction in the model with a fixed dimensionality. With a more stringent set of conditions for the sparsity and the dimensionality, the rate can be further improved (as a function of $n$).

# Improvement of convergence rates for averaged classifiers

The object of interest in the "first stage" estimation was to recover the function $\theta(\cdot)$. We now consider the properties of the second stage estimation whose goal is the weighted average value of the classifier which may be described as

$$\alpha = E[w(X)\theta(X)]. \tag{12}$$

We consider the case where $\theta(\cdot)$ is delivered by some algorithmic model selection rule (from the list analyzed in the previous subsection) and $w(\cdot)$ are either directly imputed or estimated. Newey (1994) provides an insight that the asymptotic properties of an empirical analog of (12) may not depend on the properties of a particular non-parametric estimator which was employed to estimate $\theta(\cdot)$. This result provides a powerful implication for our estimation approach. We proved that an algorithmic model selection rule provides a consistent estimator for $\theta(\cdot)$ that, however, converges at the subparametric $n^{1/3}$ rate to a non-standard distribution characterized by a Brownian bridge. If the result in Newey (1994) can be applied, this would mean that the properties of an averaged classifier will be standard and thus be immune to the choice of the algorithmic model selection rule used for estimation of $\theta(\cdot)$.

First of all, consider the empirical analog of $\alpha$ where $\theta(\cdot)$ and $w(\cdot)$ are replaced with their empirical counterparts $\widehat{\theta}(\cdot)$ and $\widehat{w}(\cdot)$:

$$\widehat{\alpha} = \frac{1}{n}\sum_{i=1}^{n} \widehat{w}(x_i)\widehat{\theta}(x_i). \tag{13}$$

We assume (for the purposes of normalization) that $E[\widehat{w}(X)] = E[w(X)] = 0$. Without loss of generality, this assumption can be omitted. Now consider the following decomposition

$$\begin{aligned}
\widehat{\alpha} - \alpha \;&= \tfrac{1}{n}\sum_{i=1}^{n} \widehat{w}(x_i)\widehat{\theta}(x_i) - E[w(X)\theta(X)] \\
&= \tfrac{1}{n}\sum_{i=1}^{n} (\widehat{w}(x_i) - w(x_i))(\widehat{\theta}(x_i) - \theta(x_i) - E[\widehat{\theta}(X)] + E[\theta(X)]) \\
&+ \tfrac{1}{n}\sum_{i=1}^{n} w(x_i)(\widehat{\theta}(x_i) - \theta(x_i) - E[\widehat{\theta}(X)] + E[\theta(X)]) \\
&+ \tfrac{1}{n}\sum_{i=1}^{n} (\theta(x_i) - E[\theta(X)])\,(\widehat{w}(x_i) - w(x_i)) + \tfrac{1}{n}\sum_{i=1}^{n} (E[\theta(X)] - E[\theta(X)])\,(\widehat{w}(x_i) - w(x_i)) \\
&\tfrac{1}{n}\sum_{i=1}^{n} E[\theta(X)](\widehat{w} - w(x_i)) + \tfrac{1}{n}\sum_{i=1}^{n} (w(x_i)\theta(x_i) - E[w(X)\theta(X)])
\end{aligned}$$

We have established the result that for the algorithmic model selection rule we can provide a uniform convergence rate $\|\widehat{\theta}(x) - \theta(x) - E[\widehat{\theta}(X)] + E[\theta(X)]\|_\infty = O_p(n^{-1/3})$. Also, the bias condition reduces to $|E[\widehat{\theta}(X)] - E[\theta(X)]| = o_p(n^{-1/3})$. Now suppose that $\widehat{w}(\cdot)$ is estimated at the uniform rate $n^{-r}$. We now establish the requirement for this rate that guaratees the parametric convergence rate for $\widehat{\alpha}$. First of all note that under the i.i.d. sampling requirement and restriction that $E\left[w^{2+\delta}(X)\theta^{2+\delta}(x_i)\right]$ we can apply the Lindeberg CLT to the last term of the above decomposition. This means that the last term converges in distribution to the normal random variable at parametric rate. Now consider the same decomposition for $\sqrt{n}(\widehat{\alpha} - \alpha)$. We evaluate each of the terms of decomposition multiplied by $\sqrt{n}$:

$$\begin{aligned}
\sqrt{n}\,(\widehat{\alpha} - \alpha) &= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (\widehat{w}(x_i) - w(x_i))(\widehat{\theta}(x_i) - \theta(x_i) - E[\widehat{\theta}(X)] + E[\theta(X)]) \\
&+ \frac{1}{\sqrt{n}} \sum_{i=1}^{n} w(x_i)(\widehat{\theta}(x_i) - \theta(x_i) - E[\widehat{\theta}(X)] + E[\theta(X)]) \\
&+ \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (\theta(x_i) - E[\theta(X)])\,(\widehat{w}(x_i) - w(x_i)) \\
&- \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (E[\theta(X)] - E[\theta(X)])\,(\widehat{w}(x_i) - w(x_i)) \\
&+ \frac{1}{\sqrt{n}} \sum_{i=1}^{n} E[\theta(X)])(\widehat{w}(x_i) - w(x_i)) + \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (w(x_i)\theta(x_i) - E[w(X)\theta(X)]) \\
&= O_p(n^{1/6-r}) + O_p(n^{-1/3}) + O_p(n^{-r}) + o_p(n^{1/6-r}) + O_p(1) + O_p(1).
\end{aligned} \tag{14}$$

The first evaluation comes from the fact that

$$\begin{aligned}
&\left\| \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (\widehat{w}(x_i) - w(x_i))(\widehat{\theta}(x_i) - \theta(x_i) - E[\widehat{\theta}(X)] + E[\theta(X)]) \right\| \\
&\qquad \leq \sqrt{n} \|\widehat{w}(x) - w(x)\|_\infty \|\widehat{\theta}(x) - \theta(x) - E[\widehat{\theta}(X)] + E[\theta(X)\|_\infty = O_p(n^{1/6-r}).
\end{aligned} \tag{15}$$

The second evaluation comes from the fact that

$$\begin{aligned}
\tfrac{1}{\sqrt{n}} \sum_{i=1}^{n} \quad & w(x_i)(\widehat{\theta}(x_i) - \theta(x_i) - E[\widehat{\theta}(X)] + E[\theta(X)]) \\
&\leq \|\widehat{\theta}(x) - \theta(x) - E[\widehat{\theta}(X)] + E[\theta(X)\|_\infty \tfrac{1}{\sqrt{n}} \sum_{i=1}^{n} w(x_i) = O_p(n^{-1/3}),
\end{aligned}$$

provided that $E[w^{2+\delta}(X)] < \infty$ and thus CLT can be applied to the mean of $w(\cdot)$. Analogously, we can evaluate the next term, provided that $E[\theta^{2+\delta}(X)] < \infty$:

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \left(\theta(x_i) - E[\theta(X)]\right) \left(\widehat{w}(x_i) - w(x_i)\right) \leq \|\widehat{w}(x) - w(x)\|_\infty \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \theta(x_i) = O_p(n^{-r}).$$

Then

$$\begin{aligned} \tfrac{1}{\sqrt{n}} \sum_{i=1}^{n} \left(E[\theta(X)] - E[\theta(X)]\right) \left(\widehat{w}(x_i) - w(x_i)\right) &\leq \sqrt{n} \|\widehat{w}(x) - w(x)\|_\infty |E[\theta(X)] - E[\theta(X)]| \\ &= O_p(n^{1/6-r}). \end{aligned}$$

The last two terms of the expansion will be leading and, thus, deliver the standard parametric asymptotics if the first three terms are negligible.

**THEOREM 3** *Suppose that* $n^{1/6} \sup_x \|\widehat{w}(x) - w(x)\| \xrightarrow{p} 0$. *Then for some* $V_\alpha < \infty$*:*

$$\sqrt{n} \left(\widehat{\alpha} - \alpha\right) \xrightarrow{d} N(0,\, V_\alpha).$$

*In particular, if the weights are fixed then the convergence of the averaged algorithmic classifier to its expectation at parametric rate is guarateed regardless of the method used to obtain the classifier.*

*Proof:* Note that if $n^{1/6} \sup_x \|\widehat{w}(x) - w(x)\| \xrightarrow{p} 0$, then our previous analysis establishes that

$$\sqrt{n} \left(\widehat{\alpha} - \alpha\right) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \psi(x_i) + o_p(1),$$

where

$$\psi(x_i) = (w(x_i)\theta(x_i) - E[w(X)\theta(X)]) + E[\theta(X)](\widehat{w}(x_i) - w(x_i)).$$

This means that the conditions of Proposition 2 in Newey (1994) are satisfied and, thus, the parameter of interest $\widehat{\alpha}$ converges at the parametric rate to the normal asymptotic distribution whose properties are immune to the choice of estimator for $\widehat{\theta}(\cdot)$ selected from the class of algorithmic model selection estimators considered in the previous section. *Q.E.D.*

Note that even in case the weights $\widehat{w}(\cdot)$ we need a very weak requirement for the convergence of these weights to their population counterparts at the rate $n^{-1/6}$ which allows one

to use a wide range of non-parametric methods for construction of such weights to perform the averaging of the prediction from the algorithmic model selection rule.

# 4 Monte Carlo

To demonstrate the usefulness of our estimation approach, we conduct several Monte Carlo exercises. We consider several variations of a discrete choice model: we use both aggregate and individual data; compare the logit model to our machine learning methods with and without specification error; show the performance of our methods in a model with irrelevant regressors; and finally consider the performance of the logit and machine learning methods on a model with a highly nonlinear utility function. We also report the convergence rates of the individual estimators and demonstrate the superior statistical properties of the linear combination model.

We first consider the classic case of a discrete choice model using aggregate data. In each of $M$ markets, there are $J = 2$ products with $K = 2$ characteristics each. For each product, characteristics are drawn from the following distributions:

$$x_1 \sim logN(0, 1), x_2 \sim logN(0, 1.5)$$

In each market, we assume there are an infinite number of consumers. Each consumer makes a single choice from the set of options which maximizes their utility. Utilities are given by

$$u_{ij} = u(X_j, \beta) + \epsilon_{ij}, \tag{16}$$

where $X_j$ is a matrix of products and their characteristics, $\beta$ is a vector of utility parameters, and $\epsilon_{ij}$ is an individual- and choice-specific idiosyncratic error with a Type I extreme value distribution. This model generates shares (or choice probabilities in the case of individual data) of the form:

$$s_j = \frac{\exp(u(X_j, \beta))}{\sum_k \exp(u(X_k, \beta))}. \tag{17}$$

We consider several variants of Equation 16. In the baseline model, utility is linear in the unknown parameters:

$$u_{ij} = x_j\beta + \epsilon_{ij}. \tag{18}$$

We set the marginal utilities to be equal to $\beta = \{0.1, -0.02\}$. This is the standard specification used in most discrete choice applications, and will provide a natural baseline for

comparing the performance of the flexible approximation methods to more traditional approaches.

To test the performance of our estimators against more complex functional forms, we also consider a model with a highly-nonlinear utility function given by:

$$10 * \sin\left((0.03X_{j1} + 0.1X_{j2})^2\right) - 0.5\ln(X_{j2}) - 1(0.3 < X_{j1} < 1.5)6X_{j1} + 1(X_{j1} > 2)X_{j1} + 1(0.15 < X_{j2} < 1)X_{j2} + 1(X_{j2} > 2.3)X_{j2}. \quad (19)$$

To test our methods in the presence of irrelevant regressors, we consider a version of Equation 16 with five additional "junk" data regressors that receive zero marginal utilities. The first two are drawn from a standard log-normal distribution, the third is the sine function of a uniform random variable, the fourth is a "time trend" type variable which counts up from 1 for each observation. The fifth is similar to the fourth but adds a squared term to the count, specifically $x = 0.5n + 0.2n^2$ where $n$ is the observation number.

We also consider a micro-data version of the model above, with market-level data replaced by individual-level observations. This specification introduces sampling error in that there are a finite number of consumers in each market, given by $N$.

In order to assess the performance of our approach, we consider a three-step process. First, using a training sample of size $M$, we estimate the parameters of the component models. Then, using a second out-of-sample training data set, we estimate weights on the best linear predictor of 2,500 observed market shares. Third, we then take the combined linear model and use it to predict an additional 2,500 out-of-sample market shares. We report the results of the in-sample fit from stage one, the weights from stage two, and the out-of-sample fits from stage three in our results.

We now describe the specific algorithms and parameters choices for the various models that are used in the Monte Carlo.

**Bagging** The steps taken are as follows:

1. Generate shares and characteristics data as described above for $M$ markets.
2. Sample with replacement from this data 200 times.
3. For each subsample, train a regression tree.
4. Predict the market shares of test data by averaging the prediction across all grown trees.

**Random Forest** The steps taken are the same as described in the bagging section, except that when trees are grown, at each node, 2 characteristics are chosen at random to be candidate characteristics for splitting into new leaves.

**Linear Regression** This model regresses product 1's market shares on all product characteristics and their 2nd and 3rd powers, including all interactions between polynomials (e.g, $x_1^2 x_2$ and $x_1 x_2^2$).

**Logit** This model conducts a logistic regression of product 1's market shares on all product characteristics. It fits perfectly, so for now is not depicted in the figures below, and does not factor into the best linear combination model.

**Lasso** This model conducts a Lasso regression, which is OLS (the same model with up to 3rd order polynomials) with an added constraint that the sum of estimation coefficients is less than some threshold parameter. This is equivalently written as a Lagrangian multiplier which equally penalizes any positive estimator values. Matlab represents the Lasso penalty parameter in this way. To choose a threshold parameter for each market size, we conduct the Lasso regression 100 times over a grid of possible parameter values, and then choose the penalty parameter (lambda) that minimizes the RMSE in the first out-of-sample test data.

**Stepwise Regression** This model implements Matlab's *stepwisefit* function. The initial feature set is null, and the model successively adds a regressor to the model with the lowest coefficient p-values, until the p-values surpass 0.05. The resulting coefficient vector includes the coefficients from the selected features, with zeros elsewhere.

**Stagewise Regression** We implement a Stagewise Regression algorithm, which works as follows. Add a constant to the matrix of $X$ (including the polynomials as above).

1. Set $\beta = 0$.
2. Fit a constant equal to $\min(Y)$ and compute the residual ($\epsilon$).
3. Step through each column of $X$ and compute $\text{corr}(X_j, \epsilon)$.
4. Choose the $X_j$ with the highest correlation and regress $X_j$ on $\epsilon$.
5. Add the resulting coefficient to $\beta_j$.

Table 1: Monte Carlo: Baseline Model

| Market Size | 400 | | | 6400 | | | 51,200 | | |
|---|---|---|---|---|---|---|---|---|---|
| MODEL | RMSE (IS) | RMSE | Weight | RMSE (IS) | RMSE | Weight | RMSE (IS) | RMSE | Weight |
| | | | | Aggregate | | | | | |
| Bagging | 0.0124 | 0.0328 | 0.1136 | 0.014 | 0.011 | 0.3655 | 0.0117 | 0.0025 | 0.2768 |
| Rand Forest | 0.0125 | 0.0326 | 0 | 0.0143 | 0.0107 | 0.0676 | 0.0116 | 0.0025 | 0.2689 |
| Linear | 0.0215 | 1.5187 | 0 | 0.0761 | 0.3239 | 0.0056 | 0.1184 | 0.0259 | 0.018 |
| Lasso | 0.0902 | 0.2244 | 0 | 0.2485 | 0.1548 | 0 | 0.3162 | 0.07 | 0 |
| Stepwise | 0.0907 | 0.2268 | 0.0187 | 0.2492 | 0.1558 | 0.0178 | 0.3183 | 0.0703 | 0.0047 |
| Stagewise | 0.0027 | 0.019 | 0.8677 | 0.0154 | 0.0561 | 0.1365 | 0.0124 | 0.0025 | 0.4317 |
| IF Stagewise | 0.0648 | 1.2199 | 0 | 0.0152 | 0.0149 | 0.407 | 0.0195 | 0.0041 | 0 |
| Best Linear | | 0.0171 | | | 0.0088 | | | 0.0017 | |
| | | | | Individual | | | | | |
| Bagging | 0.116 | 0.2993 | 0 | 0.3107 | 0.2024 | 0 | 0.3998 | 0.0903 | 0 |
| Rand Forest | 0.1171 | 0.2985 | 0.9904 | 0.3135 | 0.2024 | 0.1386 | 0.4019 | 0.0903 | 0.1065 |
| Logit | 0.1665 | 0.3118 | 0 | 0.446 | 0.2861 | 0 | 0.5623 | 0.1155 | 0.0242 |
| Linear | 0.1139 | 3.2092 | 0 | 0.3304 | 0.2056 | 0.1037 | 0.4176 | 0.0913 | 0.2123 |
| Lasso | 0.128 | 0.683 | 0.0041 | 0.3741 | 0.2345 | 0.0535 | 0.472 | 0.1034 | 0 |
| Stepwise | 0.1516 | 0.6201 | 0.0055 | 0.3736 | 0.236 | 0 | 0.4756 | 0.1048 | 0 |
| Stagewise | 0.1151 | 0.3167 | 0 | 0.327 | 0.202 | 0.6778 | 0.412 | 0.0903 | 0.657 |
| IF Stagewise | 0.1268 | 39.1139 | 0 | 0.3275 | 0.2022 | 0.0264 | 1.2317 | 2.2218 | 0 |
| Best Linear | | 0.2984 | | | 0.202 | | | 0.0903 | |

6. Update $\epsilon = Y - X'\beta$.

7. Looping back to step 3, repeat until either the maximum correlation is less than 0.01 or 5,000 iterations have passed.

**Incremental Forward Stagewise Regression** This model is the same as the Stagewise Regression except that in step 5, we update the coefficient by only moving a small distance $\delta$ towards the estimated coefficient.

**Best Linear Combination** In this method, we find an optimal linear combination of the models using linear regression (constrained so that weights are within [0,1] and sum to 1). The optimum is chosen with respect to the first out-of-sample test data, and then reported with respect to the second out-of-sample test data. The thinking here is to prevent over-fitting the weights to the training data. This method is similar to choosing weights to minimize cross-validation error and then reporting out-of-sample fits. Then the RMSE is calculated between the market shares and the weighted predictions of the two models.

## 4.1 Results

We first consider the baseline model with aggregate shares. The top panel of Table 1 reports the significant variance within and across sample sizes with respect to in- and out-of-sample fit. Bagging and the random forest do well in all sample sizes, and stagewise regression has the best fit at the smallest sample size. Linear regression suffers from tremendous overfitting, as it has decent in-sample fits and terrible out-of-sample fit. At the higher sample sizes, random forest, bagging, and stagewise regression dominate the in-sample and out-of-sample fits compared to the rest of the models. This is also reflected in the weights that these three estimators receive in the linear combination. The RMSE for the combined model is 32 percent better than any of the individual components.

The bottom panel of Table 1 reports the results for the individual-level model. The essential difference between the two settings is the idiosyncratic error which is absent in the aggregate model but plays a key role in the individual-level data. Broadly speaking, each of the individual models, along with the linear combination, have errors which are two orders of magnitude larger than the aggregate models. We have included the logit model in the bottom panel; interestingly, although it is the most efficient estimator, it does dominate the machine learning methods for in- or out-of-sample fit. It also does not receive any weights in the linear combination for the smaller samples sizes, and only contributes two percent in the largest sample. Stagewise again receives a large share of the weights in the linear combination, while bagging and random forest share stagewise regression's level of prediction performance.

Table 2 reports the results for the discontinuous utility function, which should be more of a challenge for both the econometrician and estimation. The econometrician would likely not guess the functional form of the underlying data-generating process. Estimation is going to be more difficult given the discontinuous nonlinearity of the underlying utility function.

## 4.2 Rates of Error Convergence

Tables 5 shows the ratio of standard errors for each doubling of the sample size from $n = 100$ to $n = 102,400$. This exercise is useful for demonstrating the empirical convergence rates of our various component models and also the combination model. Looking at the average rates from Table 5, it is useful to compare these ratios against the parametric rate, which is $\sqrt{2} \simeq 1.4142$, and the cube-root rate, which is $\sqrt[3]{2} \simeq 1.2599$. The average rates for bagging and the random forest are slightly below the parametric rate, but substantially better than the cube-root lower bound provided by theory. The polynomial, stagewise, and incremental

Table 2: Monte Carlo: Discontinuous Utility

| Market Size | | 400 | | | 6400 | | | 51,200 | |
|---|---|---|---|---|---|---|---|---|---|
| MODEL | RMSE (IS) | RMSE | Weight | RMSE (IS) | RMSE | Weight | RMSE (IS) | RMSE | Weight |
| | | | | Aggregate | | | | | |
| Bagging | 0.0757 | 0.2031 | 0.8773 | 0.1284 | 0.0832 | 0.4848 | 0.116 | 0.0262 | 0.6455 |
| Rand Forest | 0.0772 | 0.2082 | 0.1096 | 0.1287 | 0.0842 | 0.5152 | 0.1153 | 0.0261 | 0.3545 |
| Logit | 0.1115 | 0.2884 | 0 | 0.3548 | 0.2217 | 0 | 0.4455 | 0.0981 | 0 |
| Linear | 0.0818 | 54.6065 | 0 | 0.2727 | 0.1733 | 0 | 0.3566 | 0.1716 | 0 |
| Lasso | 0.1187 | 1.7018 | 0 | 0.3617 | 0.2261 | 0 | 0.4473 | 0.1967 | 0 |
| Stepwise | 0.1514 | 40.126 | 0.0033 | 0.3812 | 0.2371 | 0 | 0.4464 | 0.1972 | 0 |
| Stagewise | 0.0906 | 5.2107 | 0.0098 | 0.2739 | 0.1717 | 0 | 0.37 | 0.0854 | 0 |
| IF Stagewise | 0.1025 | 0.2848 | 0 | 0.3022 | 0.1881 | 0 | 0.3839 | 0.0865 | 0 |
| Best Linear | | 0.2188 | | | 0.0828 | | | 0.0257 | |
| | | | | Individual | | | | | |
| Bagging | 0.1082 | 0.2772 | 0.9209 | 0.2779 | 0.1817 | 0.5016 | 0.3564 | 0.0811 | 0.9244 |
| Rand Forest | 0.1095 | 0.2801 | 0 | 0.2808 | 0.1817 | 0.4386 | 0.359 | 0.0813 | 0.0756 |
| Logit | 0.1296 | 0.3064 | 0.0485 | 0.3991 | 0.2415 | 0.0429 | 0.6125 | 0.1364 | 0 |
| Linear | 0.1077 | 6.8276 | 0 | 0.3089 | 0.2391 | 0.0009 | 0.3968 | 0.0878 | 0 |
| Lasso | 0.1388 | 0.344 | 0 | 0.3678 | 0.2324 | 0 | 0.4576 | 0.1021 | 0 |
| Stepwise | 0.1484 | 7.9849 | 0 | 0.3906 | 0.2421 | 0.016 | 0.4598 | 0.1026 | 0 |
| Stagewise | 0.113 | 0.3125 | 0.0306 | 0.3127 | 0.1939 | 0 | 0.3965 | 0.0881 | 0 |
| IF Stagewise | 0.1195 | 0.4755 | 0 | 0.4278 | 0.2445 | 0 | 0.4029 | 0.0898 | 0 |
| Best Linear | | 0.2771 | | | 0.1813 | | | 0.0811 | |

Table 3: Monte Carlo: Irrelevant Regressors

| Market Size | | 400 | | | 6400 | | | 51,200 | |
|---|---|---|---|---|---|---|---|---|---|
| MODEL | RMSE (IS) | RMSE | Weight | RMSE (IS) | RMSE | Weight | RMSE (IS) | RMSE | Weight |
| | | | | Aggregate | | | | | |
| Bagging | 0.0116 | 0.0287 | 0.0466 | 0.0134 | 0.0085 | 0.2328 | 0.0116 | 0.002 | 0.4452 |
| Rand Forest | 0.0125 | 0.0309 | 0 | 0.0164 | 0.0101 | 0 | 0.0153 | 0.0029 | 0 |
| Logit | 0.043 | 0.1065 | 0 | 0.1186 | 0.0732 | 0 | 0.1516 | 0.0332 | 0 |
| Linear | 0.0112 | 0.4661 | 0.0019 | 0.0408 | 0.058 | 0.0137 | 0.0614 | 0.013 | 0.035 |
| Lasso | 0.0896 | 0.2238 | 0 | 0.2485 | 0.1549 | 0 | 0.3163 | 0.0699 | 0 |
| Stepwise | 0.0909 | 0.2274 | 0.0273 | 0.2499 | 0.1562 | 0.0286 | 0.3168 | 0.07 | 0.0107 |
| Stagewise | 0.0042 | 0.0128 | 0.9133 | 0.01 | 0.0063 | 0.149 | 0.0135 | 0.0029 | 0.5091 |
| IF Stagewise | 0.0104 | 0.0394 | 0.0109 | 0.0129 | 0.0082 | 0.576 | 0.023 | 0.0047 | 0 |
| Best Linear | | 0.011 | | | 0.0044 | | | 0.0016 | |
| | | | | Individual | | | | | |
| Bagging | 0.1086 | 0.3043 | 0 | 0.3072 | 0.2013 | 0.3436 | 0.3979 | 0.0905 | 0 |
| Rand Forest | 0.1104 | 0.303 | 0.6239 | 0.312 | 0.2009 | 0 | 0.4015 | 0.0905 | 0.6356 |
| Logit | 0.1181 | 0.3059 | 0.1295 | 0.3281 | 0.2007 | 0.2215 | 0.4142 | 0.0911 | 0.1083 |
| Linear | 0.0987 | 0.5509 | 0 | 0.3238 | 0.203 | 0.2011 | 0.412 | 0.0908 | 0.2561 |
| Lasso | 0.0997 | 0.4604 | 0 | 0.3752 | 0.2292 | 0.0079 | 0.4707 | 0.103 | 0 |
| Stepwise | 0.1527 | 0.4436 | 0 | 0.4077 | 0.2497 | 0 | 0.4902 | 0.107 | 0 |
| Stagewise | 0.1107 | 0.3179 | 0 | 0.3251 | 0.2007 | 0.2259 | 0.4118 | 0.0904 | 0 |
| IF Stagewise | 0.1105 | 0.3137 | 0.2465 | 0.3256 | 0.2009 | 0 | 0.4119 | 0.0904 | 0 |
| Best Linear | | 0.3026 | | | 0.2006 | | | 0.0905 | |

Table 4: Monte Carlo: Out-of-Sample Root Mean Squared Prediction Error

| N | bagging | RF | poly | lasso | step | stage | IFS | best |
|---|---|---|---|---|---|---|---|---|
| 100 | 0.04366 | 0.04366 | 83.70950 | 0.23643 | 0.25931 | 1.25375 | 16.54100 | 0.11526 |
| 200 | 0.03516 | 0.03608 | 27.69800 | 0.23034 | 0.25320 | 0.73421 | 2.53325 | 0.18970 |
| 400 | 0.02939 | 0.02980 | 10.21270 | 0.22693 | 0.24085 | 0.27002 | 33.43815 | 0.05714 |
| 800 | 0.02377 | 0.02378 | 3.65680 | 0.22368 | 0.22511 | 0.11785 | 1.29285 | 0.02776 |
| 1600 | 0.01839 | 0.01839 | 2.64430 | 0.20718 | 0.21256 | 0.08971 | 1.30170 | 0.02344 |
| 3200 | 0.01344 | 0.01369 | 1.56610 | 0.18560 | 0.18739 | 0.05324 | 2.40746 | 0.01516 |
| 6400 | 0.00931 | 0.00950 | 0.31950 | 0.15507 | 0.15708 | 0.01397 | 0.62162 | 0.00639 |
| 12800 | 0.00633 | 0.00636 | 0.18308 | 0.12408 | 0.12616 | 0.00935 | 0.34044 | 0.00548 |
| 25600 | 0.00414 | 0.00416 | 0.09440 | 0.09461 | 0.10077 | 0.00995 | 0.19680 | 0.00764 |
| 51200 | 0.00257 | 0.00262 | 0.06922 | 0.06983 | 0.07218 | 0.00520 | 0.11254 | 0.00236 |
| 102400 | 0.00165 | 0.00161 | 0.02730 | 0.05052 | 0.05083 | 0.00153 | 0.12369 | 0.00093 |

Table 5: Monte Carlo: Convergence Rates

| N | bagging | RF | poly | lasso | step | stage | IFS | best |
|---|---|---|---|---|---|---|---|---|
| 200 | 1.2419 | 1.2102 | 3.0222 | 1.0264 | 1.0241 | 1.7076 | 6.5296 | 0.6076 |
| 400 | 1.1963 | 1.2106 | 2.7121 | 1.0150 | 1.0513 | 2.7191 | 0.0758 | 3.3200 |
| 800 | 1.2363 | 1.2532 | 2.7928 | 1.0145 | 1.0699 | 2.2913 | 25.8639 | 2.0581 |
| 1600 | 1.2925 | 1.2931 | 1.3829 | 1.0796 | 1.0590 | 1.3136 | 0.9932 | 1.1846 |
| 3200 | 1.3680 | 1.3434 | 1.6885 | 1.1163 | 1.1343 | 1.6849 | 0.5407 | 1.5456 |
| 6400 | 1.4435 | 1.4415 | 4.9017 | 1.1968 | 1.1930 | 3.8110 | 3.8729 | 2.3720 |
| 12800 | 1.4705 | 1.4936 | 1.7451 | 1.2498 | 1.2451 | 1.4940 | 1.8259 | 1.1656 |
| 25600 | 1.5306 | 1.5286 | 1.9395 | 1.3115 | 1.2520 | 0.9396 | 1.7299 | 0.7178 |
| 51200 | 1.6100 | 1.5874 | 1.3637 | 1.3549 | 1.3961 | 1.9148 | 1.7487 | 3.2438 |
| 102400 | 1.5551 | 1.6256 | 2.5360 | 1.3821 | 1.4202 | 3.4022 | 0.9099 | 2.5333 |
| Average | 1.3945 | 1.3987 | 2.4085 | 1.1747 | 1.1845 | 2.1278 | 4.4090 | 1.8748 |

forward stagewise models all have significantly better than the parametric lower bound rate of convergence. The LASSO model and the stepwise models both have empirical rates of convergence below the cube-root lower bound, but this is due to those two models having additional degrees of complexity which increase with the sample size. In the case of the LASSO model, the Matlab implementation changes the penalty parameter, $\lambda$, in accordance with the sample size. Most importantly, the model combination exhibits a rate of convergence that is slightly faster than the parametric lower bound.

# 5 Empirical Application

This section compares econometric models with machine learning ones using a typical demand estimation scenario – grocery store sales. The machine learning models in general produce better out-of-sample fits than linear models without loss of in-sample goodness of fit . If we combine all the models linearly with non-negative weights, the resulting combination of models produces better out-of-sample fit than any model in the combination.

This section also illustrates how machine learning models could work with unstructured data or sparse data. Unstructured data is not organized to feed into models directly like structured data. For instance, the text description of a bag of chips and the image of the bag are unstructured data. Sparse data is a type of data where most of the elements are zeros. Both of unstructured and sparse data would be hard to handle in econometric models.

The last part of this section uses the same data and model structures to estimate the promotional lift of sales.

The data we use is provided by IRI Marketing Research via an academic license at the University of Chicago. It contains scanner panel data from grocery stores within one grocery store chain for six years. We used sales data on salty snacks, which is one of the categories in the IRI data.

A unit of observation is product $j$, uniquely defined by a UPC (Universal Product Code), in store $m$ at week $t$. The number of observations are 1510563, which includes 3149 unique products. Let $q_{jmt}$ be the number of bags of salty snack $j$ sold in store $m$ at week $t$. If $q_{jmt} = 0$, we do not know if it is due to no sale or out-of-stock and the observation is not filled in. The price $p_{jmt}$ is defined as the quantity weighted average of prices for product $j$ in store $m$ at week $t$. Therefore if $q_{jmt} = 0$, the weight is also 0. In addition to price and quantity, the data contains attributes of the products (such as brand, volume, flavor, cut type, cooking method, package size, fat and salt levels) and promotional variables (promotion, display and

feature).

Table 6 shows the summary statistics of quantity, price and volume. Table 7 shows the three most common values of product attributes - brand, volume, flavor, cut type, cooking method, package size, fat and salt levels.

Table 6: Summary Statistics

| **Variable** | **Mean** | **Min** | **Max** | **1st Qu.** | **Median** | **3rd Qu.** |
|---|---|---|---|---|---|---|
| Quantity | 20.08 | 1.00 | 3402.00 | 4.00 | 9.00 | 19.00 |
| Price | 2.05 | 0.10 | 5.69 | 1.39 | 1.99 | 2.87 |
| Volume | 0.52 | 0.06 | 1.25 | 0.34 | 0.41 | 0.59 |

Table 7: Tabulate Category Variables

| **Variable** | **Three Most Frequent Values** | | |
|---|---|---|---|
| Brand | Pringles | Utz | Lays |
| Product Type | Potato Chip | Potato Crisp | Potato Chip and Dip |
| Packaging | Bag | Canister | Cardboard Canister |
| Flavor | Original | Sour Cream & Onion | BBQ |
| Fat Content | Missing | Low Fat | Fat Free |
| Cooking Method | Missing | Kettle Cooked | Old Fashion Kettle |
| Salt Content | Missing | Lightly Salted | Sea Salt |
| Cutting Type | Flat | Missing | Ripple |

In our application, we compare nine alternative models of demand to predict $q_{jmt}$. The nine models are linear model, stepwise, forward stagewise, LASSO, random forest, support vector machine, gradient boosting trees, Logit and Logit with gradient boosting variable selection. Linear model and Logit are traditional econometric models where the others are popular machine learning algorithms. We also perform a linear combination of all the models in the effort to increase prediction accuracy. We use R to compute all the results and a list of related R packages is provided in this section.

## 5.1 Linear Regression Models

The linear regression is a typical approach to estimate demand by approximating the demand using a function form of the following:

$$log(q_{jmt}) = \beta' X_{jmt} + \zeta_{mt} + \eta_{jm} + \epsilon_{jmt}$$

where $X_{jmt}$ is the matrix of attributes including log of own prices, product attributes, advertising and promotion indicators, $\zeta_{mt}$ is the market specific seasonal factor, $\eta_{jm}$ is the product specific market effect and $\epsilon_{jmt}$ is an idiosyncratic shock to each product, market and time.

Table 8 shows the output of linear model where only the significant coefficients are displayed.

## 5.2 Logit

We followed Berry et al. (1995) to do a logit-type model of market shares. Assuming market sizes are fixed, we estimate the shares in two ways: 1) by using traditional regression; 2) by using gradient boosting (R package **gbm**). Gradient boosting is where tree models are utilized as base-learners. Specifically, we are interested in L2(Euclidean distance) loss in the boosting tree in regression. Thus it is nothing else than repeated least squares fitting of residuals. Both approach 1) and 2) project estimated $\hat{q}_{jmt}$ on product attributes dummies, store fixed effects and week fixed effects.

Then we sum $\hat{q}_{jmt}$ over stores and weeks. Assuming that market sizes are fixed, we calculate market share by dividing $\hat{q}_{jmt}$ by market size. The log of market share is taken as the dependent variable in our Logit model. Table 9 shows the output of Logit model with traditional regression where only the significant coefficients are displayed.

## 5.3 Stepwise, Forward Stagewise and LASSO

In practice, all three models can be realized in R package **lars**. We take the default parameter $t$ and $\lambda$ in the package. These three model converge in similar ways where in each step, the most important variable is added to the model. We limit the maximum number of steps to 100 in our practice for demonstration purposes because it takes significantly longer to converge if the number of steps is larger. Table 10 shows how many variables have non-zero coefficients and which they are.

## 5.4 Random Forest

Random forecast is implemented in R package **randomForest**. The two parameters are the number of trees and the number of variables to try at each split.

Table 11 displays the variable importance of the twelve most important variables in determining Log Quantity using two metrics. The percentage *increase in mean squared error* is the increased percentage of mean squared error if a variable is excluded from the

Table 8: Linear Regression

| Log Quantity | Estimate | Std. Error | $t$ value | $\Pr(>\|t\|)$ | |
|---|---|---|---|---|---|
| Log Price | -0.639 | 0.055 | -11.708 | < 2e-16 | *** |
| Promotion | 0.466 | 0.039 | 11.926 | < 2e-16 | *** |
| Feature: None | -0.630 | 0.067 | -9.334 | < 2e-16 | *** |
| Display: | | | | | |
| Minor | 0.708 | 0.049 | 14.341 | < 2e-16 | *** |
| Major | 0.637 | 0.049 | 13.119 | < 2e-16 | *** |
| Brand: | | | | | |
| Herrs | -0.351 | 0.156 | -2.253 | 0.024 | * |
| Jays | -1.101 | 0.244 | -4.516 | 0.000 | *** |
| Kettle Chips | -0.995 | 0.236 | -4.217 | 0.000 | *** |
| Lays | -0.337 | 0.159 | -2.124 | 0.034 | * |
| Lays Bistro Gourmet | -0.656 | 0.188 | -3.480 | 0.001 | *** |
| Lays Natural | -1.662 | 0.327 | -5.079 | 0.000 | *** |
| Lays Stax | -1.481 | 0.183 | -8.104 | 0.000 | *** |
| Lays Wow | -0.485 | 0.204 | -2.379 | 0.017 | * |
| Michael Seasons | -1.655 | 0.239 | -6.921 | 0.000 | *** |
| Pringles | -0.794 | 0.156 | -5.090 | 0.000 | *** |
| Pringles Cheezums | -0.644 | 0.211 | -3.055 | 0.002 | ** |
| Pringles Fat Free | -0.624 | 0.189 | -3.308 | 0.001 | *** |
| Pringles Prints | -1.876 | 0.314 | -5.982 | 0.000 | *** |
| Pringles Right Crisps | -0.881 | 0.128 | -6.892 | 0.000 | *** |
| Ruffles Natural | -1.379 | 0.389 | -3.549 | 0.000 | *** |
| Ruffles Snack Kit | -1.555 | 0.307 | -5.061 | 0.000 | *** |
| Utz | -0.543 | 0.149 | -3.635 | 0.000 | *** |
| Wise | -0.505 | 0.165 | -3.062 | 0.002 | ** |
| Wise Ridgies | -0.984 | 0.167 | -5.888 | 0.000 | *** |
| Volume | 0.469 | 0.113 | 4.142 | 0.000 | *** |
| Package: | | | | | |
| Canister | 0.437 | 0.091 | 4.800 | 0.000 | *** |
| Canister In Box | 0.453 | 0.130 | 3.494 | 0.000 | *** |
| Flavor: | | | | | |
| BBQ | 0.167 | 0.066 | 2.534 | 0.011 | * |
| Cheddar | 0.241 | 0.080 | 3.026 | 0.002 | ** |
| Cheese | -0.443 | 0.205 | -2.164 | 0.031 | * |
| Ketchup | -0.680 | 0.244 | -2.787 | 0.005 | ** |
| Onion | 0.339 | 0.066 | 5.107 | 0.000 | *** |
| Original | 0.704 | 0.061 | 11.588 | < 2e-16 | *** |
| Spicy | -0.211 | 0.105 | -2.005 | 0.045 | * |
| Salt: No Salt | -0.446 | 0.212 | -2.099 | 0.036 | * |
| Type of Cut: Flat | 0.308 | 0.070 | 4.411 | 0.000 | *** |
| Store Fixed Effects | Yes | | | | |
| Week Fixed Effects | Yes | | | | |
| Adjusted R-squared | 0.884 | | | | |
| Significance | 0 *** | 0.001 ** | 0.01 * | 0.05 . | |

Table 9: Logit with Regression Selection

| Log Share | Estimate | Std. Error | $t$ value | Pr($>\|t\|$) | |
|---|---|---|---|---|---|
| Log Price | 0.296 | 0.113 | 2.624 | 0.009 | ** |
| Promotion | -0.441 | 5.192 | -0.085 | 0.932 | |
| Feature: None | 0.263 | 0.151 | 1.745 | 0.081 | . |
| Display: | | | | | |
| Minor | -0.215 | 0.104 | -2.080 | 0.038 | * |
| Major | -0.338 | 0.113 | -3.000 | 0.003 | ** |
| Brand: | | | | | |
| Herrs | 0.515 | 1.846 | 0.279 | 0.780 | |
| Jays | 0.743 | 1.864 | 0.398 | 0.690 | |
| Kettle Chips | 0.956 | 1.942 | 0.492 | 0.622 | |
| Lays | 0.328 | 1.844 | 0.178 | 0.859 | |
| Lays Bistro Gourmet | 0.145 | 2.331 | 0.062 | 0.951 | |
| Lays Natural | 0.630 | 2.627 | 0.240 | 0.811 | |
| Lays Stax | 1.169 | 3.187 | 0.367 | 0.714 | |
| Lays Wow | | | | | |
| Michael Seasons | 0.651 | 1.980 | 0.329 | 0.742 | |
| Pringles | 0.915 | 3.178 | 0.288 | 0.773 | |
| Pringles Cheezums | 1.607 | 3.456 | 0.465 | 0.642 | |
| Pringles Fat Free | 0.965 | 3.359 | 0.287 | 0.774 | |
| Pringles Prints | 1.755 | 3.251 | 0.540 | 0.589 | |
| Pringles Right Crisps | 0.863 | 3.210 | 0.269 | 0.788 | |
| Ruffles Natural | -0.116 | 2.755 | -0.042 | 0.966 | |
| Ruffles Snack Kit | -0.294 | 2.660 | -0.110 | 0.912 | |
| Utz | 0.447 | 1.845 | 0.242 | 0.809 | |
| Wise | 0.738 | 1.859 | 0.397 | 0.691 | |
| Wise Ridgies | 0.737 | 1.854 | 0.397 | 0.691 | |
| Volume | -0.024 | 0.451 | -0.054 | 0.957 | |
| Package: | | | | | |
| Canister | -0.432 | 0.514 | -0.842 | 0.400 | |
| Canister In Box | -0.296 | 0.670 | -0.442 | 0.658 | |
| Flavor: | | | | | |
| Bbq | -0.976 | 1.367 | -0.714 | 0.475 | |
| Cheddar | -0.984 | 1.412 | -0.697 | 0.486 | |
| Cheese | -1.608 | 2.141 | -0.751 | 0.453 | |
| Ketchup | -0.753 | 1.986 | -0.379 | 0.705 | |
| Onion | -0.951 | 1.425 | -0.668 | 0.504 | |
| Original | -1.158 | 1.311 | -0.883 | 0.377 | |
| Spicy | -0.608 | 1.381 | -0.440 | 0.660 | |
| Salt: No Salt | 0.644 | 1.742 | 0.370 | 0.712 | |
| Type of Cut: Flat | -0.556 | 0.554 | -1.003 | 0.316 | |
| Store Fixed Effects | No | | | | |
| Week Fixed Effects | No | | | | |
| AIC | 6884.4 | | | | |
| Significance | 0 *** | 0.001 ** | 0.01 * | 0.05 . | |

Table 10: Non-zero Coefficients

| | | **Stepwise** | **Forward Stagewise** | **LASSO** |
|---|---|---|---|---|
| # of Covariates | | 466 | 466 | 466 |
| # Non-zero Covariates | | 40 | 39 | 37 |
| Non-zero Covariates | | Promotion | Promotion | Promotion |
| | | Volume | Volume | Volume |
| | Display | Minor | Minor | Minor |
| | | Major | Major | Major |
| | Brand | Baked Ruffles | Lays | Lays |
| | | Herrs | Ruffles | Ruffles |
| | | Jays Krunchers | Ruffles Wow | Ruffles Wow |
| | | Wavy Lays | Jays Krunchers | Jays Krunchers |
| | | Wise | Wavy Lays | Wavy Lays |
| | | | Wise | |
| | Flavor | BBQ | Onion | Onion |
| | | Chedder | Original | Original |
| | | Onion | | |
| | | Original | | |
| | Cooking | Crispy | Crispy | Kettle Cooked |
| | | Kettle Cooked | Kettle Cooked | |
| | Salt | Missing | Missing | Missing |

model. Node purity measures how much the additional variable or tree split reduces the residual sum of squares. Thus, the *increase in node purity* measures the size change in node purity if a variable is excluded from the model.

Table 11: Random Forest Variable Importance

| Log Quantity | %Increase in Mean Squared Error | Increase in Node Purity |
|---|---|---|
| Log Price | 0.34 | 580.21 |
| Volume | 0.18 | 330.54 |
| Promotion | 0.09 | 230.10 |
| Brand: Lays | 0.07 | 125.44 |
| Feature: None | 0.06 | 250.47 |
| Product Type: Potato Chip | 0.06 | 23.52 |
| Display: Major | 0.05 | 181.25 |
| Type of Cut: Missing | 0.05 | 18.89 |
| Product Type: Potato Crisp | 0.04 | 19.78 |
| Brand: Wavy Lays | 0.04 | 77.36 |
| Display: Minor | 0.04 | 133.92 |
| Flavor: Classic | 0.03 | 90.09 |
| R-Squared | 0.40 | |

## 5.5 Support Vector Machine

Support vector machines are a penalized method of regression. The tuning parameter, $\epsilon$, controls which errors are included in the regression. Errors of sufficiently small size are treated as zeros. Typically only a partial set of the covariates are assigned a non-zero value in support vector machine regressions. R package **e1071** is used for support vector machine.

## 5.6 Bagging

Bagging, short for Bootstrap Aggregation, is an early ensemble method that builds multiple trees by resampling training data with replacement, and voting the trees for a prediction. We use the `bagging` function and its default parameters in R package **ipred**.

## 5.7 Combined Model

In order to compare models, we want to split the data into training and testing set, where we train the model using the training set and pretend the dependent variable in the test set is unknown and predict on the test set. Because we have eight models, we want to assign weight to each model when building a linear combination. However, the weight based on the training set is biased. Some models like linear model tend to get a good in-sample fit but a bad out-of-sample fit, and this grants these model a very large in-sample weight which could be misleading.

Therefore, we randomly split the data into three pieces to do cross validation, where the model weights are determined on the validating set. This three way data partitioning mitigates the possible large out-of-sample error for some models when over fitting happens. 25% of the data is used as the test set, 15% is used as the validate set, and the remaining 60% is used as the training set. Table 12 shows how the data is sliced into three pieces.

In Table 12, we compare nine models: two of them are traditional econometrics models and seven of them are more in the context of machine learning (as introduced in the main paper). Our purpose is to run a horse race of models by comparing out-of-sample prediction errors. First, all the models are used to fit the data in the train set. Next, make out-of-sample prediction on the validate set and get the weight for each model by combining them linearly. Last, use the fitted models to predict in the test set, and use the weights from validate set to form the linearly combined prediction.

The response variable is log of quantity sold per week. The covariates are log of price, product attributes variables, promotional variables, store fixed effects, and week fixed effects. We provide the same covariate matrix to all of the models expect for the Logit model, where all the fixed effects are excluded.

Table 12 shows the comparison of the models. In the scenario of out-of-sample prediction error, the best two models are random forest and support vector machine. The combined model, where we regress the actual value of the response variable on a constrained linear model of the predictions from eight models, outperforms all the eight models, which follows the optimal combination of forecasts in Bates and Granger (1969). Random forest and support vector machine get more weights in the combined model due to their good performance out-of-sample.

Based on Section 3, the combination of models converges to asymptotic normal distribution at $\sqrt{n}$ rate, regardless of the what individual models there are. Therefore, we could bootstrap the combined model to get the confidence interval, knowing that it's asymptotic

Table 12: Model Comparison: Prediction Error

| | Validation | | Test | | |
|---|---|---|---|---|---|
| | RMSE | Std. Err. | RMSE | Std. Err. | Weight |
| Linear | 1.169 | 0.022 | 1.193 | 0.020 | 6.62% |
| Stepwise | 0.983 | 0.012 | 1.004 | 0.011 | 12.13% |
| Forward Stagewise | 0.988 | 0.013 | 1.003 | 0.012 | 0.00% |
| Lasso | 1.178 | 0.017 | 1.222 | 0.012 | 0.00% |
| Random Forest | 0.943 | 0.017 | 0.965 | 0.015 | 65.56% |
| SVM | 1.046 | 0.024 | 1.068 | 0.018 | 15.69% |
| Bagging | 1.355 | 0.030 | 1.321 | 0.025 | 0.00% |
| Logit (Boosting) | 1.190 | 0.019 | 1.234 | 0.017 | 0.00% |
| Logit | 1.190 | 0.020 | 1.234 | 0.018 | 0.00% |
| Combined | 0.924 | | 0.946 | | 100.00% |
| # of Obs | 226952 | | 376980 | | |
| Total Obs | 1510563 | | | | |
| % of Total | 15.0% | | 25.0% | | |

normal.

Table 13 provides the summary statistics of the residual between predicted and actual values of quantity in the testing set. This is meant to be a measure of accuracy of each model. It is obvious that the Machine Learning models fit better out of sample and the combined model fits better than any of the individual models.

## 5.8 Discussion

### 5.8.1 Variable Selection

When the number of independent variables is large, it is common to have some degree of multicollinearity. The shrinkage models could help us reduce the multicollinearity intelligently. A usual statistics to determine multicollinearity is Variance Inflation Factors(VIF). The VIF for covariate $X_j$ is defined as:

$$VIF_j = \frac{1}{1 - R^2_{-j}} \tag{20}$$

where $R^2_{-j}$ is the R-squared by regressing covariate $X_j$ on the rest of covariates. Table 14 shows some VIFs for the independent variables used in the linear regression model. To

Table 13: Summary Statistics of Residual in Test Prediction

| Residuals | Mean | Mean Std.Err. | Std.Dev. | Min | Max |
|---|---|---|---|---|---|
| Linear | -1.63 | 0.70 | 25.32 | -252.27 | 141.00 |
| Stepwise | -5.00 | 0.66 | 24.12 | -314.02 | 129.16 |
| Forward Stagewise | -6.14 | 0.69 | 25.27 | -323.43 | 26.64 |
| Lasso | -7.06 | 0.76 | 27.70 | -346.19 | 5.81 |
| Random Forest | -5.56 | 0.68 | 24.89 | -325.79 | 34.13 |
| SVM | -3.66 | 0.65 | 23.55 | -277.56 | 66.60 |
| Bagging | -5.82 | 0.77 | 28.23 | -347.98 | 39.54 |
| Logit (Boosting) | -7.23 | 0.76 | 27.78 | -347.22 | 15.00 |
| Logit | -7.23 | 0.76 | 27.78 | -347.29 | 15.00 |
| Combined | -5.44 | 0.67 | 24.40 | -316.41 | 39.79 |

compare, after LASSO selects variable, we also have the VIFs for the independent variables in the linear regression using only selected variables. The VIFs in the non-selecting case are much higher than the LASSO selection for many variables. Using 5 as a threshold for multicollinearity, LASSO successfully reduces multicollinearity in the independent variables. We believe it's more attractive than ad hoc variable selection procedures commonly used in practice.

### 5.8.2 Bag of Words

The mere feat of encoding a vector of hedonic attributes commonly used in demand estimation for this data could be cumbersome. In applied econometrics, researchers commonly restrict attention to the products with the largest demand and encode regressors for brands and a small vector of product attributes. In our application, we propose using unsupervised learning to construct product level regressors. In particular, we use the unstructured text that describes the product in the raw data and apply the bag of words model. This has the advantage of being a simpler computation and allows us to avoid the arduous task of encoding attributes for thousands of products. Since it is more scalable, it allows us to model the demand for all of the products rather than restricting attention to the products with the largest demand. Also, it could be viewed as less ad hoc than hand coding product attributes since it imposes fewer a priori restrictions on the hedonic attributes that we should include as regressors in our model. We note that there is a large literature on this form of unsupervised learning which sometimes goes by the name of feature extraction in computer

Table 14: Variance Inflation Factors

| | VIFs after Selection | |
|---|---|---|
| **Variable** | **Linear** | **LASSO** |
| Product Type - Potato Chip And Dip | $+\infty$ | 3.5084 |
| Brand - Ruffles Snack Kit | $+\infty$ | 3.4729 |
| Logprice | 4.1750 | 3.2319 |
| Volumn | 3.9775 | 3.1541 |
| Cooking - Missing | $+\infty$ | 3.1100 |
| Cooking - Kettle | $+\infty$ | 2.6495 |
| Package - Canister | $+\infty$ | 1.8047 |
| Fat - Regular | 76.6610 | 1.5930 |
| Brand - Lays | 104.5904 | 1.5187 |
| Promotion | 1.4806 | 1.4388 |
| Feature - None | 2.3398 | 1.3369 |
| Brand - Kettle Chips | 27.3608 | 1.3222 |
| Flavor - Original | 2.8610 | 1.2875 |
| Brand - Ruffles | 50.1427 | 1.2802 |
| Salt - Regular | 3.0660 | 1.2732 |
| Brand - Wavy Lays | 36.1675 | 1.1925 |
| Brand - Lays Stax | $+\infty$ | 1.1688 |
| Display - Major | 1.1710 | 1.1498 |
| Brand - Ruffles Wow | 19.3494 | 1.1398 |
| Brand - Baked Ruffles | $+\infty$ | 1.1291 |
| Display - Minor | 1.1577 | 1.1191 |
| Brand - Lays Wow | 16.0920 | 1.1173 |
| Flavor - Missing | 1.5811 | 1.0942 |
| Brand - Ruffles Light | 10.5084 | 1.0898 |
| Type Of Cut - Thick | 11.0995 | 1.0757 |
| Brand - Wise Ridgies | 12.4572 | 1.0712 |
| Brand - Wachusett | 1.7886 | 1.0662 |
| Brand - Michael Seasons | 4.5692 | 1.0594 |
| Brand - Tastee | 1.8020 | 1.0529 |
| Brand - Michaels Gold N Good | 2.2383 | 1.0462 |
| Brand - Better Made Special | 2.7512 | 1.0386 |
| Brand - Pringles Prints | $+\infty$ | 1.0281 |
| Brand - Laura Scudder | 3.6605 | 1.0236 |
| Brand - Lance Thunder | 3.1482 | 1.0218 |
| ... | | |

science. With the exception of Gentzkow and Shapiro (2010), there has been relatively little use of this method in economics. We believe that this is promising for demand estimation because it allows a new source of data to construct covariates such as the raw text in product descriptions or product reviews.

Aside from structured features like package type, cut type of potato chips, the hedonic attributes of the product can also be defined by unstructured texts that describe the product. A bag of words model could easily turn the unstructured texts into large amounts of features. Unsupervised learning (clustering) on these features could be a more scalable approach than hand coding them. The procedures following the unsupervised learning are the same as the way we deal with structured features. Yet this straightforward approach has better prediction power than structured models.

The bag of words model analyzes a corpus of $K$ documents, comprising a dictionary of $M$ words, and finds the relations of words and documents. In our case, the $K$ documents are the $n$ unique potato chips descriptions. We cluster the descriptions, via manipulation of the document-term matrix. A document-term matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of descriptions. In a document-term matrix, rows correspond to each unique product in the collection and columns correspond to terms. Where it is tedious to encode all the features of a product, bag of words provides a simple way to exploit the rich features of products.

In a high dimensional learning problem, only some parts of an observation are important to prediction. For example, the information to correctly categorizing a product may lie in a handful of its words. The extraneous words could prove computational burdensome, so word regularization may be helpful. Possible methods for regularization include LASSO and other shrinkage models. Therefore it's natural to combine the technique of bag of words with our models. Bag of words

### 5.8.3 Top/Tail Products

In marketing literature, people usually prune out the tail products (for example Nevo (2001)). But, with the new methods, we want to show that a training data set with all possible products predicts better than a data set with only the top products. To show that, we take only the top twenty products to train the model and predict the sales of the tail products and compare the prediction fit to those in Table 12.

We use the same nine models as modeling examples to demonstrate the difference. We rank all the products by total units sold and only mark the top twenty as the training set.

Using exactly the same methodology in our main application, we get the root mean squared error as a measure of fit for nine plus combine model. The prediction error and weights is presented in Table 15.

As the table shows, when we only train on top twenty products, the fit out of sample is worse than when we train on a randomly split training set for all models. As the top twenty products may not necessarily explain all the features of the rest of the products, the predicting power is therefore weakened.

Table 15: Top 20 Products vs. the Other Products

| | Top 20 Products | | Other Products | | |
|---|---|---|---|---|---|
| | RMSE | Std. Err. | RMSE | Std. Err. | Weight |
| Linear | 0.2381 | 0.0068 | 2.1910 | 0.0827 | 9.66% |
| Stepwise | 0.7059 | 0.0104 | 1.6025 | 0.0248 | 0.00% |
| Forward Stagewise | 0.8657 | 0.0156 | 1.1517 | 0.0270 | 53.38% |
| Lasso | 0.8918 | 0.0175 | 1.1532 | 0.0202 | 0.00% |
| Random Forest | 0.8728 | 0.0160 | 1.1834 | 0.0182 | 21.93% |
| SVM | 0.4165 | 0.0128 | 1.7190 | 0.0513 | 0.00% |
| Bagging | 0.5472 | 0.0162 | 1.3573 | 0.0323 | 15.04% |
| Logit (Boosting) | 1.0509 | 0.0185 | 1.7017 | 0.0289 | 0.00% |
| Logit | 1.2076 | 0.0278 | 1.5339 | 0.0420 | 0.00% |
| Combined | 0.2381 | | 1.1183 | | 100.00% |
| # of Obs | 250149 | | 1260414 | | |
| Total Obs | 1510563 | | | | |
| % of Total | 16.56% | | 83.44% | | |

### 5.8.4 Practical Advantages

There are some other practical advantages to the machine learning models or their generic approaches.

In random forest, the missing values could be imputed, for example, as the median of its column. This imputation will not effect accuracy much since the randomness of subsampling and the trees grown. Another approach for categorical variables is to simply create a new category called "missing". This new category might capture the behavioral differences in observations with missing values and the ones not missing.

If we have a large dataset and we want to do model selection or model assessment, the best approach is to randomly divide the dataset into three parts: a training set, a validation

set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. The test set should be kept separately and be brought out only at the end of the analysis.

If, however, we do not have enough data to split three ways, we can efficiently re-use the sample data by cross-validation or bootstrap.

In addition to the models we mentioned in our paper, there are some other very popular machine learning approaches that could be helpful to readers in economics. For example, EM (Expectation-Maximization) algorithm for simplifying difficult maximum likelihood problems; Graphical models for complicated conditional expectations; Neural Networks for extracting linear combination of features and modeling response variable as a non-linear function of the features, and so on.

## 5.9 Lift Estimates

The other interesting problem we want to look at is estimating promotional lift. In our data, a product is tagged as on promotion if the price reduction is greater than 5%. By considering promotion as a randomized treatment, we follow the methodology suggested by Varian (2014) to estimate the promotional lift. Our models are trained on the control group (no promotion) and then used to predict out of sample on the treated group (on promotion). The control/treatment partition is based on whether the product is on price reduction promotion, instead of random assignment in the last example.

The most important variables we want to study are the price promotion as well as how the store display the products. A product is tagged as in promotion if the price reduction is greater than 5%. There are four levels of feature: large, medium, small and no ad. There are three levels of display: major(including lobby and end-aisle), minor and none. Table 16 ensures that every value of the variables has some level of presentation in each data set.

If everything else is the same in the control group and the treatment group, we believe the difference between predicted and actual in treatment group is our treatment effects. Based on model coefficients from the control group, we predict the quantities using the independent variables in the treated group. The difference between the predicted and the actual value of quantity sold in the treated group is therefore our lift estimates.

The lift estimates from eight models mentioned before are in Table 17. We also use the same weights as in the comparison of prediction error to construct the lift estimate for a linearly combined model. All the machine-learning models calculate a positive promotional

lift when only the linear model produces a negative lift for promotion. The big variance in the linear model residual indicates that the negative sign of the promotional lift in the linear model comes from variance instead of bias.

After this step, we regress the fitted residual of lift on dummies using LASSO. The dummies we use are product attributes, store dummies and week dummies. Chernozhukov (2013) suggests that using the selected variables from LASSO can increase the fit of the model. We then use the selected variables in an ordinary least square and fit for the lift residual.

Table 16: Promotion Variables

| | Frequency | Percent |
|---|---|---|
| Promotion | | |
| Price Reduction<5% | 1143490 | 75.7% |
| Price Reduction>5% | 367373 | 24.3% |
| Total | 1510563 | |

Table 17: Model Comparison:Promotional Lift

| | Mean | Mean Std.Err. | Std.Dev. | Min | Max |
|---|---|---|---|---|---|
| Linear | 12.79 | 0.59 | 27.22 | 0.23 | 1007.06 |
| Stepwise | 10.79 | 0.25 | 11.82 | 0.22 | 302.28 |
| Forward Stagewise | 8.23 | 0.09 | 4.32 | 2.42 | 33.73 |
| Lasso | 8.24 | 0.09 | 4.32 | 2.45 | 33.76 |
| Random Forest | 9.25 | 0.13 | 5.96 | 1.96 | 50.93 |
| SVM | 11.98 | 0.27 | 12.43 | 0.48 | 131.62 |
| Bagging | 6.30 | 0.09 | 3.95 | 1.36 | 29.00 |
| Logit (Boosting) | 7.34 | 0.04 | 1.92 | 0.00 | 17.00 |
| Logit | 9.97 | 0.19 | 9.05 | 1.00 | 174.00 |
| Combined | 9.49 | 0.14 | 6.46 | 1.63 | 59.90 |

## 5.10 Computation Tools

In this application we face constraints on memory and CPU when processing the data with millions of observations with complicated machine learning models. The time it takes to compute a Random Forest object with data of such size using a single work station could

be days, even weeks. A solution to solve the computation constraints is to utilize high performance computing tools, such as parallel processing using Revolution R® and MATLAB® Parallel Computing Toolbox™. Both of them are available on Amazon Web Services(AWS) Marketplace at low costs.

Revolution R has two major packages - RevoR and ScaleR. RevoR automatically uses all available cores and processors to reduce computation times without modifying a standard R script. A real time simulation shows it speeds up computation by three to thirty times compared to a single core standard R process. ScaleR scales up data size to 1 to 16 tera-bytes easily by dividing data into pieces and accessing the pieces with different processors.

In these parallel processing frameworks, a common implementation is MapReduce. It is useful when the data is too large to be held or processed in memory. There are two stages in MapReduce: Map and Reduce. In the Map stage the program (Revolution R® or MATLAB®) pre-processes each distributed data trunk in parallel and performs preliminary statistical modeling on each data trunk in parallel. In the Reduce stage, the program gathers all the information for each distributed data trunk from Map stage, summarizes the information and returns it to the user. This is much faster and takes less memory than storing and accessing data directly from memory.

# 6 Conclusion

In this paper, we survey a set of models from statistics and computer science. We select a few machine learning models to compare with traditional econometric models. The models we focus on in this paper include linear regression as the baseline model, logit as the econometric model, stepwise, forward stagewise, LASSO, random forest, support vector machine and bagging as the machine learning models. We derive novel asymptotic properties for the machine learning models.We use Monte Carlo simulation to demonstrate the properties of the models and also show that combining all the underlying models with a linear regression improves out-of-sample prediction accuracy.

We illustrate the properties of these models by using a real world data set with scanner panel sales data of potato chips. First, we compare the prediction accuracy of these models and the machine learning models consistently give better out-of-sample prediction accuracy while holding in-sample prediction error comparable. By combining all the models via weighted linear regression, we are able to improve the out-of-sample prediction accuracy even more. Second, we compare two scenarios where one, as the traditional marketing litera-

ture suggests, prunes the market to only the top sold products, and the other, our approach, uses a mix of both top and tail products. Our approach has better prediction accuracy in demand. Third, we estimate the promotion lift using the predictions for treatment effects. Last, we explored the unstructured text of product description from the raw data and apply the bag of words model. This has the advantage of being a simpler computation and allows us to avoid the arduous task of encoding attributes for thousands of products.

Our approach is robust to a large number of potentially collinear regressors; it scales easily to large data sets; the linear combination method selects the best model automatically and produces the best in-sample and out-of-sample prediction accuracy; and the method can flexibly approximate arbitrary non-linear functions, even when the set of regressors is high dimensional and we also allow for fixed effects.While demand estimation is our motivating application, we believe that the approach we have proposed can be useful in many other microeconometric settings.

# References


Abadie, A. and J. Gardeazabal (2003). The economic costs of conflict: A case study of the basque country. *American economic review*, 113–132.

Bates, J. M. and C. W. Granger (1969). The combination of forecasts. *Or*, 451–468.

Belloni, A., D. Chen, V. Chernozhukov, and C. Hansen (2012). Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica 80*(6), 2369–2429.

Belloni, A., V. Chernozhukov, and L. Wang (2011). Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika 98*(4), 791–806.

Berry, S., J. Levinsohn, and A. Pakes (1995). Automobile prices in equilibrium. *Econometrica 63*(4), 841–90.

Breiman, L. (1996). Bagging predictors. *Machine learning 24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Chernozhukov, V. (2013). Least squares after model selection in high-dimensional sparse models. *Bernoulli 19*, 521–547.

Gentzkow, M. and J. M. Shapiro (2010). What drives media slant? evidence from u.s. daily newspapers. *Econometrica 78*(1).

Kim, J. and D. Pollard (1990). Cube root asymptotics. *The Annals of Statistics*, 191–219.

Komarova, T. (2013). Binary choice models with discrete regressors: Identification and misspecification. *Journal of Econometrics 177*(1), 14–33.

Nevo, A. (2001). Measuring market power in the ready-to-eat cereal industry. *Econometrica 69*(2), 307–342.

Rajaraman, A. and J. D. Ullman (2011). Mining of massive datasets. *Lecture Notes for Stanford CS345A Web Mining 67*(3), 328.

Scornet, E. (2014). On the asymptotics of random forests. *arXiv preprint arXiv:1409.2090*.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley New York.

Varian, H. R. (2014). Big data: New tricks for econometrics. *The Journal of Economic Perspectives 28*(2), 3–27.