# 多元统计分析—R 与 Python 的实现

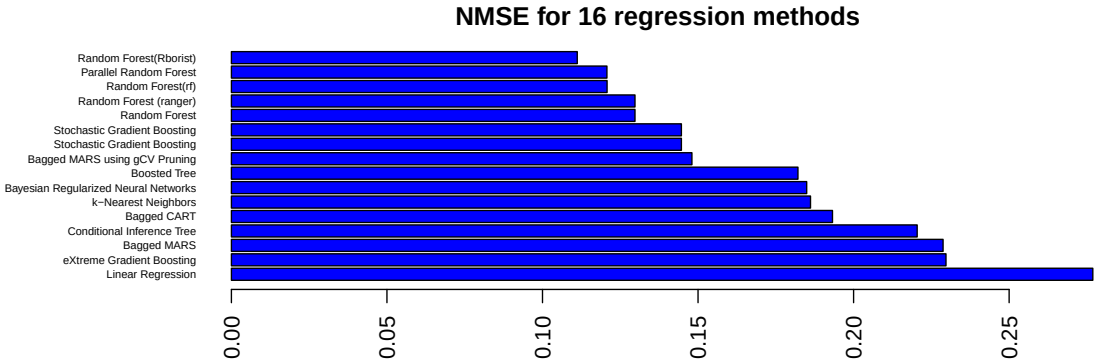
## 回归

吴喜之

June 28, 2019

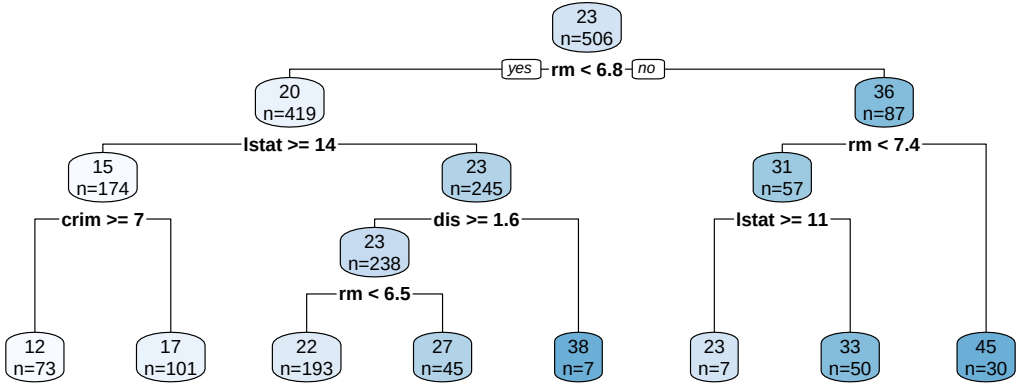
# 机器学习回归简介及案例: 例子和机器学习方法的优势

图中展示包括经典线性回归在内对波士顿住房数据回归的 16 种方法预测标准化均方误差 (NMSE) 的 10 折交叉验证结果.




最下面的线性回归误差最大, 上面的 5 种随机森林方法 (及延伸) 误差最小

# 决策树回归 对例1波士顿住房数据做决策树回归的结果



# 决策树回归 对例1波士顿住房数据做决策树回归的结果

```
n= 506

node), split, n, deviance, yval
    * denotes terminal node

1) root 506 42580.0 22.53
  2) rm< 6.838 419 16420.0 19.71
    4) lstat>=14.4 174  3112.0 14.83
      8) crim>=6.992 73   842.1 11.75 *
      9) crim< 6.992 101  1079.0 17.05 *
    5) lstat< 14.4 245  6205.0 23.18
      10) dis>=1.551 238  3195.0 22.75
        20) rm< 6.543 193  1499.0 21.69 *
        21) rm>=6.543 45   555.0 27.28 *
      11) dis< 1.551 7  1429.0 38.00 *
  3) rm>=6.838 87  6851.0 36.09
    6) rm< 7.437 57  2033.0 31.34
      12) lstat>=11.32 7   390.5 22.60 *
      13) lstat< 11.32 50  1033.0 32.57 *
    7) rm>=7.437 30  1099.0 45.10 *
```

## 决策树和线性回归的交叉验证比较

得到的交叉验证的输出表明, 两个回归的标准化均方误差相差不大, 其中决策树为 0.264, 而线性回归为 0.2739, 稍微差一点.

# 决策树的组合算法: bagging 回归

一棵决策树的效果可能很平常, 但是借助对数据多次放回抽样 (即自助法抽样 (bootstrap sampling)) 所得到的许多棵决策树却会得到非常好的模型, 这就是组合模型 (ensemble model), 组合模型中包括 bagging、boosting 和随机森林等非常优秀的模型.

bagging(bootstrap aggregating) 是由 Breiman 提出的一个简单的组合模型, 它对原始数据集做很多次放回抽样, 每次抽取和样本量同样多的观测值, 放回抽样使得每次都有大约 30%多的观测值没有被抽到, 另一些观测值则会被重复抽到. 如此, 得到很多不同的数据集, 然后针对每个数据集建立一个决策树, 因此产生大量决策树. 对于回归来说, 一个新的观测值通过如此多的决策树得到很多预测值, 最终结果为这些预测值的简单平均.

## bagging 和线性回归的交叉验证比较

对例1波士顿住房数据做和前面一样 (同样的 10 折数据) 的 bagging 回归的 10 折交叉验证, 得到标准化均方误差为 0.1789336, 只有最小二乘线性回归的 65.3%.

# 决策树的组合方法: 随机森林回归

随机森林是由 Breiman 提出的, 该方法把分类和回归的精度提到了前所未有的高度. 随机森林不会过拟合, 不怕维数诅咒, 变量多多益善, 对数据没有限制和假定. Breiman (2001) 用 "随机森林在基因芯片淋巴瘤数据集上运行, 有三个类别, 样本量为 81, 有 4682 个变量 (基因), 没有任何变量选择". 他指出, "从科学观点来看, 有趣的是估计了 4682 个基因表达的重要性". 对于如此少的样本量和如此多的变量, 经典统计是无能为力的. 图1是 Breiman 文章这一段的原文截图.

> Random forests was run on a microarray lymphoma data set with three classes, sample size of 81 and 4,682 variables (genes) without any variable selection [for more information about this data set, see Dudoit, Fridlyand and Speed, (2000)]. The error rate was low. What was also interesting from a scientific viewpoint was an estimate of the importance of each of the 4,682 gene expressions.

## 随机森林原理

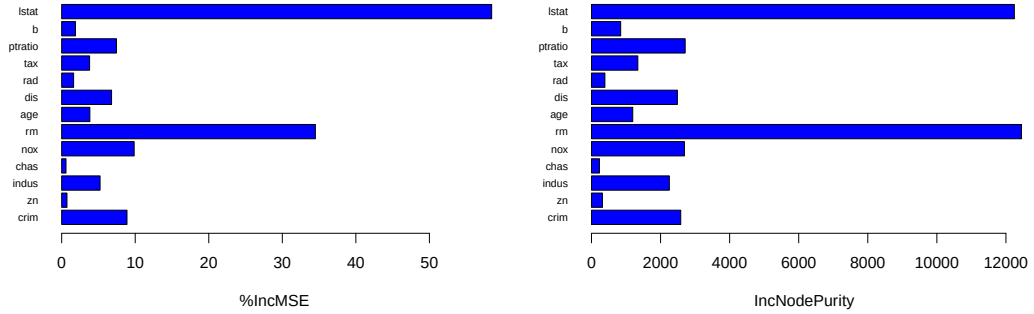
随机森林的原理并不复杂, 和 bagging 类似, 它对原始数据集做很多次放回抽样, 每次抽取和样本量同样多的观测值, 由于是放回抽样, 每次都有一些观测值没有被抽到, 有一些观测值会被重复抽到. 如此, 会得到很多不同的数据集, 然后针对每个数据集建立一棵决策树, 因此产生大量决策树. 和 bagging 不同的是, 在随机森林每棵树的每个节点, 拆分变量不是由所有变量竞争, 而是由随机挑选的少数变量竞争, 而且每棵树都长到底. 拆分变量候选者的数目限制可以避免由于强势变量主宰而忽略的数据关系中的细节, 因而大大提高了模型对数据的代表性. 随机森林的最终结果是所有树的结果的平均, 也就是说, 一个新的观测值通过许多棵树 (比如 $n$ 棵) 得到 $n$ 个预测值, 最终用这 $n$ 个预测值的平均作为最终结果.

## 用随机森林拟合例1波士顿住房数据

首先拟合所有数据:

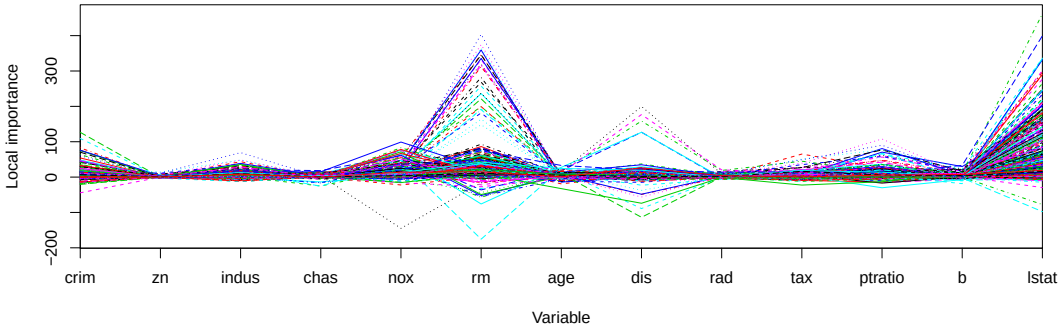
```
w=read.csv("BostonHousing2.csv")[,-c(1:5)]
a.rf=randomForest(cmedv~.,w,importance=TRUE,
   localImp=TRUE,proximity=TRUE)
```

左图是用 OOB 数据作为训练集所做的交叉验证得到的度量, 意义为删除某个自变量后均方误差增加的程度 (增加越多, 变量越重要), 右图为综合了每个自变量在各个节点的表现 (使得数据变 “纯”, 即使均方误差变小的效率) 而产生的重要性度量. 随机森林拟合例1波士顿住房数据的变量重要性图

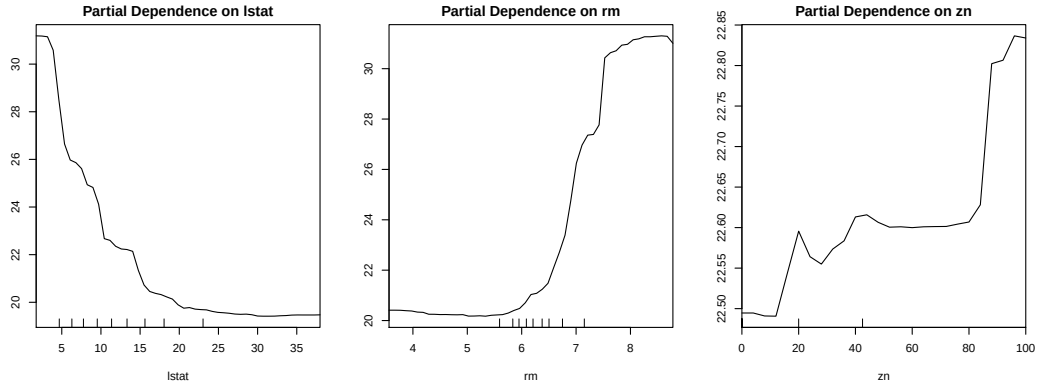


## 随机森林拟合例1波士顿住房数据的局部变量重要性图

这是每个自变量对每一个观测值的影响 ( `rf$local`), 对于例1的 13 个变量和 506 个观测值, 它是一个 $13 \times 506$ 矩阵, 图形 (见图41) 的每一条线代表一个观测值 (一共 506 个), 而横坐标为变量名, 纵坐标为局部重要性的度量.

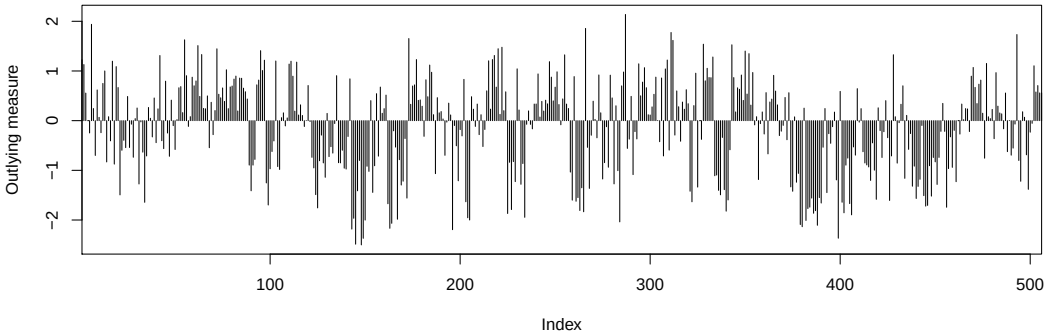


## 波士顿住房数据对三个变量的部分依赖性图 我们把重要的两个变量 lstat、rm 和不那么重要的 zn 点图




图左边及中间两个重要变量可以造成因变量幅度 10 (从 20 到 30 多) 的变化 (纵坐标是变化的一种度量), 而右边不重要的变量 zn 对因变量的影响幅度仅仅有 0.35 (从 22.50 到 22.85). 该图还说明因变量对第一个变量大体上是递减关系, 而对后两个大体上是递增关系.

随机森林拟合例1波士顿住房数据的离群点图 随机森林还根据是否经常出现在同样的节点给出了观测值之间的亲疏程度，根据这个度量 (`a.rf$proximity`) 可以得到离群点图



## 随机森林拟合例1波士顿住房数据预测精度的交叉验证

得到的标准化均方误差为 0.1202, 比前面决策树的 0.264, bagging 的 0.1789 及线性回归的 0.2739 都低, 这说明随机森林基于决策树, 但大大优于决策树. 关于随机森林的想法十几年来已经扩展了很多, 出现了很多衍生模型. 下面一节会在方法比较的计算中使用到其中的一些.

对例1波士顿住房数据的九种方法的误差比较 这里要比较的方法所用的函数包括 `lm`(线性回归)、`bagging` (bagging 回归)、`blackboost` (梯度自助回归)、`cforest` (条件随机森林)、`ctree` (条件推断树)、`ranger` (快速随机森林)、`kknn` ($k$ 最近邻法)、`randomForest` (随机森林)、`ksvm` (支持向量机).

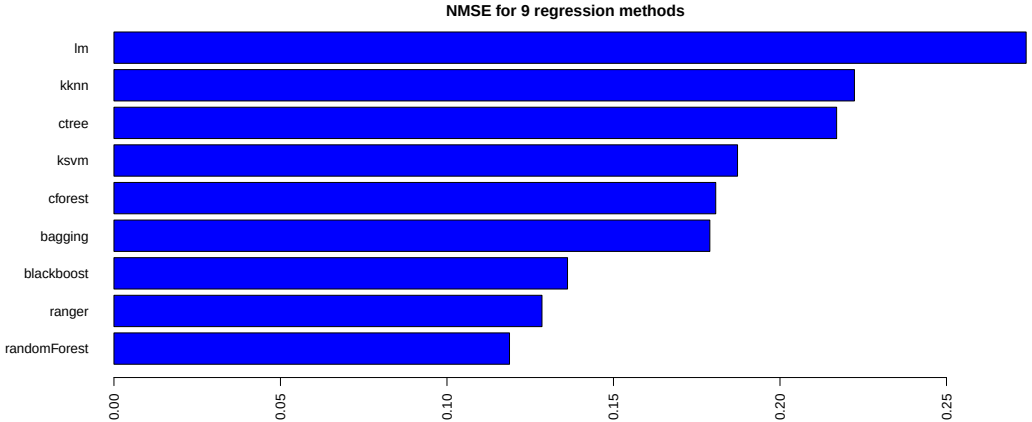



Figure: 九种方法拟合例1波士顿住房数据的 **NMSE** 图

# 经典线性回归与各种机器学习回归的比较 (总共 16 种方法)

```
> (RR=perf.grid[order(perf.grid[,2],decreasing = T),])
                                Predictor      NMSE   Rsquare        RMSE         MAE   time
1                       Linear Regression 0.2769006 0.7230994  108.58086    1703.192  0.708
7             eXtreme Gradient Boosting 0.2297022 0.7702978 1027.74592 149934.582 14.510
3                             Bagged MARS 0.2287623 0.7712377  170.94019    4448.486 14.787
6             Conditional Inference Tree 0.2204393 0.7795607  167.80172    4465.628  1.419
2                             Bagged CART 0.1932147 0.8067853   90.70086    1372.664  1.741
16                   k-Nearest Neighbors 0.1861523 0.8138477  154.20062    3675.379  1.065
14 Bayesian Regularized Neural Networks 0.1849227 0.8150773  153.69049    4112.808  2.306
5                             Boosted Tree 0.1821114 0.8178886  264.16854   12286.469 10.231
4          Bagged MARS using gCV Pruning 0.1480328 0.8519672   79.39081    1218.099  5.911
9           Stochastic Gradient Boosting 0.1446443 0.8553557  235.43079   10821.341  1.951
15          Stochastic Gradient Boosting 0.1446443 0.8553557  235.43079   10821.341  1.900
8                       Random Forest 0.1297286 0.8702714  182.04751    6667.209  7.433
11               Random Forest (ranger) 0.1297286 0.8702714  182.04751    6667.209  7.252
13                    Random Forest(rf) 0.1207560 0.8792440  124.19564    3261.894 20.848
10               Parallel Random Forest 0.1207127 0.8792873  124.17336    3254.063 21.745
12               Random Forest(Rborist) 0.1111774 0.8888226  119.16812    3265.148 15.691
```

以上结果已经在本节开始的图31中展示.