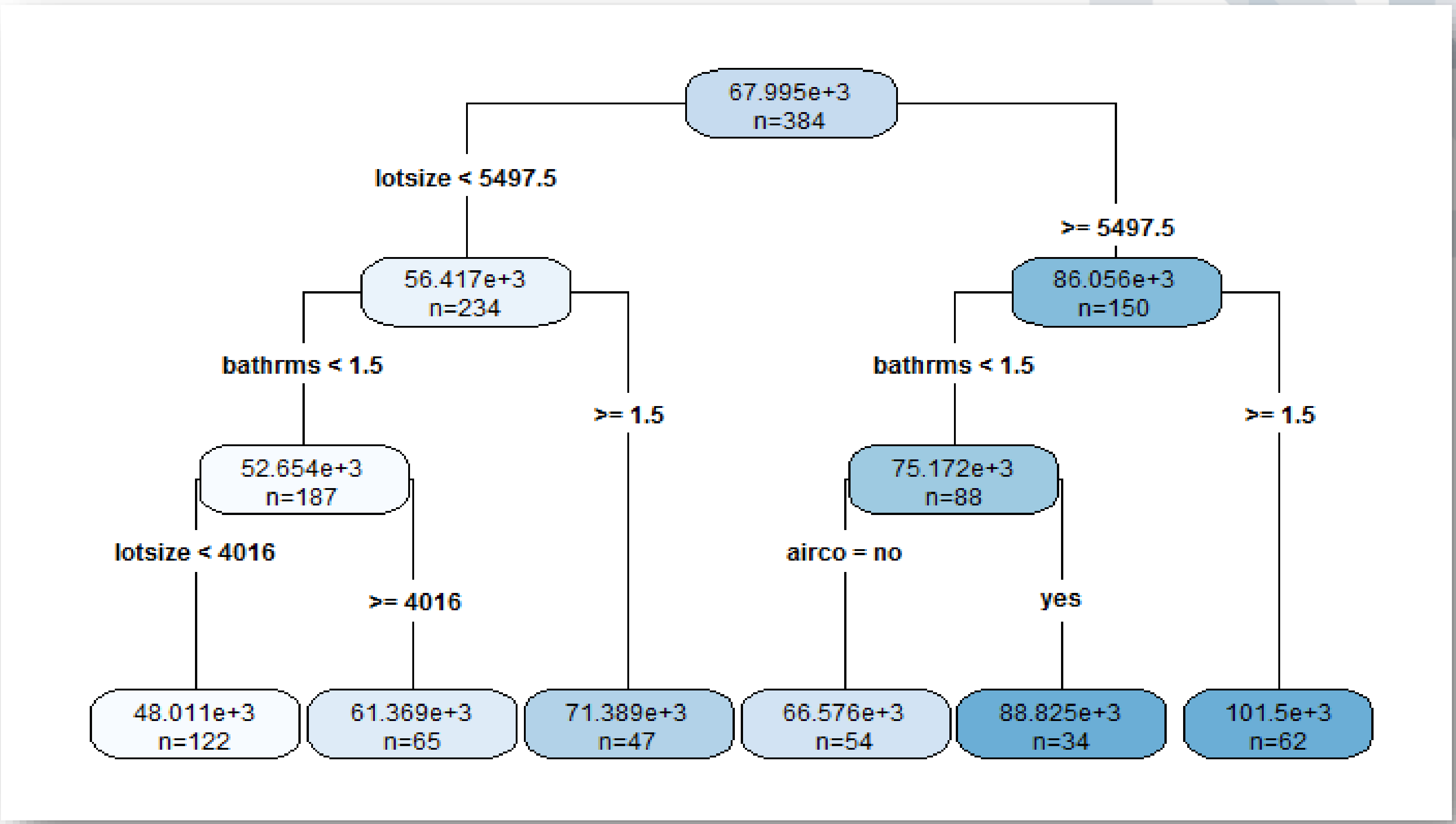
Regression Tree

Comparison of Types of Trees

	Classification	Regression
Response Variable Type	Categorical	Numerical
Measuring Homogeneity	Gini, Entropy	Deviance
Prediction	Majority Class in the leaf node	Mean of response variable in the data in leaf node
Evaluation	Confusion Matrix metrics, ROC(only for 2 categories)	MSE, MAE, R^2 RMSE, MAPE, RMSPE

Typical Regression Tree Output

- In case of regression trees, the difference is that on the leaf nodes we have the means of the response variable values.



Regression Tree

- The data gets divided into two parts in the interest of decreasing the variation of response variable
- The child nodes have lesser variation than their respective parent nodes for response variable

Regression Tree in R

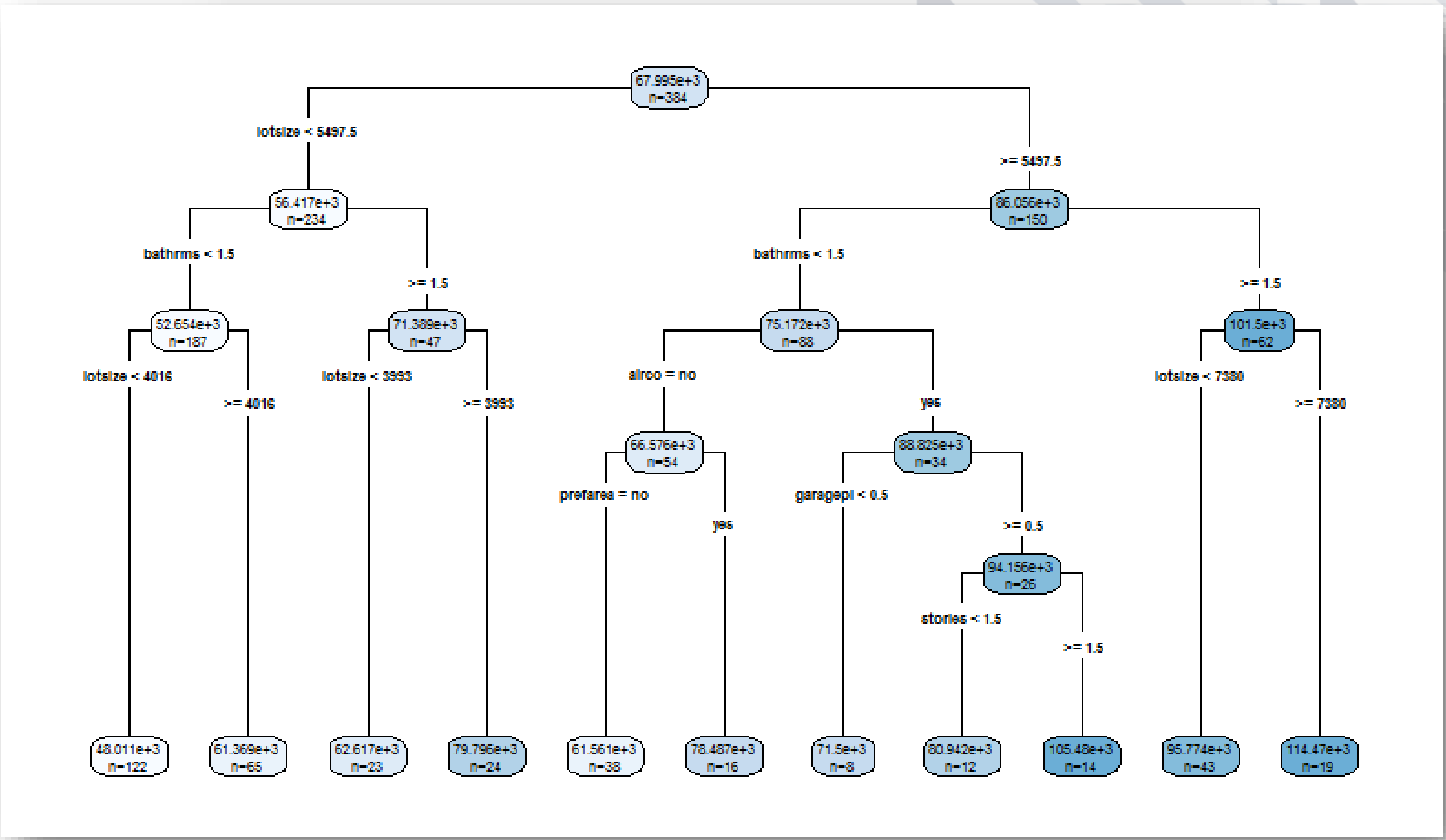
- For implementing Regression Tree in rpart( ), option method should be specified with "anova".
- For implementing Regression Tree in ctree( ), there is no option to be specified differently. It identifies the response variable type and executes accordingly

Example : Sales Prices of Houses in the City of Windsor

- **Description**
  - a cross-section from 1987
  - *number of observations* : 546
  - *country* : Canada
- A dataframe containing :
  - **price :** sale price of a house
  - **lotsize :** the lot size of a property in square feet
  - **bedrooms :** number of bedrooms
  - **bathrms :** number of full bathrooms
  - **stories :** number of stories excluding basement
  - **driveway :** does the house has a driveway ?
  - **recroom :** does the house has a recreational room ?
  - **fullbase :** does the house has a full finished basement ?
  - **gashw :** does the house uses gas for hot water heating ?
  - **airco :** does the house has central air conditioning ?
  - **garagepl :** number of garage places
  - **prefarea :** is the house located in the preferred neighbourhood of the city ?

Program and Output – Using rpart

```
rpart.plot(fitRT,type = 4,extra = 1, digits = 5)
```



Model Evaluation: RMSE

- RMSE : Root Mean Square Error

$$RMSE = \sqrt{\frac{\sum(y_i - \widehat{y_i}\ )^2}{n}}$$

where

y_i = Observed Values
$\widehat{y_i}$ = Predicted Values
n = No. of observations

Model Evaluation: MAPE

- MAPE: Mean Absolute Percentage Error

$$\text{MAPE} = \frac{\sum \left| \frac{y_i - \widehat{y_i}}{y_i} \right|}{n}$$

- Where

y_i = Observed Values

$\widehat{y_i}$ = Predicted Values

n = No. of observations

Model Evaluation: RMSPE

- RMSPE: Root Mean Square Percentage Error
  - Often used at Kaggle competitions

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$$

- Where

y_i = Observed Values

$\hat{y}_i$ = Predicted Values

n = No. of observations

Model Evaluation in R

- About MAPE, RMSE and RMPSE, only one criterion holds: **Smaller** their value, **Better** is the model prediction
- RMSE is calculated as directly by function postResample( )
- For MAPE and RMSPE we can create a functions

```
MAPE <- function(y, yhat) {
  mean(abs((y - yhat)/y))
}
```

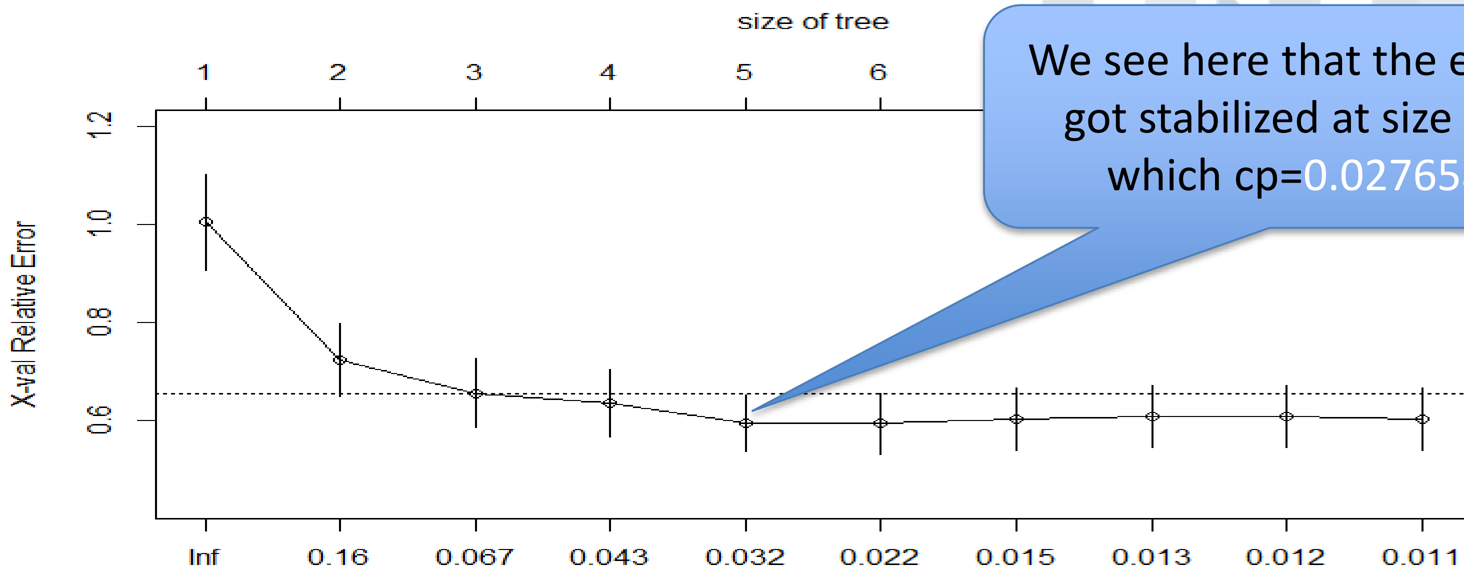
```
RMPSE<- function(y, yhat) {
  sqrt(mean((y-yhat)/y)^2)
}
```

Output

```
> pred.RT <- predict(fitRT,newdata = validation )
>
> postResample(pred.RT , validation$price)
        RMSE     Rsquared
1.809074e+04 5.414778e-01
>
> MAPE <- function(y, yhat) {
+   mean(abs((y - yhat)/y))
+ }
>
> MAPE(validation$price , pred.RT)
[1] 0.2036191
>
> RMSPE<- function(y, yhat) {
+   sqrt(mean((y-yhat)/y)^2)
+ }
>
> RMSPE(validation$price , pred.RT)
[1] 0.04037694
>
```

Pruning

- We take a look at plotcp( ) output

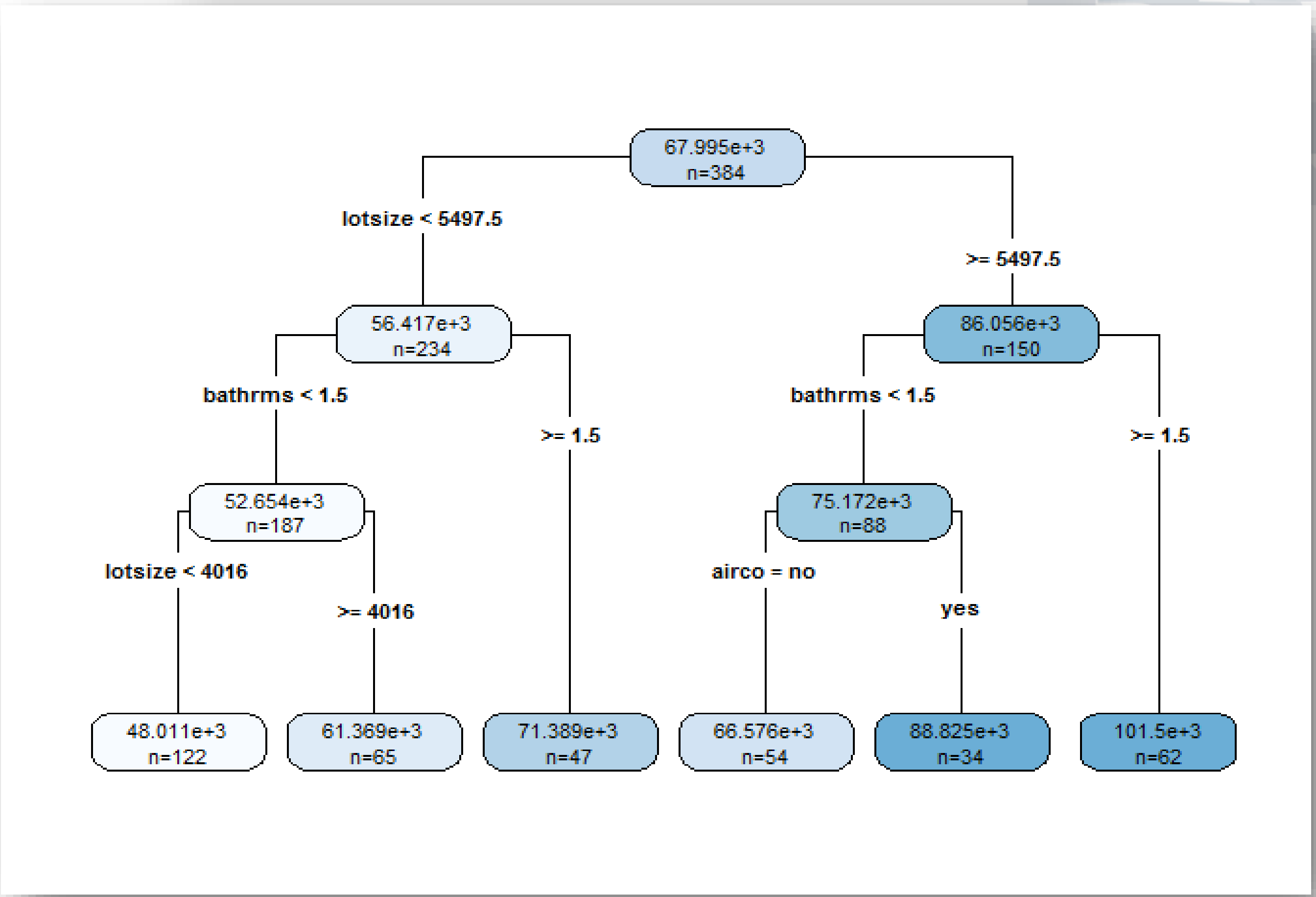


We see here that the error has got stabilized at size = 5 for which cp=0.02765860.

```
> fitRT$cptable
           CP nsplit rel error    xerror       xstd
1  0.29348617      0 1.0000000 1.0051535 0.09724243
2  0.09217662      1 0.7065138 0.7250328 0.07373928
3  0.04818811      2 0.6143372 0.6572006 0.06866341
4  0.03775017      3 0.5661491 0.6367268 0.06719953
5  0.02765860      4 0.5283989 0.5957366 0.05874151
6  0.01683250      5 0.5007403 0.5945977 0.06134276
7  0.01285050      6 0.4839078 0.6042144 0.06289123
8  0.01266817      8 0.4582068 0.6093850 0.06347051
9  0.01179162      9 0.4455387 0.6086444 0.06344482
10 0.01000000     10 0.4337470 0.6039693 0.06348830
```

Pruning

```
fitRT.pruned <- prune(fitRT , cp=0.02765860  )
```



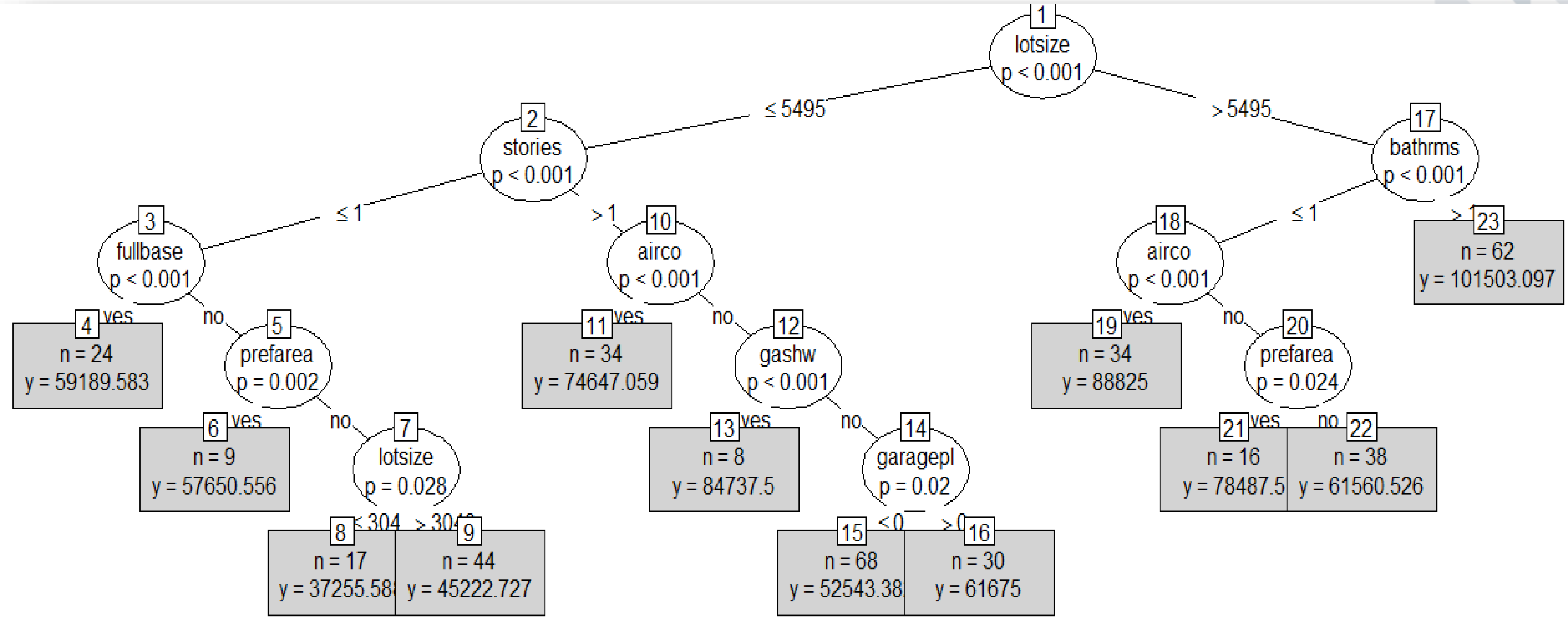
Output

```
> pred.RT.pruned <- predict(fitRT.pruned , newdata = validation)
> postResample(pred.RT.pruned , validation$price)
        RMSE     Rsquared 
1.895032e+04 4.967799e-01 
> MAPE(validation$price , pred.RT.pruned)
[1] 0.2180033
> RMSPE(validation$price , pred.RT.pruned)
[1] 0.04583555
```

Though we might observe that the accuracy has been affected negatively by pruning, still we have minimized the risk of overfitting of the model

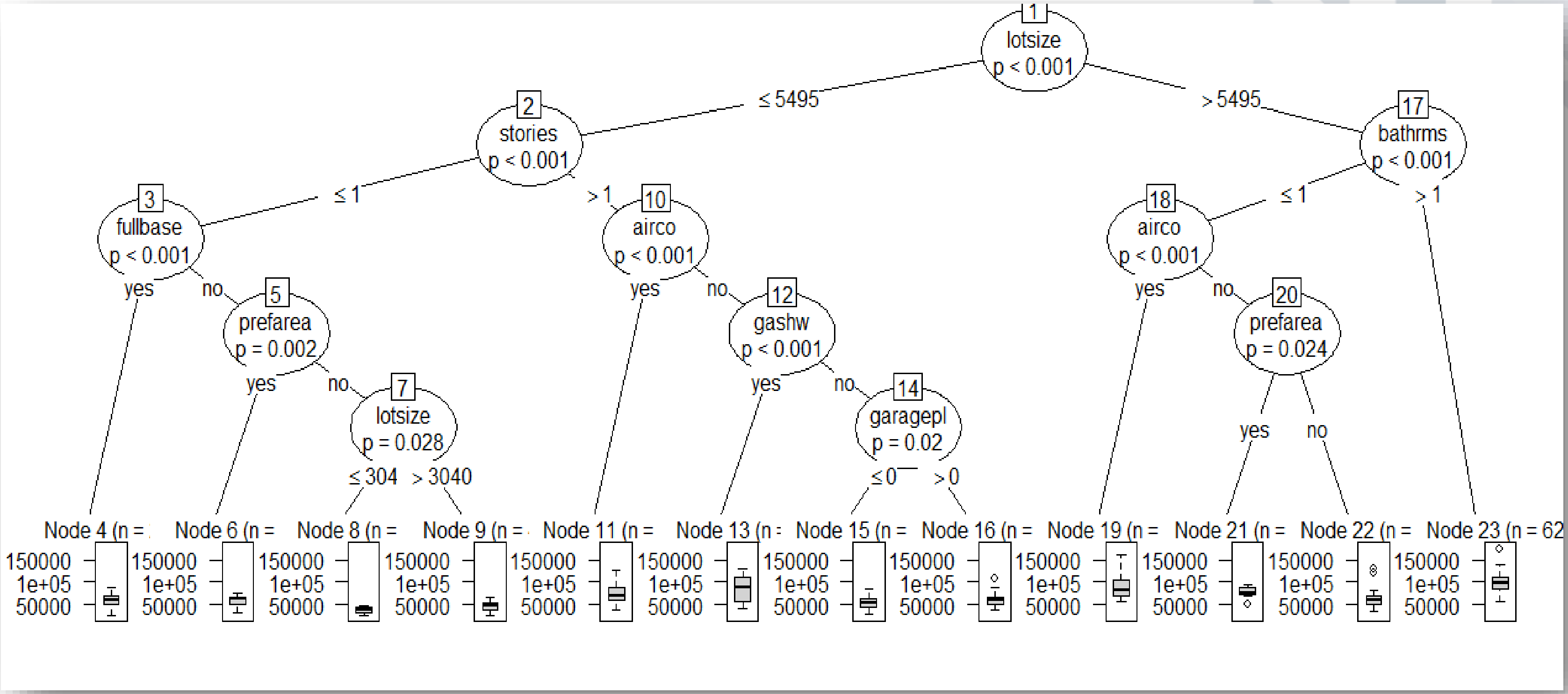
Program and Output – Using party

```
library(party)

fitCT <- ctree(price ~ . , data = training )

plot(fitCT , type="simple")
```



Program and Output – Using party

```
plot(fitCT , type="extended" )
```



Accuracy Measures : party

```
> pred.CT <- predict(fitCT , newdata=validation)
>
> postResample(pred.CT , validation$price)
        RMSE     Rsquared
1.831758e+04 5.279237e-01
> MAPE(validation$price , pred.CT)
[1] 0.2145092
> RMSPE(validation$price , pred.CT)
[1] 0.06729213
```