Regression_tree

Nivedita

2024-04-09

Objective

To perform fitting of regression tree, bagging, boosting and random forest for Boston dataset in MASS package.

Analysis

```
# Show number of rows and columns of the Boston dataset
dim(Boston)
```

```
## [1] 506  14
```

In the context of R, the Boston dataset is found in the MASS library and has 506 rows and 14 columns.

```
# Show first 6 rows of data
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

```
## Print out the column names of Boston dataset using names function
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

Fitting of regression tree

We want to fit regression tree for this dataset to predict median value of a home, medv. First, we create a training set, and fit the tree to the training data.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.3
```

```
set.seed(1)
train <-sample(1:nrow(Boston), nrow(Boston) / 2)
tree.boston <-tree(medv~., Boston, subset = train)
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

From summary we can see that only four of the variables have been used in constructing the tree that are “rm”, “lstat”, “crim” , “age” . For visual representation of fitted regression tree we use plot command.

```
plot(tree.boston)
text(tree.boston, pretty = 0)
```

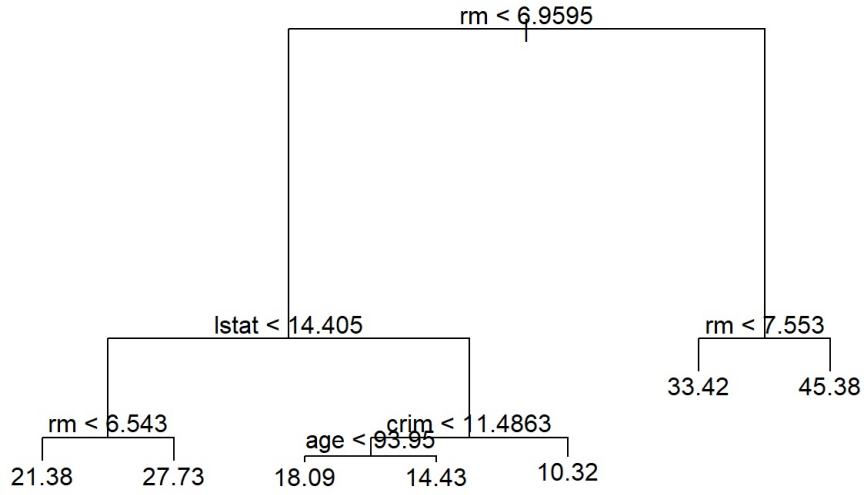


Fig. 1 Plot of regression tree

The variable lstat measures the percentage of individuals with lower socioeconomic status, while the variable rm corresponds to the average num ber of rooms. The tree indicates that larger values of rm, or lower values of lstat, correspond to more expensive houses. For example, the tree predicts a median house price of $45,400 for homes in census tracts in which rm >=7.553.

Now we use the cv.tree() function to see whether pruning the tree will improve performance.

```
cv.boston <- cv.tree(tree.boston)
plot(cv.boston$size, cv.boston$dev, type = "b",main='Plot of error rate function of size ')
```

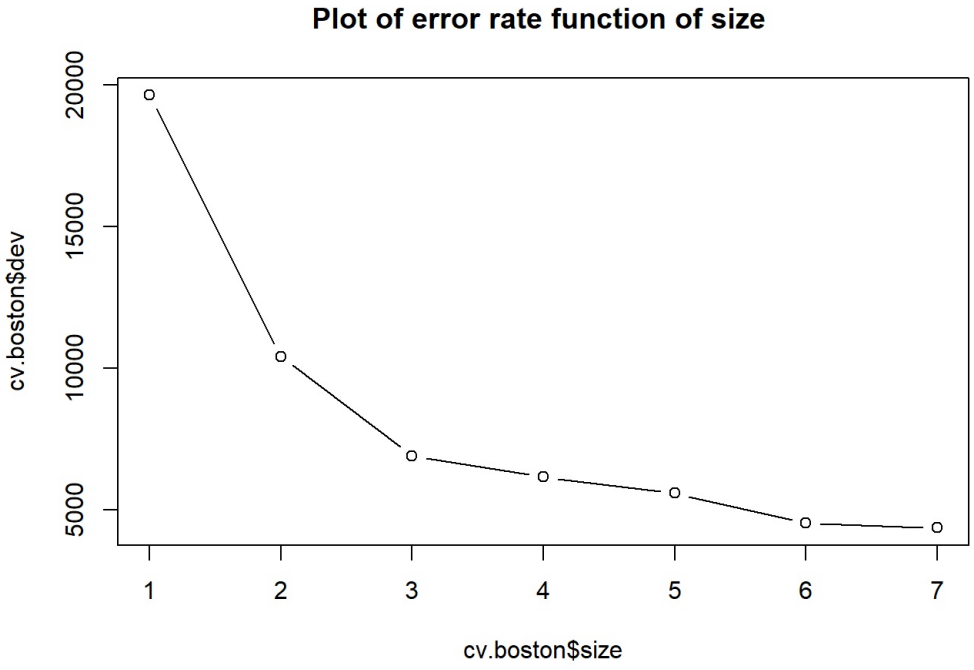


Fig. 2 Plot of error rate function of size

Form Fig. 2 we can see that we can see that 7 terminal nodes results minnimum cross validation errors and we have 7 terminal nodes in the fitted regression tree. So in this case, the most complex tree under consideration is selected by cross validation that means there is no need to perform prining process.

In keeping with the cross-validation results, we use the unpruned tree to make predictions on the test set.

```
yhat <- predict(tree.boston, newdata = Boston[-train, ])
boston.test <- Boston[-train, "medv"]
plot(yhat, boston.test,xlab='Predicted value of medv',ylab = 'Vales of medv in test data', main='Plot of observed
and predicted value of medv for test data' )
abline(0, 1)
```

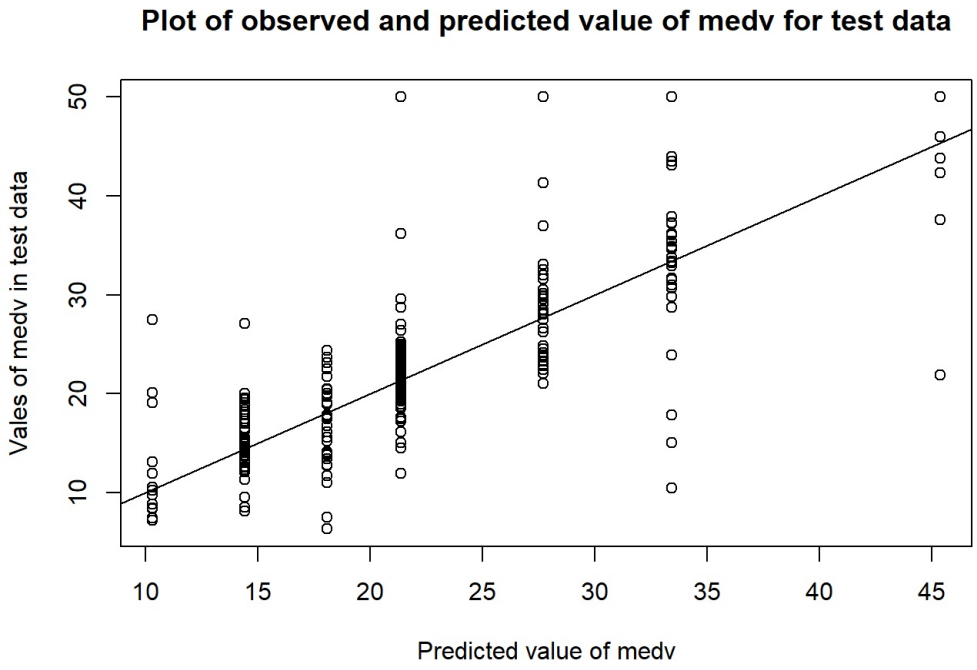


Fig. 3 Plot of observed and predicted value of medv for test data

```
mean((yhat- boston.test)^2)
```

```
## [1] 35.28688
```

We can see that the test set MSE associated with the regression tree is 35.29. The square root of the MSE is therefore around 5.941 indicates that this model leads to test predictions that are (on average) within approximately $5,941 of the true median home value for the census tract.

Bagging

Here we apply bagging to the Boston data, using the randomForest package in R. The argument mtry = 12 indicates that all 12 predictors should be considered for each split of the tree—in other words, that bagging should be done.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.boston <- randomForest(medv ~ ., data = Boston,subset = train, mtry = 12, importance = TRUE)
bag.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry = 12, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 11.10176
##                     % Var explained: 85.56
```

Now to check how well does this bagged model perform on the test set.

```
yhat.bag <-predict(bag.boston, newdata = Boston[-train, ])
plot(yhat.bag, boston.test, main='Plot of observed and predicted value from bagged model of medv for test data',c
ol='skyblue')
abline(0, 1)
```

Fig. 4 Plot to check the performance of bagged model

```
mean((yhat.bag- boston.test)^2)
```

```
## [1] 23.38773
```

The test set MSE associated with the bagged regression tree is 23.42, about two-thirds of that obtained using an optimally-pruned single tree.

Random Forest

Here we apply random forest to the Boston data, using same randomForest package in R. Now we use $\sqrt{p}$ variables when building a random forest of classification trees. Here we use mtry=6.

```
set.seed(1)
rf.boston <-randomForest(medv~., data = Boston, subset = train, mtry = 6, importance = TRUE)
yhat.rf <-predict(rf.boston, newdata = Boston[-train, ])
mean((yhat.rf- boston.test)^2)
```

```
## [1] 19.62021
```

The test set MSE is 20.07; this indicates that random forests yielded an improvement over bagging in this case.

We can use the importance() function, to view the importance of each variable. From this function two measures of variable importance are reported. The first is based upon the mean decrease of accuracy in predictions on the out of bag samples when a given variable is permuted. The second is a measure of the total decrease in node impurity that results from splits over that variable, averaged over all trees.

```
# to check importance
importance(rf.boston)
```

```
##           %IncMSE IncNodePurity
## crim    16.697017    1076.08786
## zn       3.625784      88.35342
## indus    4.968621     609.53356
## chas     1.061432      52.21793
## nox     13.518179     709.87339
## rm      32.343305    7857.65451
## age     13.272498     612.21424
## dis      9.032477     714.94674
## rad      2.878434      95.80598
## tax      9.118801     364.92479
## ptratio  8.467062     823.93341
## black    7.579482     275.62272
## lstat   27.129817    6027.63740
```

```
#plot
varImpPlot(rf.boston,main = 'Plots of importance measures')
```

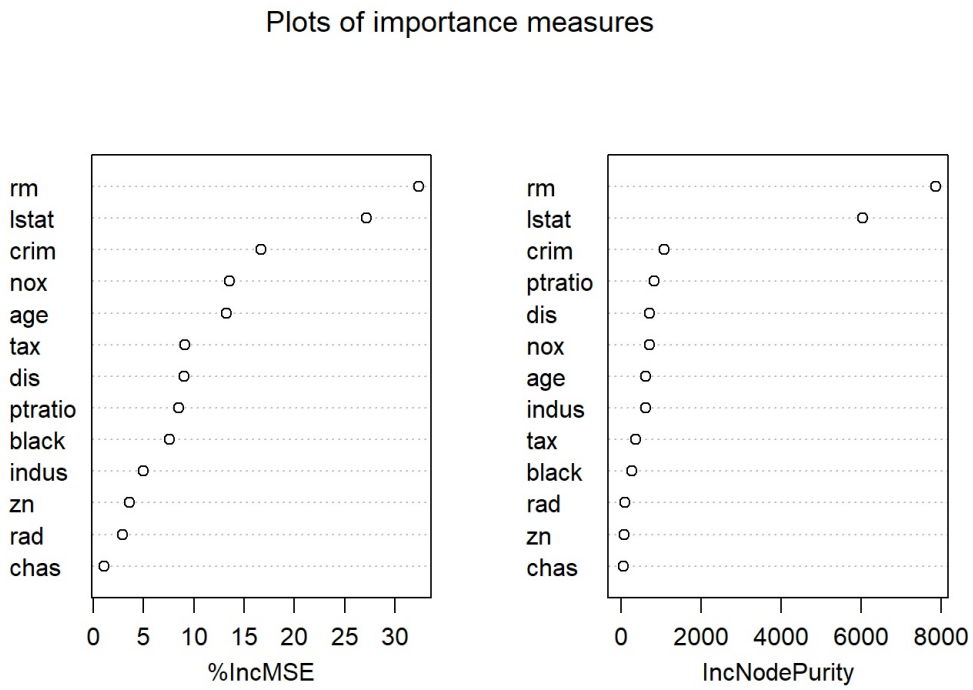


Fig. 5 Plots of importance measures

From Fig. 5 we can see that across all of the trees considered in the random forest, the wealth of the community (lstat) and the house size (rm) are by far the two most important variables.

Boosting

Here we use the gbm package, and within it the gbm() function, to fit boosted gbm() regression trees to the Boston data set.

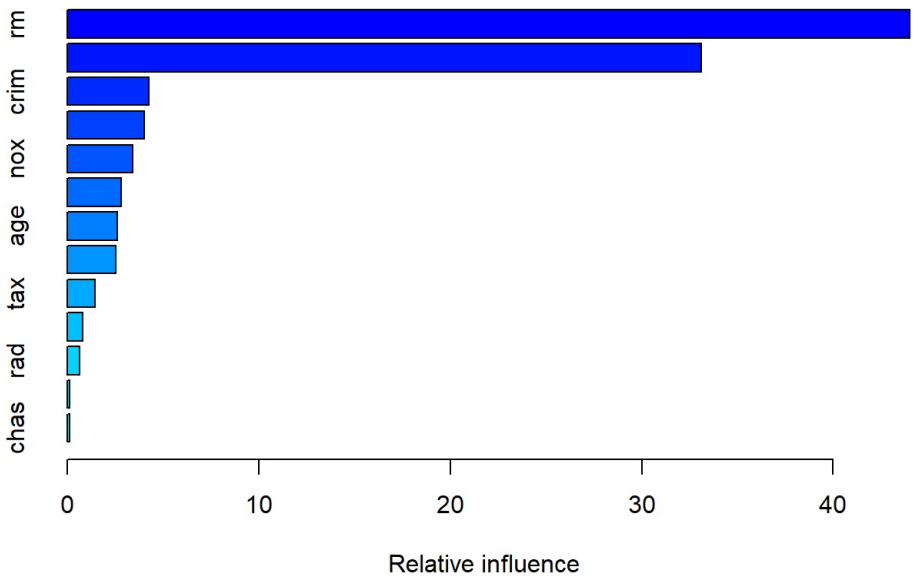
```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.3
```

```
## Loaded gbm 2.1.9
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-dev
elopers/gbm3
```

```
set.seed(1)
boost.boston <- gbm(medv~., data = Boston[train, ], distribution = "gaussian", n.trees = 5000, interaction.depth
= 4)
summary(boost.boston)
```




```
##             var    rel.inf
## rm           rm 43.9919329
## lstat     lstat 33.1216941
## crim       crim  4.2604167
## dis         dis  4.0111090
## nox         nox  3.4353017
## black     black  2.8267554
## age         age  2.6113938
## ptratio ptratio  2.5403035
## tax         tax  1.4565654
## indus     indus  0.8008740
## rad         rad  0.6546400
## zn           zn  0.1446149
## chas       chas  0.1443986
```

From summary we can see that lstat and rm are by far the most important variables. We can also produce partial dependence plots for these two variables

```
plot(boost.boston, i = "rm", main='Partial dependence plot of rm')
```

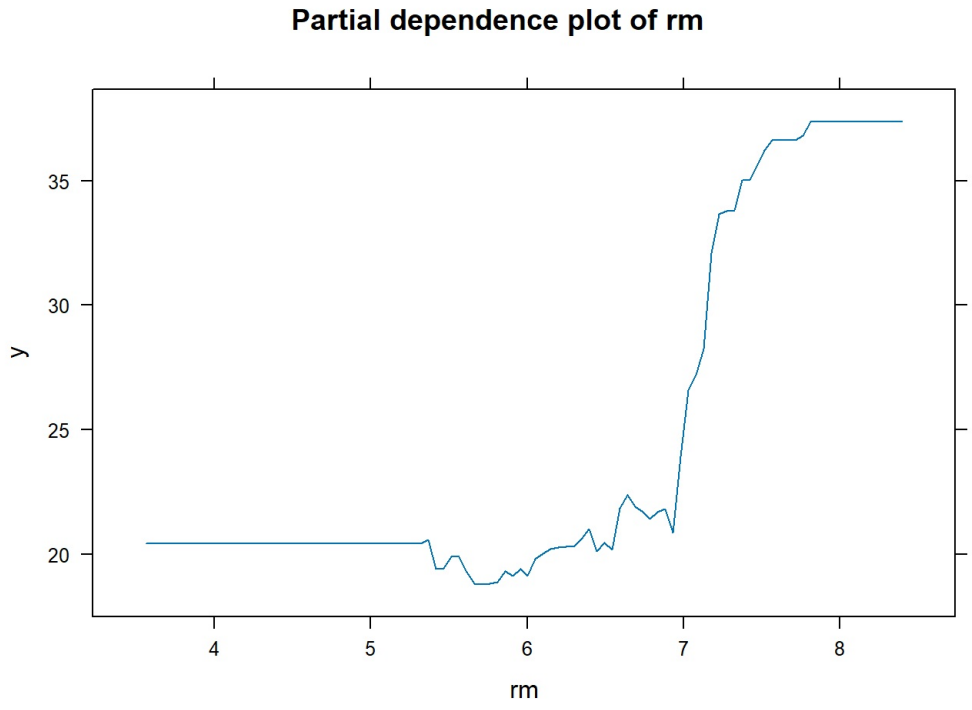


Fig. 6 Partial dependence plot of rm

```
plot(boost.boston, i = "lstat", main='Partial dependence plot of lstat')
```

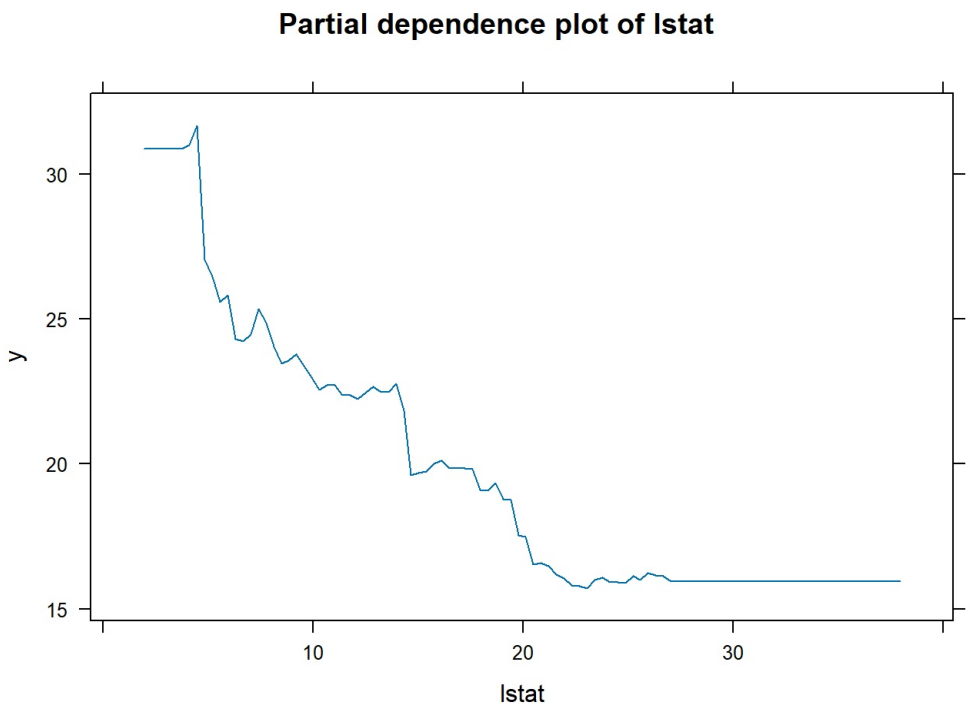


Fig. 7 Partial dependence plot of lstat

We now use the boosted model to predict medv on the test set:

```
yhat.boost <- predict(boost.boston, newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost- boston.test)^2)
```

```
## [1] 18.84709
```

The test MSE obtained is 18.39: this is superior to the test MSE of random forests and bagging.

Conclusion

We performed fitting of regression tree, bagging, boosting and random forest for Boston data set in MASS package. We observed that test MSE obtained in boosting is minimum among all and lstat and rm are the most important variables. We conclude that larger values of average number of rooms per dwelling (rm), or lower values of the percentage of individuals with lower socioeconomic status (lstat), correspond to more expensive houses.