Prophet model

Prophet model

Prophet: forecasting model developed by Facebook Data Science Team

Let us take a time series model with trend, seasonality and other components (e.g. holidays)

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

This is a specification similar to a generalized additive model, GAM

Prophet

Some advantages of this formulation:

- Flexibility
- Interpretability
- Easy to manage

Prophet
Trend

Often the trend component is modeled according to a logistic equation

$$g(t) = \frac{C}{1 + e^{-k(t)}}$$

where C is the carrying capacity and k is the growth rate
The trend can also be linear or constant.

Prophet

Change points

- It is possible to incorporate changes in the trend with change points, to account for a non-constant growth rate
- Suppose there are S change points at $s_j, j = 1, \ldots, S$
- We may define a vector of adjustments δ, where δ_j is the change at time s_j.
- So the growth rate is given by k plus adjustments: $k + \sum_{j:t>s} \delta_j$
- Change points can be manually specified by the analyst, given prior knowledge, or automatically selected.

Prophet

Seasonality

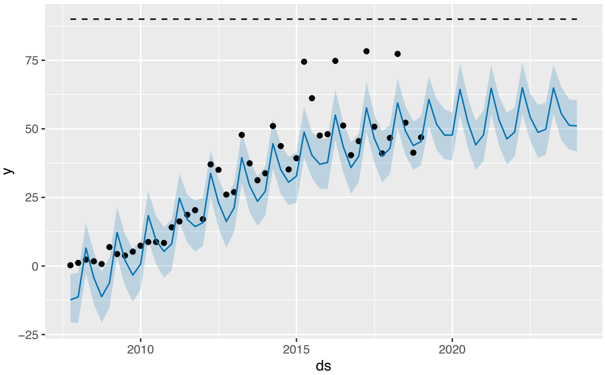
The seasonality component may be defined with a combination of Fourier series

$$s(t) = \sum_{n=1}^{N} \left(a \cos\left(\frac{2\pi n t}{P}\right) + b \sin\left(\frac{2\pi n t}{P}\right) \right)$$

Prophet

Example: Apple iPhone

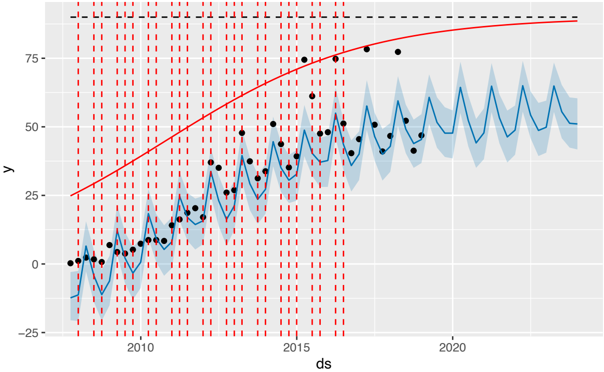
Nonlinear logistic trend and additive seasonality



Prophet

Example: Apple iPhone

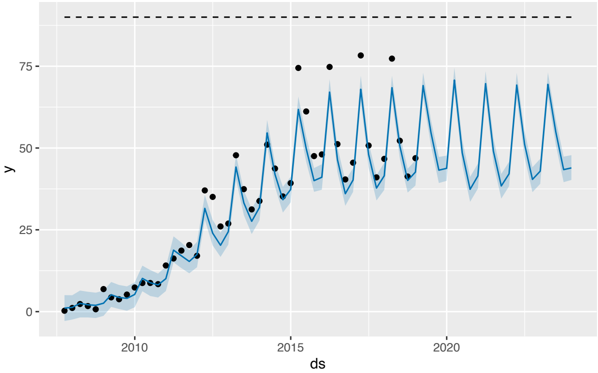
Change points in the first 80% datapoints



Prophet

Example: Apple iPhone

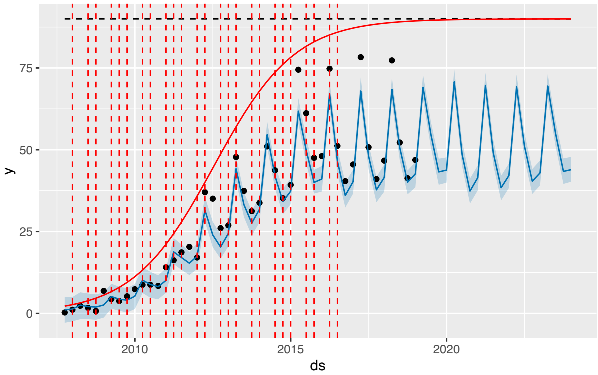
Nonlinear logistic trend and multiplicative seasonality



Prophet

Example: Apple iPhone

Change points



Gradient Boosting

Boosting

- Initially developed for classification problems, later extended to regression problems.
- Idea: assign more weight to observations badly classified, to make the model work more on these $\rightarrow$ AdaBoost
- Bagging, Boosting and Random Forests use trees as building blocks to construct more powerful models.

Gradient Boosting

- Powerful algorithm of machine learning
- Employed for both regression and classification problems
- Gradient Boosting = Gradient Descent + Boosting

Gradient Boosting: intuition

Let us consider a simple regression problem . . . with a simple case: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
We want to estimate a model $y = f(x)$ minimizing a loss function, i.e. Mean Squared Error.
Suppose that we have a good model f, but we notice some errors: $f(x_1) = 0.8$ while $y_1 = 0.9$, $f(x_2) = 1.4$ while $y_2 = 1.3$.

How can we improve the model?

Gradient Boosting: intuition

Consider that:

- we can not modify f
- but we can add to f another model, such as regression tree, h,
- so that the new prediction will be
  $y_i = f(x_i) + h(x_i)$

Gradient Boosting: intuition

The prediction is updated as follows:

$$
\begin{aligned}
f(x_1) + h(x_1) &= y_1 \\
f(x_2) + h(x_2) &= y_2 \\
&\vdots \\
f(x_n) + h(x_n) &= y_n.
\end{aligned}
$$

But we can also write

$$
\begin{aligned}
y_1 - f(x_1) &= h(x_1) \\
y_2 - f(x_2) &= h(x_2) \\
&\vdots \\
y_n - f(x_n) &= h(x_n)
\end{aligned}
$$

where $r(x_i) = y_i - f(x_i)$ are the residuals

Gradient Boosting: intuition

- Gradient Boosting $\rightarrow$ fit a regression tree, h, on data $(x_1, r_1), (x_2, r_2), \ldots, (x_n, r_n)$ to improve the prediction
- the role of h is to compensate the 'problems' of model f

Gradient Boosting: intuition

So we have a new model for y, which should be better than the previous one:

$$f_2(x) = f_1(x) + h_1(x)$$

and we can repeat this reasoning obtaining the residuals with respect to this new model $f_2(\cdot)$ and fit a new tree $h_2(x_i)$ to further improve the prediction.
Thus the prediction will be

$$f_3(x) = f_2(x) + h_2(x)$$

We can repeat this M times and at each iteration $1 < m < M$ we will have

$$f_{m+1}(x) = f_m(x) + h_m(x)$$

How is this related to the Gradient Descent?

Gradient Boosting

How is this related to the Gradient Descent?
Let us consider the quadratic loss function

$$L(y, f(x)) = \frac{1}{2}(y - f(x))^2$$

We want to minimize $J = \sum_i L(y_i, f(x_i))$

$$\frac{\partial J}{\partial f(x_i)} = \frac{\partial \sum_i L(y_i, f(x_i))}{\partial f(x_i)} = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} = f(x_i) - y_i$$

We can see the residuals as negative gradients

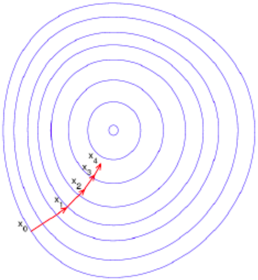
$$-g(x_i) = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right] = y_i - f(x_i)$$

Gradient Boosting

Gradient Descent

Minimizes a function going in the opposite direction with respect to the gradient

$$\vartheta_{m+1} = \vartheta_m - \rho \frac{\partial J}{\partial \vartheta_m}$$



Gradient Boosting

How is this related to the Gradient Descent?

For a regression problem with quadratic loss function,

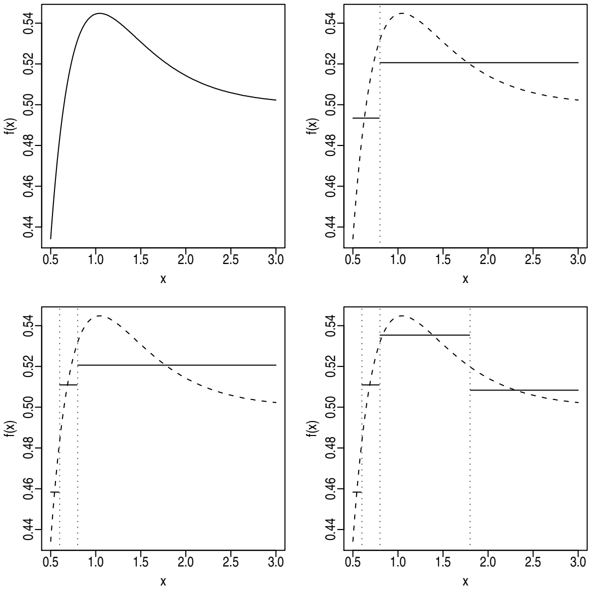
- residual $\leftrightarrow$ negative gradient
- fit h to the residual $\leftrightarrow$ fit h to the negative gradient
- update f through the residual $\leftrightarrow$ update f through the negative gradient

We are using the negative gradient

Step function

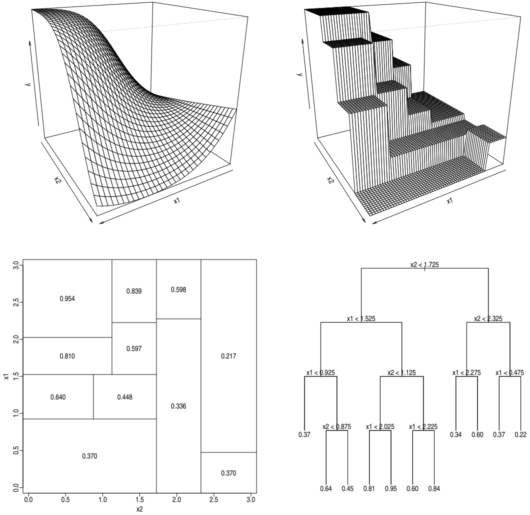
- In one sense, the simplest way to approximate a generic function $f(x)$ is to use a step function, that is, a piecewise constant function

Regression trees



Regression trees

- The previous scheme can be extended to the case of functions $f(x)$ of p variables, $x = (x_1, \ldots, x_p)$
- To keep simple the method we require that the regions with constant values are rectangles, the sides of which are parallel to the coordinate axes.
- This approximate function may be represented as a binary tree



Regression tree

- We want to estimate regression curve $f(x)$ underlying the data by

$$\hat{f}(x) = \sum_{j=1}^{J} c_j \, I(x \in R_j)$$

  where $I(x \in A)$ is the *indicator function* of the set A (and here they are rectangles) and $c_1, \ldots, c_J$ are constants
- objective function: *deviance*,

$$D = \sum_i \{y_i - \hat{f}(x_i)\}^2$$

Regression tree

- This minimization, even if we fix the number of steps J, involves very complex computation
- operatively we follow a suboptimal approach of step-by-step optimization: we construct a sequence of gradually more refined approximations and to each of these we minimize the deviance relative to the passage from the current approximation to the previous one.
- It is not ensured that we get the global maximum. This procedure is called greedy-algorithm or myopic optimization
- This operation is represented by a series of binary splits
- Each internal node represents a value query on one of the variables – e.g. '*Is* $x_3 > 0.4$*?*'. If the answer is '*Yes*', go right, else go left.
- The terminal nodes are the decision nodes. Typically each terminal node is assigned a value, c_h, given by the arithmetic mean of the observed y_i having component x_j falling in this node.

Gradient Boosting: Algorithm

A Gradient Boosting may be defined with these input elements:

- training set $(x_i, y_i) \ldots (x_n, y_n)$
- loss function $L(y, f(x))$
- number of iterations M

Gradient Boosting: Algorithm

Gradient Tree Boosting algorithm

- initialize the model with a constant value

$$f_0(x) = \text{arg}\min_{\gamma} \frac{1}{n}\sum_{i=1}^{n} L(y_i, \gamma)$$

  with quadratic loss function we have $f_0 = \bar{y}$

- at each iteration $1 < m < M$ calculate the negative gradients for $i = 1, 2, \ldots, n$

$$-g(x_i) = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right] = y_i - f(x_i)$$

Gradient Boosting: Algorithm

. . . continued

- estimate a regression tree $h_m(x)$ on $-g(x_i)$ giving terminal regions $R_{jm}, j = 1, 2, \ldots, J_m$
- for $j = 1, 2, \ldots, J_m$ calculate
  $\gamma_{jm} = \arg\min \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
- update the model $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$
- Output: $\hat{f}(x) = f_M(x)$

Note: we use the negative gradients because we can use loss functions other than the quadratic loss and derive the corresponding algorithms

Gradient Boosting

Why should we use different loss functions?
Quadratic loss function is:

- simple to handle mathematically ...
- not robust with respect to outliers

y_i	0.5	1.2	2	5*
$f(x_i)$	0.6	1.4	1.5	1.7
$L(y-f)^2/2$	0.005	0.02	0.125	5.445

$\rightarrow$ The presence of an outlier may have negative effects on the general performance of the model

Gradient Boosting

Other loss functions

- absolute loss function

$$L(y, f) = |y - f|$$

- Huber loss function → more robust with respect to outliers

$$L(y, f) = \begin{cases} \{1/2(y - f)^2 & |y - f| \leqslant \delta \\ \delta(|y - f| - \delta/2) & |y - f| > \delta \end{cases}$$

y_i	0.5	1.2	2	5*
$f(x_i)$	0.6	1.4	1.5	1.7
quadratic	0.005	0.02	0.125	5.445
absolute	0.1	0.2	0.5	3.3
Huber($\delta = 0.5$)	0.005	0.02	0.125	1.525

Gradient Boosting: regularization

As in other models, also in the case of the Gradient Boosting we can introduce some regularization techniques, in order to reduce the risk of overfitting.

Shrinkage

The update rule is modified in this way

$$f_m(x) = f_{m-1}(x) + \nu \cdot \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm})$$

Parameter $0 < \nu < 1$ controls the 'learning rate' of the boosting procedure.

Smaller values of $\nu \to$ more *shrinkage* $\to M$ bigger

Trade-off between ν and M.

Gradient Boosting

Why Gradient Boosting?

- use of ‘mixed’ data
- robust to outliers in input
- interpretability of results
- prediction power

Gradient Boosting

Comparison among models

MART $\rightarrow$ Gradient Boosting

Some characteristics of different learning methods.
Key: 🟢= good, 🟡=fair, and 🔴=poor.

Characteristic	Neural Nets	SVM	CART	GAM	KNN, kernels	MART
Natural handling of data of "mixed" type	🔴	🔴	🟢	🟢	🔴	🟢
Handling of missing values	🔴	🔴	🟢	🔴	🟢	🟢
Robustness to outliers in input space	🔴	🔴	🟢	🟡	🟢	🟢
Insensitive to monotone transformations of inputs	🔴	🔴	🟢	🟢	🔴	🟢
Computational scalability (large N)	🔴	🔴	🟢	🟢	🔴	🟢
Ability to deal with irrelevant inputs	🔴	🔴	🟢	🟡	🔴	🟢
Ability to extract linear combinations of features	🟢	🟢	🔴	🔴	🟡	🟡
Interpretability	🔴	🔴	🟡	🟢	🔴	🟢
Predictive power	🟢	🟢	🔴	🔴	🟢	🟢

Gradient Boosting: example

- Data set on house prices in California
- y= median price in hundreds of thousands dollars
- demographic variables: average income (MedInc), house density (House), average number of people per house (AveOcc), population (Population)
- house features: latitude, longitude (latitude, longitude), average number of rooms (AveRooms) average number of bedrooms (AveBedrms), age of the house (HouseAge)
- 8 variables

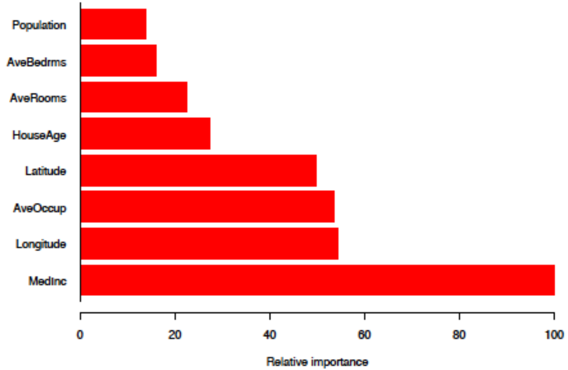
Gradient Boosting: example

Visualizing results

- relative influence plot: reduction in squared error due to each variable

Gradient Boosting: example

Gradient Boosting with tree depth= 6, shrinkage= 0.1, loss function= Huber



Case study

- What are the drivers of house prices?
- Hedonic price modelling: house prices depend on their characteristics (size, number of rooms, type of house . . .)
- anything else? we may think that there are other factors giving value to a house, and therefore to its price.

Case study

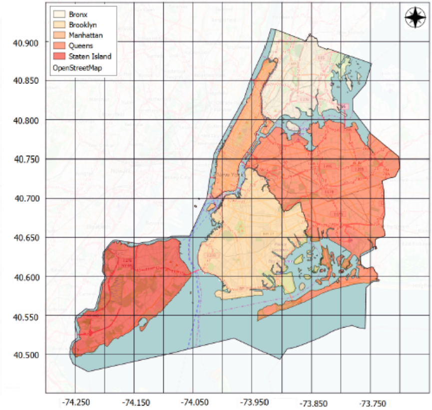
The ‘hedonic’ approach has been used to explain differences in hotel prices . . . and more recently to study the phenomenon of Airbnb in different cities around the world

- Airbnb: example of sharing economy
- Business opportunity or threat?
- Unfair competition towards hotels?
- Increasing rent prices?
- What are the factors determining Airbnb house prices?

Case study

We are in New York in 2019

we want ot understand what are the factors determining prices on Airbnb . . .



Case study

We would like to account for 3 major points:

- Airbnb diffusion
- heterogeneous districts
- availability of Open Data

Open Data: why?

We would like to account for ‘external’ information
Open Data are

- accessible
- available
- integrable
- updated periodically
- machine readable

Open Data in New York

From the website `https://opendata.cityofnewyork.us/` we may collect information referring to:

variables	*source*
major attractions	Dept Finance
hotels	NYC open data
restaurants	NYC open data
metro	Metro trans authority
spare time	NYC open data
helath services	NYC open data
crime	NY Police Dept

. . . and much more.

Airbnb in New York

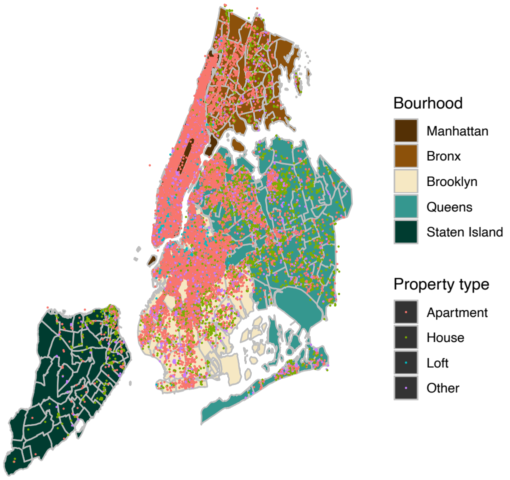
- The website `insideairbnb.com` contains the 'listings' of Airbnb houses for many cities around the world.
  For each listing, many variables are available referring to
  - property,
  - host,
  - *guest reviews*,
  - terms of service.
- of course there is also the variable 'price per night'
- what are the variables that play a major role in price determination? ...

Airbnb in New York

- We also want to take care of Open Data
- crime rate, distance from touristic attractions, metro stations ... do they have a role?
- Data Integration

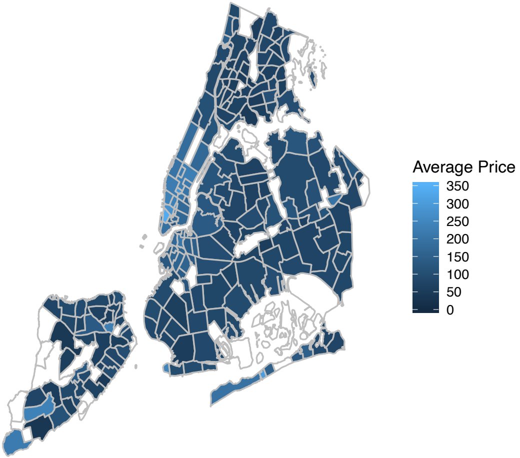
Airbnb in New York

Property position



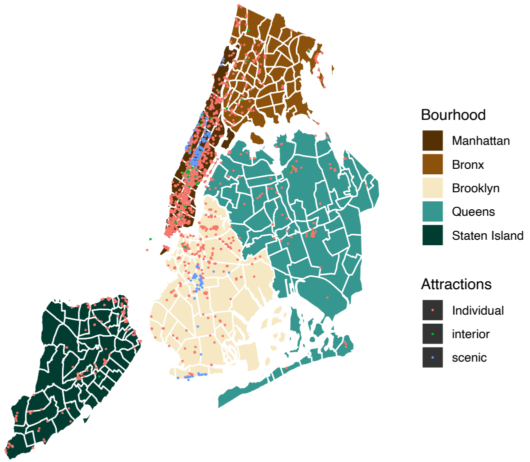
Airbnb in New York

Average price of houses



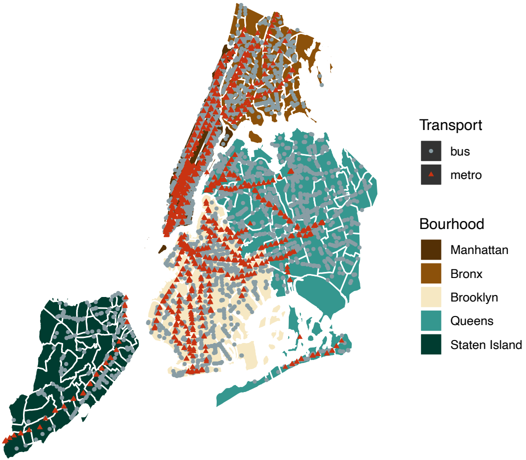
Airbnb in New York

Main attractions



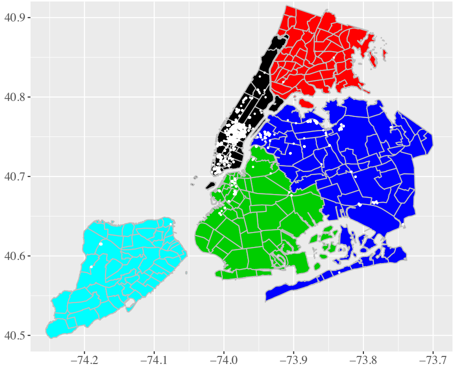
Open Data in New York

Public transport



Open Data in New York

Hotel position



Gradient Boosting for Airbnb prices

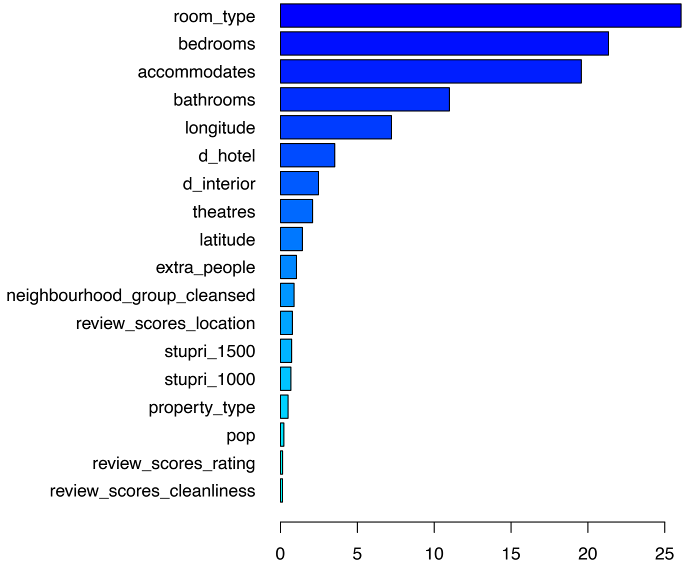
Gradient Boosting for Airbnb prices

Airbnb house prices in New York with GB

- large dataset: 77000 obs, 107 variables
- response: price/night
- training set: 50000
- initial model
  iterations= 100, tree depth= 1 (stump), shrinkage= 0.1
- other options are possible by modifying tuning parameters

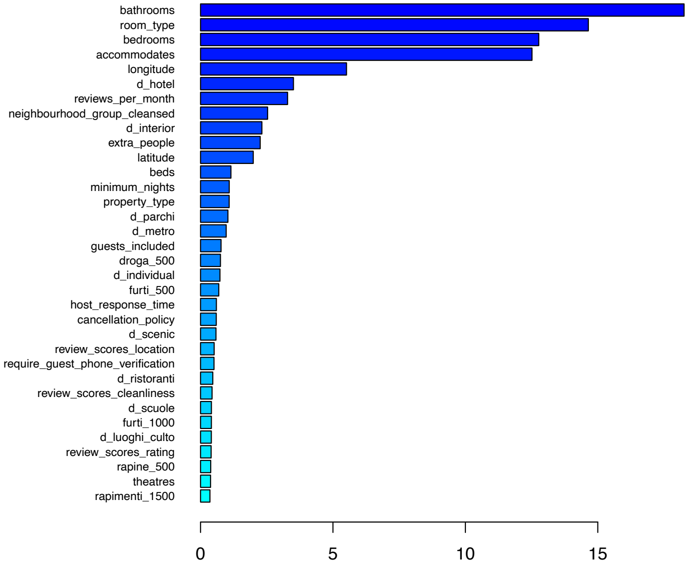
Gradient Boosting for Airbnb prices

iterations= 100, depth= 1, shrinkage= 0.1



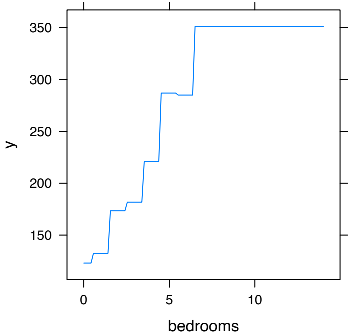
Gradient Boosting for Airbnb prices

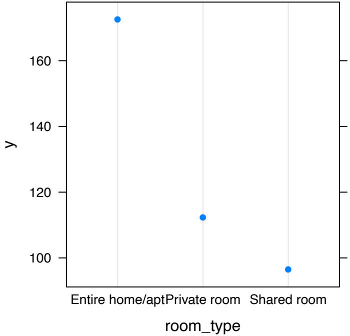
iterations= 180, depth= 4, shrinkage= 0.2



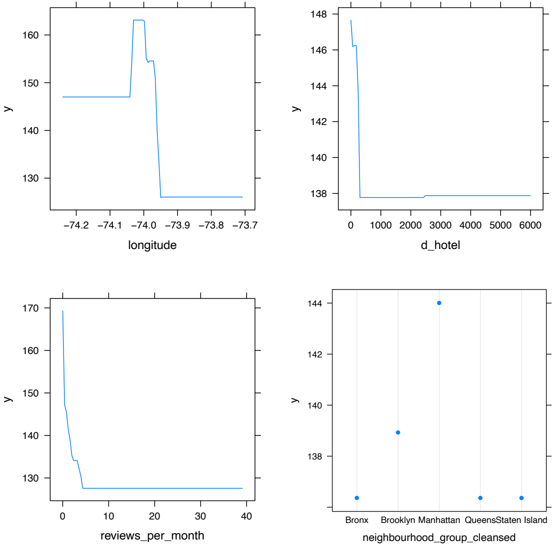
Gradient Boosting for Airbnb prices

Partial dependence plots: illustrate the marginal effect of the selected variables on the response after integrating the other variables.

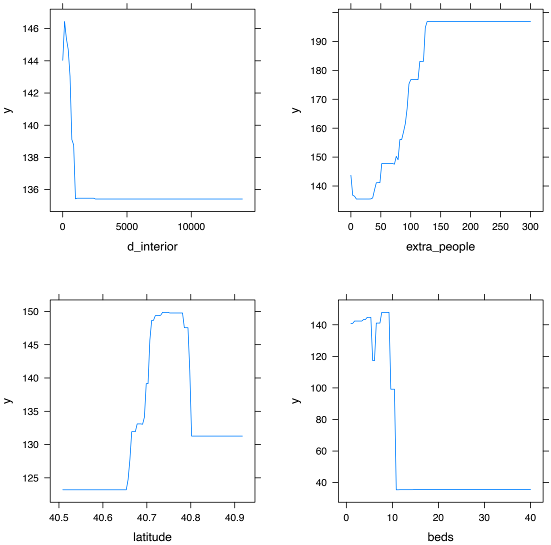




Gradient Boosting for Airbnb prices



Gradient Boosting for Airbnb prices



Gradient Boosting for Airbnb prices