Computational Data Analysis

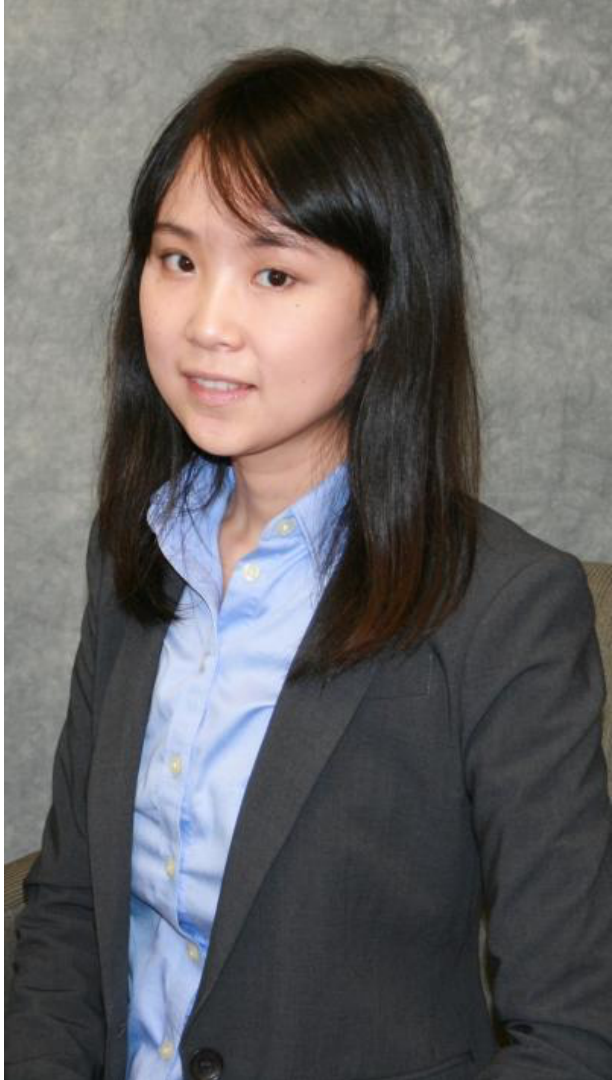
Machine Learning

Yao Xie, Ph.D.

Associate Professor

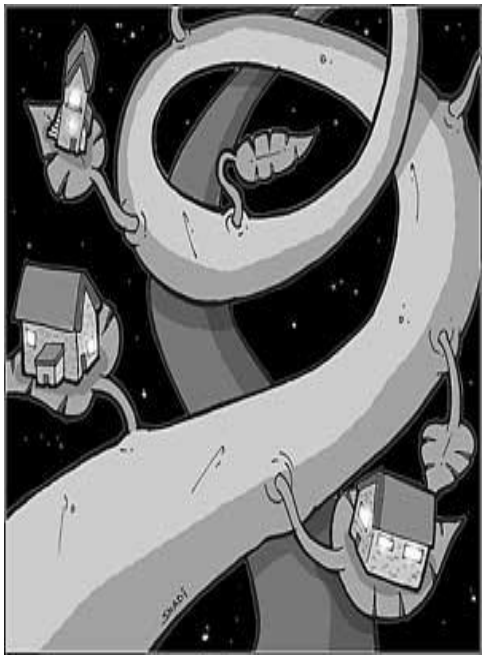
Harold R. and Mary Anne Nash Early Career Professor

H. Milton Stewart School of Industrial and Systems Engineering



Regression and Random Forest

Machine learning for apartment hunting

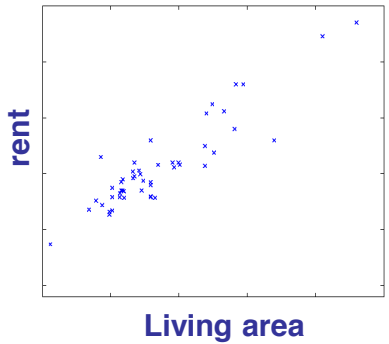


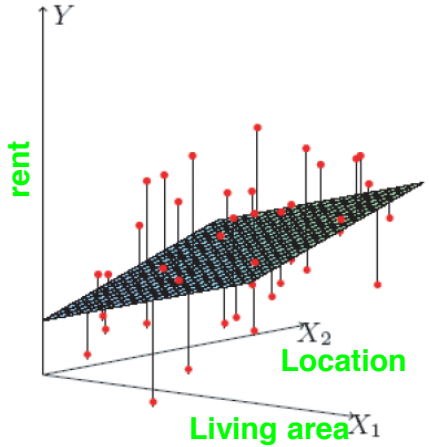
- Suppose you are to move to Atlanta
- And you want to find the **most reasonably priced** apartment satisfying your **needs:**

  square-ft., # of bedroom, distance to campus …

Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

The learning problem





- Features:
  - Living area, distance to campus, # bedroom …
  - Denote as $x = (x_1, x_2, \dots, x_n)^\top$
- Target:
  - Rent
  - Denoted as y
- Training set:
  - $X = (x^1, x^2, \dots x^m)$
  - $y = (y^1, y^2, \dots, y^m)^\top$

Linear Regression Model

- Assume y is a linear function of x (features) plus noise ϵ

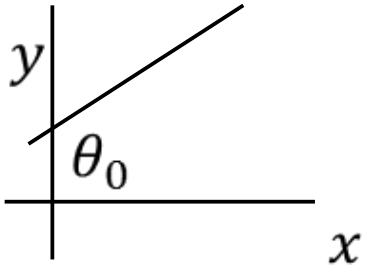
$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n + \epsilon$$

- where ϵ is an error term of unmodeled effects or random noise

- Let $\theta = (\theta_0, \theta_1, \ldots, \theta_n)^\top$, and augment data by one dimension

$$x \leftarrow (1, x)^\top$$

- Then $y = \theta^\top x + \epsilon$



Least mean square method

- Given m data points, find θ that minimizes the mean square error

$$\hat{\theta} = argmin_{\theta}\, L(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top x^i\right)^2$$

- Our usual trick: set gradient to 0 and find parameter

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top x^i\right)x^i = 0$$

$$\Leftrightarrow -\frac{2}{m}\sum_{i=1}^{m} y^i x^i + \frac{2}{m}\sum_{i=1}^{m} x^i x^{i\top}\theta = 0$$

Matrix version of the gradient

- $\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}\sum_{i=1}^{m} y^i x^i + \frac{2}{m}\sum_{i=1}^{m} x^i x^{i\top}\theta = 0$

- Equivalent to

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}(x^1 \dots, x^m)(y^1 \dots, y^m)^\top + \frac{2}{m}(x^1, \dots x^m)(x^1, \dots x^m)^\top\theta = 0$$

- Define $X = (x^1, x^2, \dots x^m), y = (y^1, y^2, \dots, y^m)^\top$, gradient becomes

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m} Xy + \frac{2}{m} XX^\top\theta = 0$$

$$\Rightarrow \hat{\theta} = (XX^\top)^{-1}Xy$$

Alternative way of obtaining θ

- The matrix inversion in $\hat{\theta} = (XX^\top)^{-1}Xy$ can be very expensive to compute

- $\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top x^i\right)x^i$

- Gradient descent

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \frac{\alpha}{m}\sum_{i=1}^{m}\left(y^i - \hat{\theta}^{t\top} x^i\right)x^i$$

- Stochastic gradient descent (use one data point at a time)

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^t + \beta_t\left(y^i - \hat{\theta}^{t\top} x^i\right)x^i$$

A recap:

- Stochastic gradient update rule

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} + \beta \left(y^{i} - \hat{\theta}^{t^{\top}} x^{i} \right) x^{i}$$

  - Pros: on-line, low per-step cost
  - Cons: coordinate, maybe slow-converging
- Gradient descent

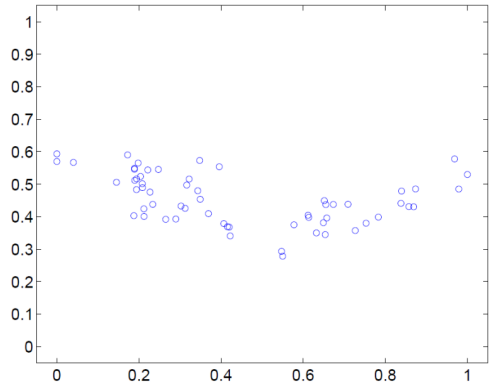
$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} + \frac{\alpha}{m} \sum_{i=1}^{m} \left(y^{i} - \hat{\theta}^{t^{\top}} x^{i} \right) x^{i}$$

  - Pros: fast-converging, easy to implement
  - Cons: need to read all data
- Solve normal equations

$$(XX^{\top})\hat{\theta} = Xy$$

  - Pros: a single-shot algorithm! Easiest to implement.
  - Cons: need to compute inverse $(X^{T}X)^{-1}$, expensive, numerical issues (e.g., matrix is singular ..)

Nonlinear regression



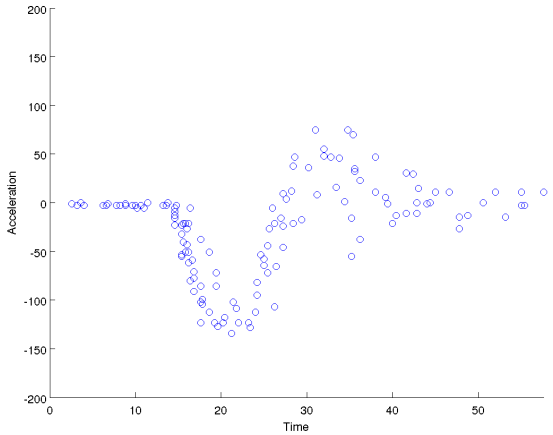
- Want to fit a polynomial regression model

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n + \epsilon$$

- Let $\tilde{x} = (1, x, x^2, \ldots, x^n)^\top$ and $\theta = (\theta_0, \theta_1, \theta_2, \ldots, \theta_n)^\top$

$$\mathrm{y} = \theta^\top \tilde{x}$$

Example: head acceleration in accident

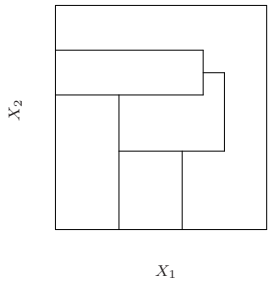


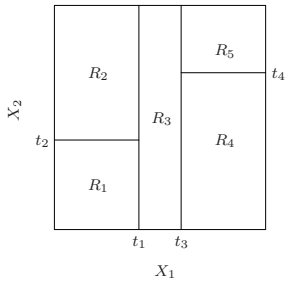
http://www.jari.jp/Portals/0/resource/pdf/AAI%20Summit/H25/4.%20MIROS.pdf

Regression tree

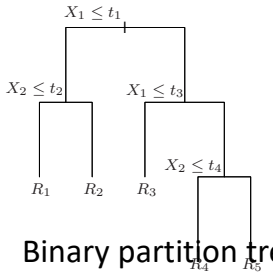
- Regression tree partition the feature space into a set of rectangles
- Fit a simple model (like a constant) in each one
- Popular method: CART

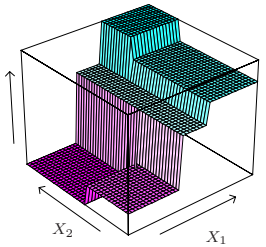
$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}.$$





$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m)$$





Binary partition tree

Greedy procedure

- Greedy approach
- Start with all of the data
- Consider a splitting variable j and split point s, and define half-spaces

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

- Solve the problem

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

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j,s))$$

$$\hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j,s)).$$

Greedy procedure

- How large should the tree be
- Very large tree may over-fit the data; very small tree does not capture the important structure
- Determine the tree depth by controlling the fitting error-complexity tradeoff
- Grow a large tree, stop when a minimum node size has reached, then performing pruning by minimizing the cost function

error

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|.$$

complexity

$$N_m = \#\{x_i \in R_m\},$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i,$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2,$$

Classification tree

- The classification outcome taking value 1,2,…, K
- The only change needed in the tree algorithm is to change the criteria for splitting nodes and pruning the tree

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k),$$

$$k(m) = \arg\max_k \hat{p}_{mk},$$

- Different measures of "error" $Q_m(T)$

Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}.$

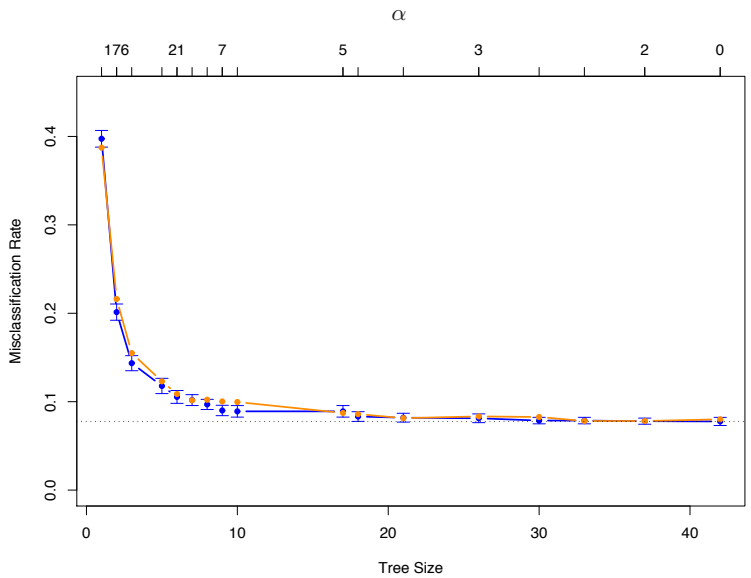
Gini index: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$

Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$

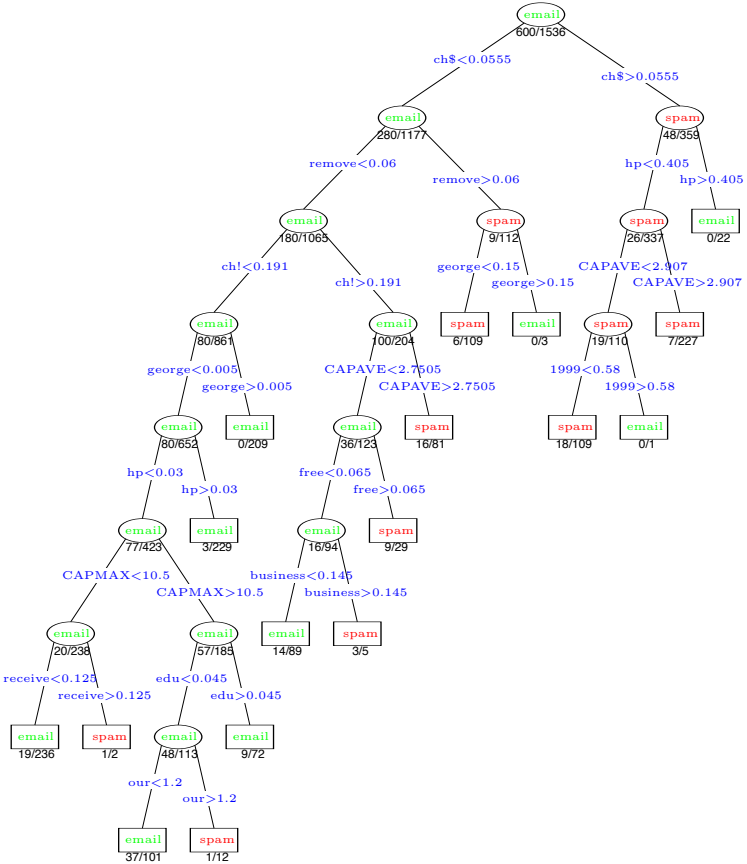
Example: Email spam

TABLE 9.3. *Spam data: confusion rates for the* 17*-node tree (chosen by cross–validation) on the test data. Overall error rate is* 9.3%.

	Predicted	
True	email	spam
email	57.3%	4.0%
spam	5.3%	33.4%

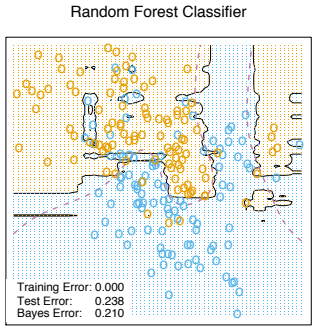


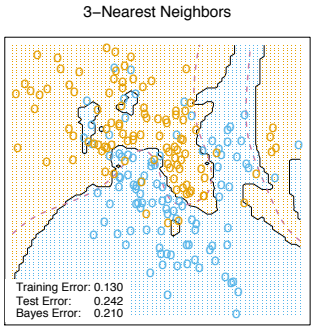
Example: Email spam

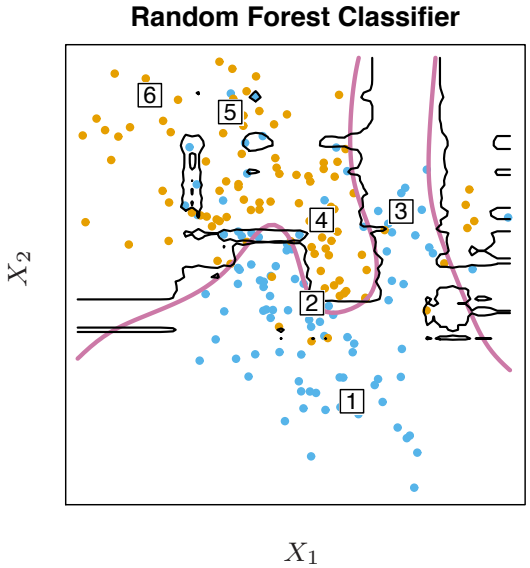


Random forest

- A main issue of the tree-based method is large variance
- Trees (if grown deep enough) have low bias: can capture complicated structure
- Trees are known to be very noisy
- They benefit greatly from averaging

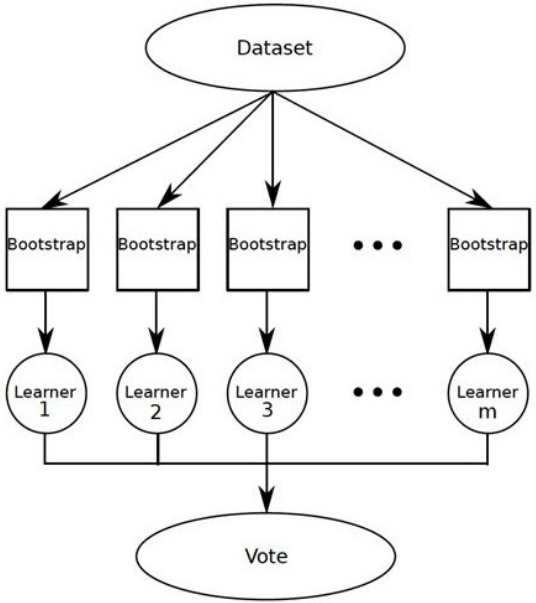






Overview

- Bagging or bootstrap aggregation is an approach to reduce variance of the prediction
- Random forest is a type of bagging algorithm that builds a large collection of de-correlated trees, and then average them.
- For many problems, performance of random forest is very similar to boosting, and they are simpler to train and tune.



California housing price

- Data set (Pace and Barry, 1997) is available from CMU StatLib
- Aggregated data from each of 20,460 neighborhood (1990 census data) in California
- 8 predictors
  - Median income
  - House density
  - Average occupancy
  - Location
  - Average number of rooms
  - Average number of bedrooms

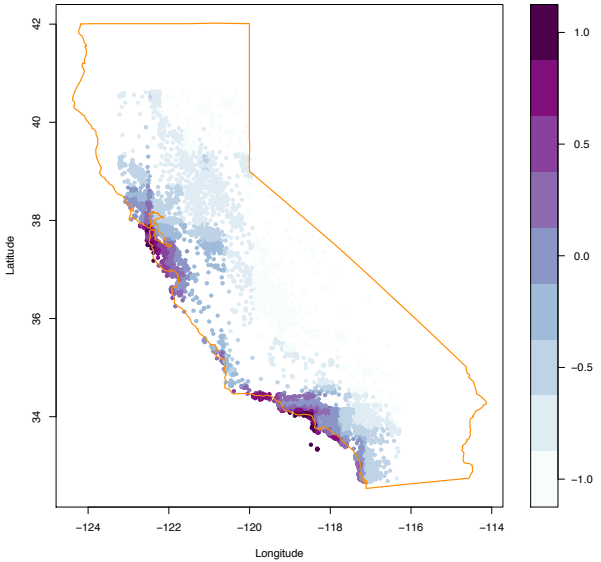
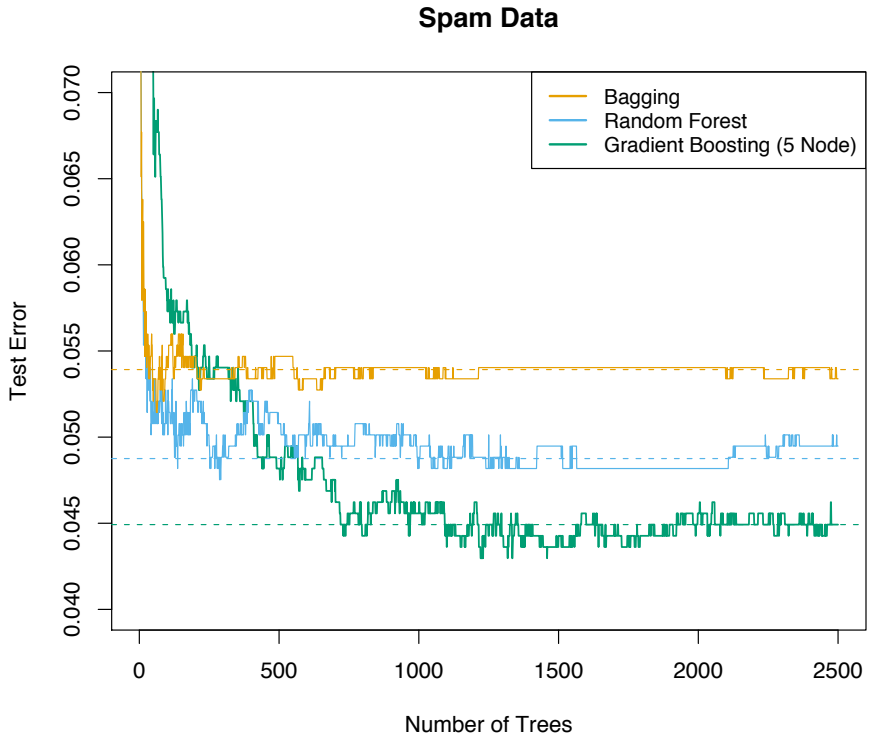


FIGURE 10.17. *Partial dependence of median house value on location in California. One unit is* $100,000, *at* 1990 *prices, and the values plotted are relative to the overall median of* $180,000.

California housing price




SUMMARY