Statistical Modeling and Learning
Statistics 201B
Professor: Chad Hazlett
Authors: Wesley Cheng, Jireh Huang, and Stephen Smith

Bayesian Additive Regression Trees (BART)

Abstract

Over the course of this quarter, our study of various machine learning methods has demonstrated the utility of employing such methods for predictive power. While we learned about regularized regression and various generative methods, there was a lot of evidence pointing to the unique capabilities of tree models for prediction. Random Forest performed quite well on numerous types of problems in comparison to other ML techniques. With that in mind, we wanted to explore the topic of tree ensemble methods further and analyze the performance of another sum-of-trees method, namely Bayesian Additive Regression Trees (BART). Therefore, in this paper we will be explaining our understanding of the method and its technical details with the aid of the creators of this method and those who wrote an optimized library in `R` to implement the method (`bartMachine`). We will also apply it to a dataset predicting housing prices in King County, Washington and compare the results we achieve with BART to other ML methods. Finally, we will provide our conclusions about the method and important considerations for use in practice.

1 Introduction

1.1 Fundamental Problem and Solution

The fundamental problem for this method is to make inferences about an unknown function f that predicts an output Y using a p-dimensional vector of inputs $\mathbf{x}$, following the model

$$Y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim N(0, \sigma^2)$$

and to solve this problem, we consider modeling or approximating $\mathbb{E}(Y \mid \mathbf{x})$. The solution in BART is additive regression trees, or

$$f(\mathbf{x}) \approx h(\mathbf{x}) = \sum_{j=1}^{m} g_j(\mathbf{x})$$

where each g_j represents a regression tree. Thus, we are using a sum-of-trees model to approximate Y.

1.2 Summary of the Method

- What distinguishes BART from other ensemble of trees models is that it assumes a different probability model.
- "Conceptually, BART can be viewed as a Bayesian nonparametric approach that fits a parameter rich model using a strongly influential prior distribution."

1. Construct priors on T_j (tree structure), M_j (terminal node parameters), and σ (noise).
2. Sample from posterior to fit model using backfitting MCMC algorithm.

2 Method Details

2.1 Structure of an Individual Tree

- Each binary tree T_j contains a set of interior node decision rules and a set of b_j terminal nodes.
- Each tree also has corresponding terminal node parameters $M_j = \{\mu_1, \mu_2, \ldots, \mu_{b_j}\}$.
- If the range of $\mathbf{x} \in \mathbb{R}^p$ is partitioned such that $\mathbf{x} \in \{A_{\mu_{1j}} \cup A_{\mu_{2j}} \cup \cdots \cup A_{\mu_{b_j j}}\}$, then

$$g(\mathbf{x}; T_j, M_j) = \sum_{i=1}^{b_j} \mu_{ij} \mathbb{1}\left\{\mathbf{x} \in A_{\mu_{ij}}\right\}.$$

  - μ_{ij} represents a main effect when $g(\mathbf{x}; T_j, M_j)$ only depends on one variable in $\mathbf{x}$.
  - Similarly, μ_{ij} represents an interaction effect when $g(\mathbf{x}; T_j, M_j)$ depends on more than one variable in $\mathbf{x}$.

2.2 Priors

- Assume tree components (T_j, M_j) are independent of each other and of σ.
- Likewise, assume terminal node parameters μ_{ij} are independent.
- Thus,

$$\begin{aligned} & p\left((T_1, M_1), \ldots, (T_m, M_m), \sigma\right) \\ &= \left[\prod_{j=1}^{m} p(M_j \mid T_j) p(T_j)\right] p(\sigma) \\ &= \left[\prod_{j=1}^{m} \left(\prod_{i=1}^{b_j} p(\mu_{ij} \mid T_j)\right) p(T_j)\right] p(\sigma). \end{aligned}$$

$$p\left((T_1, M_1), \ldots, (T_m, M_m), \sigma\right) = \left[\prod_{j=1}^{m} \left(\prod_{i=1}^{b_j} p(\mu_{ij} \mid T_j)\right) p(T_j)\right] p(\sigma)$$

- $p(T_j) = \alpha(1+d)^{-\beta}$ formulated to regularize depth d.
- $p(\mu_{ij} \mid T_j)$ manipulated to shrink tree parameters μ_{ij} towards zero when number of trees m is large or when hyperparameter k is large.
  - "Weakens" individual trees, regularizing them to be "weak learners."
- $p(\sigma)$ seeks to avoid overconcentration and overdispersion by flexibly modeling σ about data estimate $\hat{\sigma}$ using hyperparameters ν and q.

2.3 Backfitting and Markov Chain Monte Carlo (MCMC)

Elaborated,

1. Draw (T_j, M_j) for the $j = 1, 2, \ldots, m$ trees conditionally on the rest of the trees, σ, and y.

   For each j,

   (a) Fix all trees except (T_j, M_j).

   (b) Propose a potential perturbation to the jth tree T_j,

   - GROW: Adding two child nodes
   - PRUNE: Pruning two child nodes
   - CHANGE: Adjust split of non-terminal rule[1]
   - SWAP: Swap between parent and child[2]

   and accept or reject the proposal via a Metropolis-Hasting step.

   (c) Draw terminal node parameters $\mu_{ij} \in M_j$ from a normal distribution.

2. Draw σ from the full conditional $\sigma \mid T_1, \ldots, T_m, M_1, \ldots, M_m, y$.

1. Draw (T_j, M_j) for the $j = 1, 2, \ldots, m$ trees conditionally on the rest of the trees, σ, and y.

   Note that since we are drawing conditional on the remaining trees and are thus fixing all those trees, hen the draw of (T_j, M_j) is equivalent to fitting the residuals of the model excluding (T_j, M_j), or

   $$R_j = y - \sum_{k \neq j} g(\mathbf{x}; T_k, M_k).$$

   Hence, the algorithm is a form of Bayesian backfitting.

2. Draw σ from the full conditional $\sigma \mid T_1, \ldots, T_m, M_1, \ldots, M_m, y$.

   This is dependent on the full model residuals

   $$R = y - \sum_{j=1}^{m} g(\mathbf{x}; T_j, M_j).$$

3 Application and Analysis of Performance

3.1 Introduction to the Dataset

This is a dataset of housing prices in King County, Washington. Covariates include:

- Number of Bedrooms
- Number of Bathrooms

[1] adjusted in `bartMachine`

[2] not included in `bartMachine`

- Size of living room in square feet
- Size of loft in square feet
- Number of Floors
- Whether or not the property is on the waterfront
- House condition
- House grade
- Year built

3.2 Introductory Data Analysis

Since BART does not perform as well with skewed outcome variables, we observed that the housing prices are very right-skewed, so we transformed the variable by taking a natural logarithm. As shown in Figure 1, the logarithm of our pricing variable has a distribution that is unimodal and nearly symmetric, giving evidence that it could be reasonably approximated by a normal distribution. Therefore, it would make the msot sense to use our methods on this transformation of the data.

3.3 Implementations of BART in `R`

One of the reasons we wanted to take a deeper look into this method is because there have already been packages written for `R` that can compute a BART model for a given regression problem.

While one could also use the original `BayesTree` package that works for this problem, we found that the package `bartMachine` was the most optimized. In order to be installed, however, your machine must be equipped for its `rJava` dependency.

3.3.1 Parameters for `bartMachine`

The key parameters include

- Number of Trees (`num_trees`)
- Number of burn-in iterations for MCMC backfitting; default is 250 (`num_burn_in`)
- Number of iterations after burn-in for MCMC backfitting; default is 1000 (`num_iterations_after_burn`
- Hyperparameters
  - α,β - Used in Tree prior for whether a node is nonterminal or not.
  - k - Determines prior probability that $\mathbb{E}(Y|\mathbf{X}) \in (y_{min}, y_{max})$ based on a normal distribution.
  - q - Determines the aggressiveness of the fit.
  - ν - Degrees of freedom for the inverse χ^2 prior.

3.4 Implementation of BART on Housing Data

In order to test the capabilities of BART and compare its predictive power, we ran many models on our machines with adjustments to each of the important parameters each time to see how each of them affected the predictive power for estimating log(House Price). We used the following parameters in `bartMachine()`:

- Number of Trees (`m`): 1,10,25,50,100,150,200,250,300
- q: 0.75 (Least Aggressive), 0.9 (Moderately Aggressive), 0.99 (Very Aggressive)
- ν: 3, 10
- k: 2, 3, 5

Typically, to assess performance we use MSE or a similar measure. However, since we are considering an outcome variable that has been transformed via the natural logarithm, a more natural measure is the root mean squared logarithmic error. Suppose we are attempting to estimate $\mathbf{Y} = (Y_1, Y_2, \cdots, Y_N)^\top$, a vector of outcome variables, with $\hat{\mathbf{Y}} = (\hat{Y}_1, \hat{Y}_2, \cdots, \hat{Y}_N)^\top$. Then, the root mean squared logarithmic error is defined as

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\Big(\log(Y_i) - \log(\hat{Y}_i)\Big)^2}$$

3.4.1 Finding Optimal Parameters for BART

Using cross-validation to find the optimal parameters and hyperparameters for BART, we can see in Figure 2 that there is a clear drop in out-of-sample error when we go from 50 trees in our model to 200 trees. This indicates that this is perhaps the most important parameter choice we will make when finding the optimal model. We can also see tha the optimal hyperparameter choices are $k = 5, q = 0.9$, and $\nu = 3$.

3.4.2 Effect of Adding Trees to our Model

Previously, we found that it matters a great deal how many trees we choose for our model. As a result, we measured the in-sample and out-of-sample error for BART models with different numbers of trees using cross-validation. The results for this procedure in Figure 3 show that there are diminishing returns in error minimization for increasing the number of trees in our model.

3.4.3 Visualization of BART Performance on Test Data

After finding our optimal model, we used the `predict` function in `R` to predict the values for log(Price). After exponentiating our results, we can compare them to the original data for housing prices. As seen in Figure 4, our model does a very good job esimating the conditional expectation for houses with lower prices, but as the price of the houses increases, our predictions are not quite as good.

3.4.4 Variable Performance

We can also see how the inclusion proportion changes for different numbers of trees in Figure 5.
A figure showing variable inclusion proportions is shown in Figure 6.
We can see how important variables were in other methods in Table 1

4 Performance of Other Methods

4.1 Description of Methods Used

For the sake of comparison, the most important methods we wanted to consider and compare BART to were Random Forest and XGBoost. BART is a very computationally expensive method, and its usefulness for an applied problem such as this one depends on how well it can do against other methods, but especially in comparison to other tree ensemble methods.

4.1.1 Random Forest

Random Forest is a supervised learning method that works bags trees, which are weak learners, and averages the results from each weak learner to make predictions. It is a randomized method because it randomly takes constructs many decision trees with the hope of obtaining a set of trees that overfit to certain subsets of the covariates in the problem. The idea behind this method is to control the variance of the overall model while preventing overfitting to the training set by averaging over a set of trees that are made up of different subsets of covariates. The important parameters for this model are the number of trees and the depth of each tree. Trees that grow very deep (i.e. require a lot of decisions) tend to learn too much from the training data and overfit, leading to low bias but high variance. Averaging multiple trees may increase the bias, but it will also reduce the variance. As mentioned previously, a single tree model will include high variance, so the random forest model accounts for this by averaging many trees that are trained on random subsets of covariates to produce an average result among weak learners. This prevents overfitting, and reduces the variance while inducing some bias.

4.1.2 Ridge and Lasso

Ridge Regression and Lasso regression are forms of regularized regression that we have studied in this class. Ridge and Lasso use the ℓ_2 and ℓ_1 norms, respectively, to prevent overfitting the model to the training set. These norms are utilized in the optimization of the typical Ordinary Least Squares problem in order to impose constraints on the problem to prevent the magnitude of the coefficients vector from becoming too large. In Ridge regression, the ℓ_2 norm leads to shrinkage for our regression coefficients. In Lasso regression, the ℓ_1 norm leads to sparsity in our coefficients vector, indicating that certain covariates are effectively removed from our model. For the dataset we are considering, we would expect that Lasso will be preferable since we have so many covariates.

4.1.3 XGBoost

XGBoost, or Extreme Gradient Boosting, is a supervised learning method that utilizes tree ensembles to predict an objective function value. As opposed to Random Forest, XGBoost does

not attempt to bat a set of trees that are weak learners, but rather seeks to set the parameters of each tree such that the objective loss function is minimized. In order to prevent overfitting, this loss function includes regularization constraints to diminish complexity similar to Ridge and Lasso. Rather than bagging weak learners, this method uses "boosting" for all of its learners. That is, each tree is developed and pruned in order to minimize our loss function under regularized constraints. In `R`, the `xgboost` library can be used for this method.

4.2 Comparison of Methods

We can see how Random Forest performed for this dataset in Table 2

A comparison between BART and various ML algorithms can be found in Table 3.

5 Conclusions

1. BART consistently outperformed other ML methods, even with various parameter combinations
2. Though BART performs very well, it is very computationally expensive
3. We saw evidence of diminishing returns in predictive power when increasing model complexity (i.e. adding more trees)
4. Likewise, MCMC convergence is obtained fairly quickly since increasing the number of iterations provides diminishing returns in predictive power.

6 Appendix

6.1 Figures

Figure 1

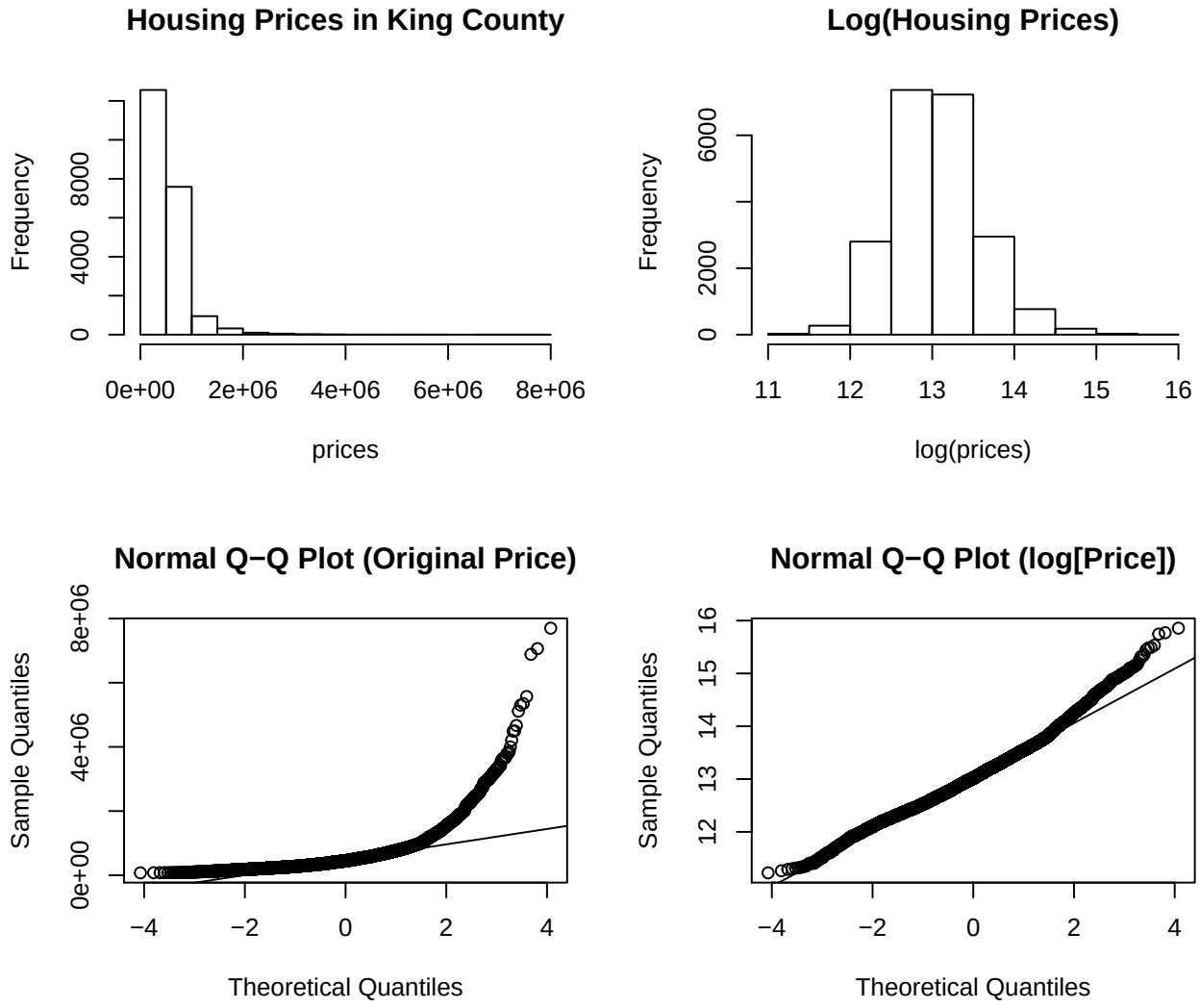


Figure 2

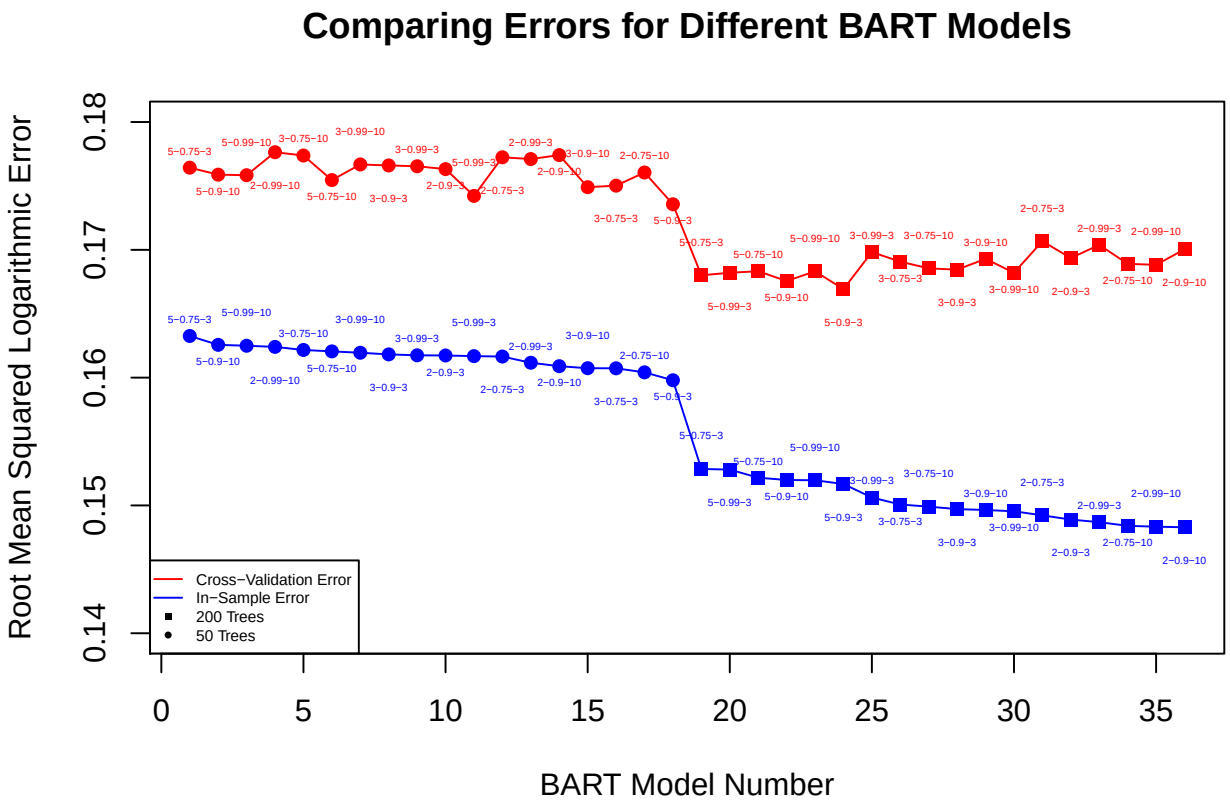

Comparing Errors for Different BART Models
Root Mean Squared Logarithmic Error
0.14
0.15
0.16
0.17
0.18
0
5
10
15
20
25
30
35
BART Model Number
Cross−Validation Error
In−Sample Error
200 Trees
50 Trees

Figure 3

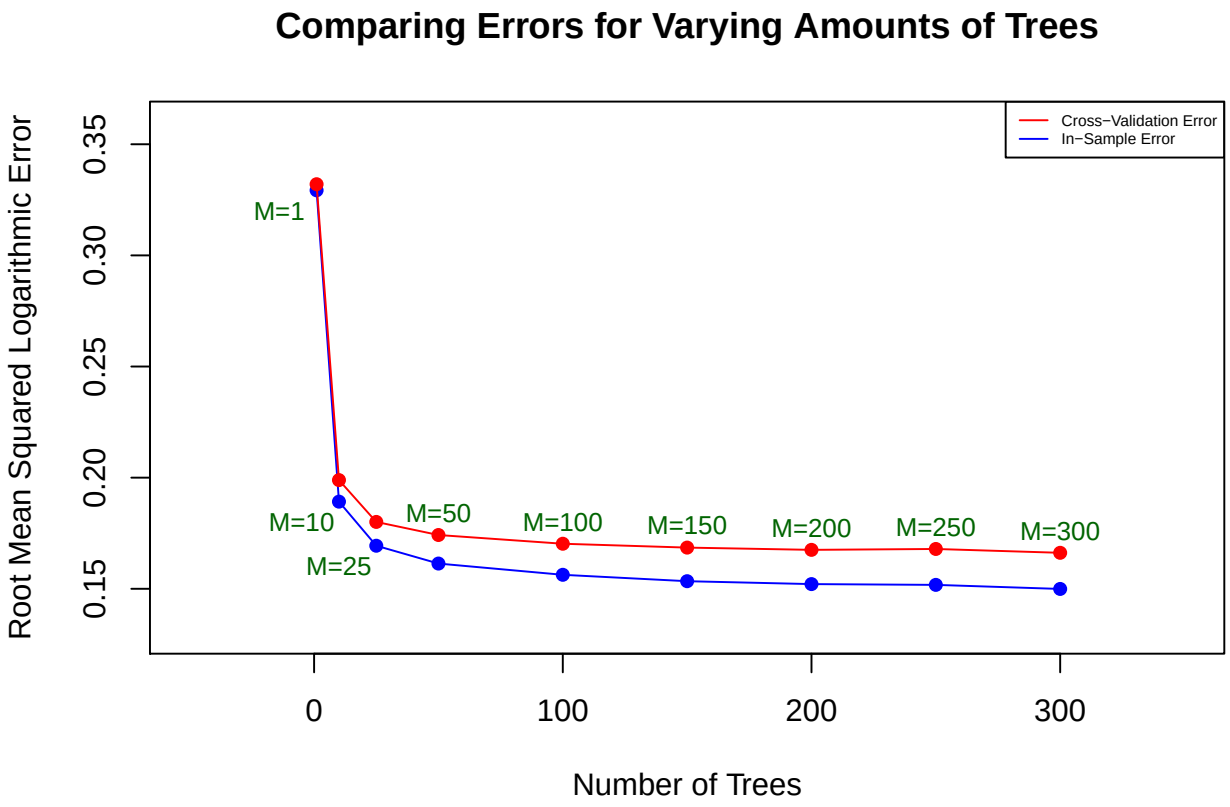

Comparing Errors for Varying Amounts of Trees
Root Mean Squared Logarithmic Error
0.15
0.20
0.25
0.30
0.35
Cross−Validation Error
In−Sample Error
M=1
M=10
M=25
M=50
M=100
M=150
M=200
M=250
M=300
0
100
200
300
Number of Trees

Figure 4

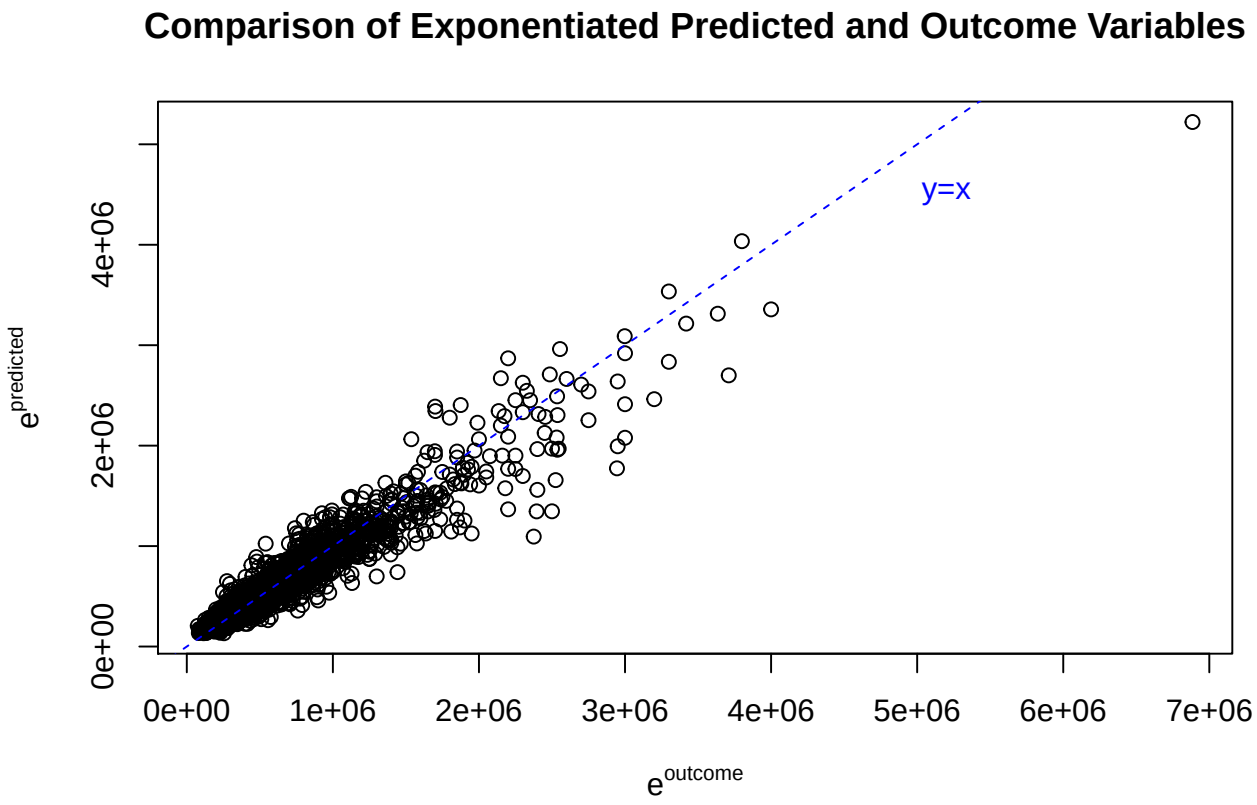


Figure 5

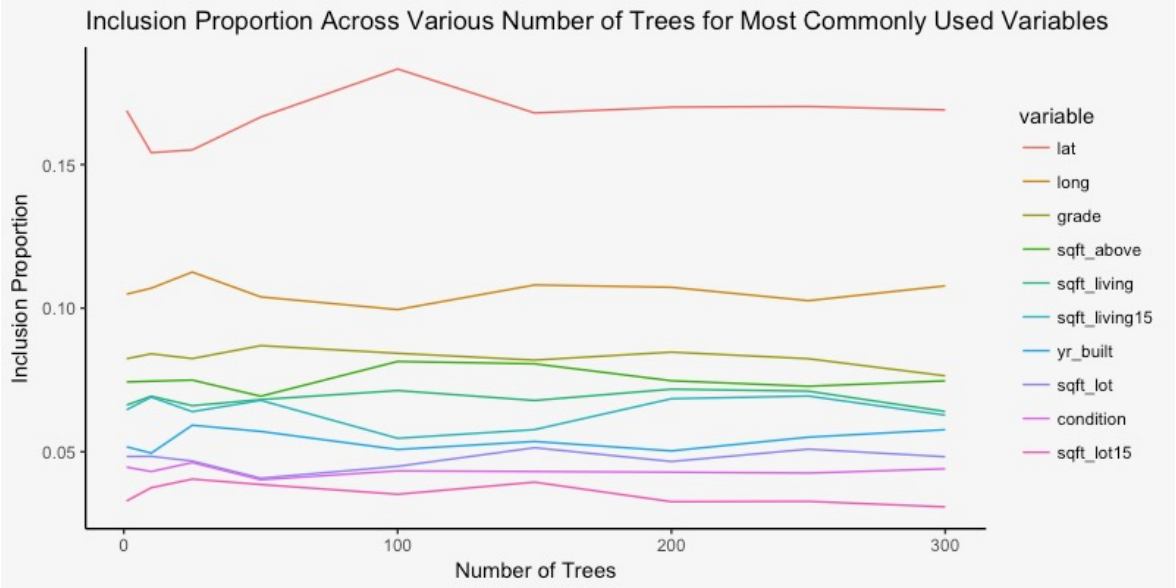
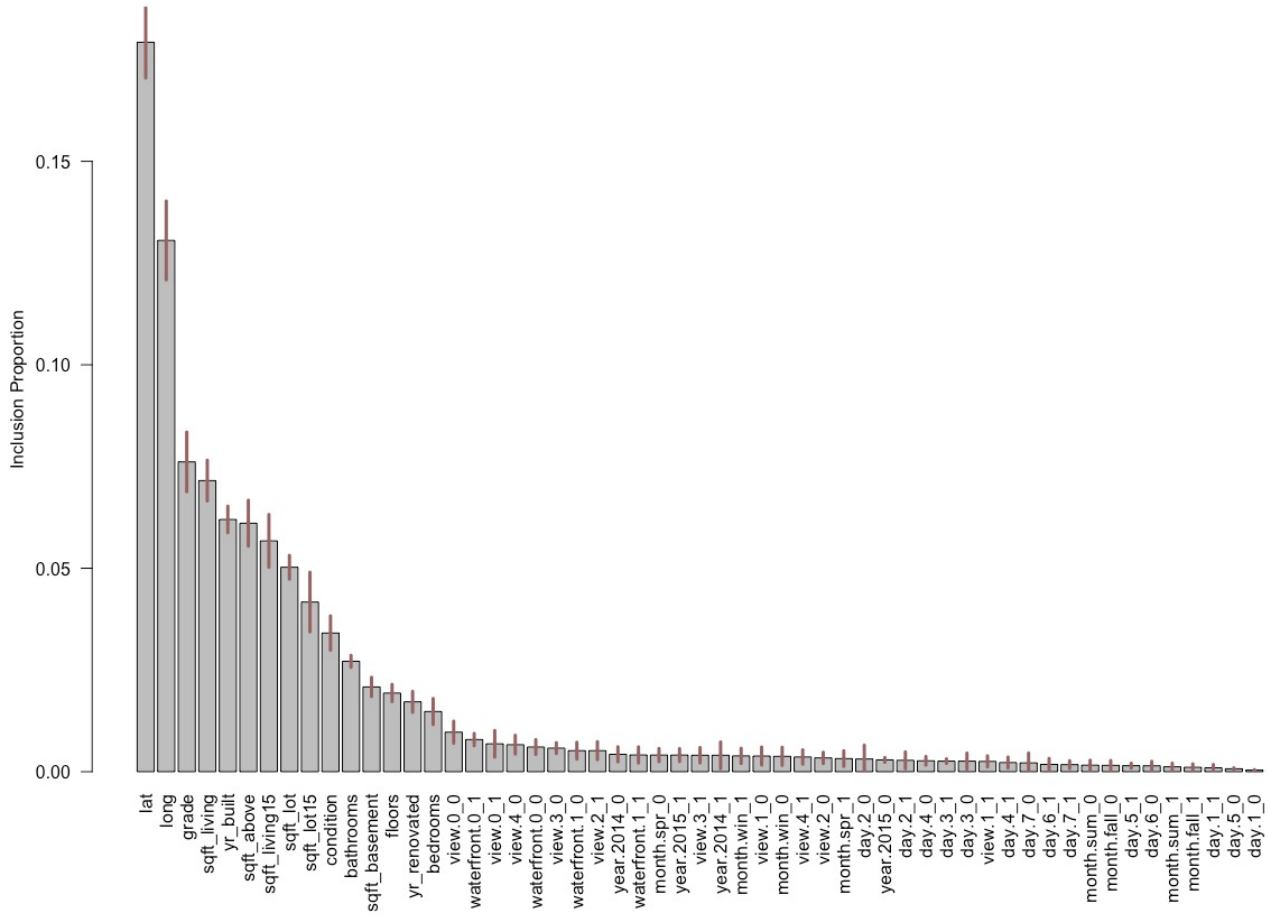


Figure 6



6.2 Tables

Table 1

Method	Var 1	Var 2	Var 3	Var 4	Var 5
BART	lat (0.198)	long (0.130)	grade (0.072)	sqft_living (0.070)	yr_built (0.060)
Random Forest	lat (1291)	grade (1090)	sqft_living (947)	sqft_living15 (220)	long (193)
Lasso	lat (1.351)	waterfront.00 (0.391)	grade (0.162)	view.01 (0.115)	view.41 (0.099)
XGBoost	lat (0.293)	grade (0.293)	sqrt_living (0.254)	long (0.041)	sqft_living15 (0.028)

Table 2

Random Forest Model	RMSLE
RF (trees=100, mtry=3.5)	0.191637587
RF (trees=100, mtry=8.75)	0.177500414
RF (trees=100, mtry=17.5)	0.173047312
RF (trees=100, mtry=35)	0.174472735
RF (trees=200, mtry=3.5)	0.191867067
RF (trees=200, mtry=8.75)	0.175589518
RF (trees=200, mtry=17.5)	0.171919974
RF (trees=200, mtry=35)	0.173349991
RF (trees=500, mtry=3.5)	0.191543125
RF (trees=500, mtry=8.75)	0.175664945
RF (trees=500, mtry=17.5)	**0.171916452**
RF (trees=500, mtry=35)	0.173200693

Table 3

Model	Parameters	Test RMSLE
BART (1,000 iterations)	ν=3,q=0.9,k=5,m=200	0.1587636
BART (3,000 iterations)	ν=3,q=0.9,k=5,m=200	0.1566555
BART (5,000 iterations)	ν=3,q=0.9,k=5,m=200	0.155946
BART (10,000 iterations)	ν=3,q=0.9,k=5,m=200	**0.1543836**
Ridge	λ=0.04068241	0.2768564
Lasso	λ=0.000604121	0.2520706
XGBoost	trees=200,η=0.25,depth=4	0.1660505
Random Forest	trees=500,`mtry`=17.5	0.1719165