Predicting Home Prices

Siddhartha Jetti

December 27, 2016

1. Problem Defnition

Predict House prices in Suburbs of Boston. The Dataset Description can be found at : https://archive.ics.uci.edu/ml/datasets/Housing (https://archive.ics.uci.edu/ml/datasets/Housing)

a. Load libraries

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 3.3.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.3.3
```

b. Load dataset

```
data(BostonHousing)
```

c. Split-out validation dataset create a list of 80% of the rows in the original dataset for training the model. select 20% of the data for validation use the remaining 80% of data to training and testing the models

```
set.seed(7)
validation_index <- createDataPartition(BostonHousing$medv, p=0.80, list=FALSE)
validation <- BostonHousing[-validation_index,]
dataset <- BostonHousing[validation_index,]
```

2. Summarize Data

Dimensions of dataset

```
dim(dataset)
```

```
## [1] 407  14
```

List types for each attribute

```
sapply(dataset, class)
```

```
##      crim        zn     indus      chas       nox        rm       age 
## "numeric" "numeric" "numeric"  "factor" "numeric" "numeric" "numeric" 
##       dis       rad       tax   ptratio         b     lstat      medv 
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
```

take a peek at the first 5 rows of the data

```
head(dataset, n=20)
```

```
##       crim   zn indus chas   nox    rm   age    dis rad tax ptratio      b
## 2  0.02731  0.0  7.07    0 0.469 6.421  78.9 4.9671   2 242    17.8 396.90
## 3  0.02729  0.0  7.07    0 0.469 7.185  61.1 4.9671   2 242    17.8 392.83
## 4  0.03237  0.0  2.18    0 0.458 6.998  45.8 6.0622   3 222    18.7 394.63
## 5  0.06905  0.0  2.18    0 0.458 7.147  54.2 6.0622   3 222    18.7 396.90
## 6  0.02985  0.0  2.18    0 0.458 6.430  58.7 6.0622   3 222    18.7 394.12
## 7  0.08829 12.5  7.87    0 0.524 6.012  66.6 5.5605   5 311    15.2 395.60
## 8  0.14455 12.5  7.87    0 0.524 6.172  96.1 5.9505   5 311    15.2 396.90
## 9  0.21124 12.5  7.87    0 0.524 5.631 100.0 6.0821   5 311    15.2 386.63
## 13 0.09378 12.5  7.87    0 0.524 5.889  39.0 5.4509   5 311    15.2 390.50
## 14 0.62976  0.0  8.14    0 0.538 5.949  61.8 4.7075   4 307    21.0 396.90
## 15 0.63796  0.0  8.14    0 0.538 6.096  84.5 4.4619   4 307    21.0 380.02
## 16 0.62739  0.0  8.14    0 0.538 5.834  56.5 4.4986   4 307    21.0 395.62
## 17 1.05393  0.0  8.14    0 0.538 5.935  29.3 4.4986   4 307    21.0 386.85
## 18 0.78420  0.0  8.14    0 0.538 5.990  81.7 4.2579   4 307    21.0 386.75
## 19 0.80271  0.0  8.14    0 0.538 5.456  36.6 3.7965   4 307    21.0 288.99
## 20 0.72580  0.0  8.14    0 0.538 5.727  69.5 3.7965   4 307    21.0 390.95
## 23 1.23247  0.0  8.14    0 0.538 6.142  91.7 3.9769   4 307    21.0 396.90
## 25 0.75026  0.0  8.14    0 0.538 5.924  94.1 4.3996   4 307    21.0 394.33
## 26 0.84054  0.0  8.14    0 0.538 5.599  85.7 4.4546   4 307    21.0 303.42
## 27 0.67191  0.0  8.14    0 0.538 5.813  90.3 4.6820   4 307    21.0 376.88
##    lstat medv
## 2   9.14 21.6
## 3   4.03 34.7
## 4   2.94 33.4
## 5   5.33 36.2
## 6   5.21 28.7
## 7  12.43 22.9
## 8  19.15 27.1
## 9  29.93 16.5
## 13 15.71 21.7
## 14  8.26 20.4
## 15 10.26 18.2
## 16  8.47 19.9
## 17  6.58 23.1
## 18 14.67 17.5
## 19 11.69 20.2
## 20 11.28 18.2
## 23 18.72 15.2
## 25 16.30 15.6
```

```
## 26 16.51 13.9
## 27 14.81 16.6
```

summarize attribute distributions

```
summary(dataset)
```

```
##       crim                zn             indus       chas   
##  Min.   : 0.00906   Min.   : 0.00   Min.   : 0.46   0:376  
##  1st Qu.: 0.08556   1st Qu.: 0.00   1st Qu.: 5.19   1: 31  
##  Median : 0.28955   Median : 0.00   Median : 9.90          
##  Mean   : 3.58281   Mean   :10.57   Mean   :11.36          
##  3rd Qu.: 3.50464   3rd Qu.: 0.00   3rd Qu.:18.10          
##  Max.   :88.97620   Max.   :95.00   Max.   :27.74          
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.863   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4530   1st Qu.:5.873   1st Qu.: 45.05   1st Qu.: 2.031  
##  Median :0.5380   Median :6.185   Median : 77.70   Median : 3.216  
##  Mean   :0.5577   Mean   :6.279   Mean   : 68.83   Mean   : 3.731  
##  3rd Qu.:0.6310   3rd Qu.:6.611   3rd Qu.: 94.55   3rd Qu.: 5.100  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :10.710  
##       rad              tax           ptratio            b         
##  Min.   : 1.000   Min.   :188.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:374.50  
##  Median : 5.000   Median :330.0   Median :19.00   Median :391.13  
##  Mean   : 9.464   Mean   :405.6   Mean   :18.49   Mean   :357.88  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.27  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat             medv      
##  Min.   : 1.730   Min.   : 5.00  
##  1st Qu.: 6.895   1st Qu.:17.05  
##  Median :11.500   Median :21.20  
##  Mean   :12.827   Mean   :22.61  
##  3rd Qu.:17.175   3rd Qu.:25.00  
##  Max.   :37.970   Max.   :50.00  
```

convert factor to numeric

```
dataset[,4] <- as.numeric(as.character(dataset[,4]))
```

a. Descriptive statistics

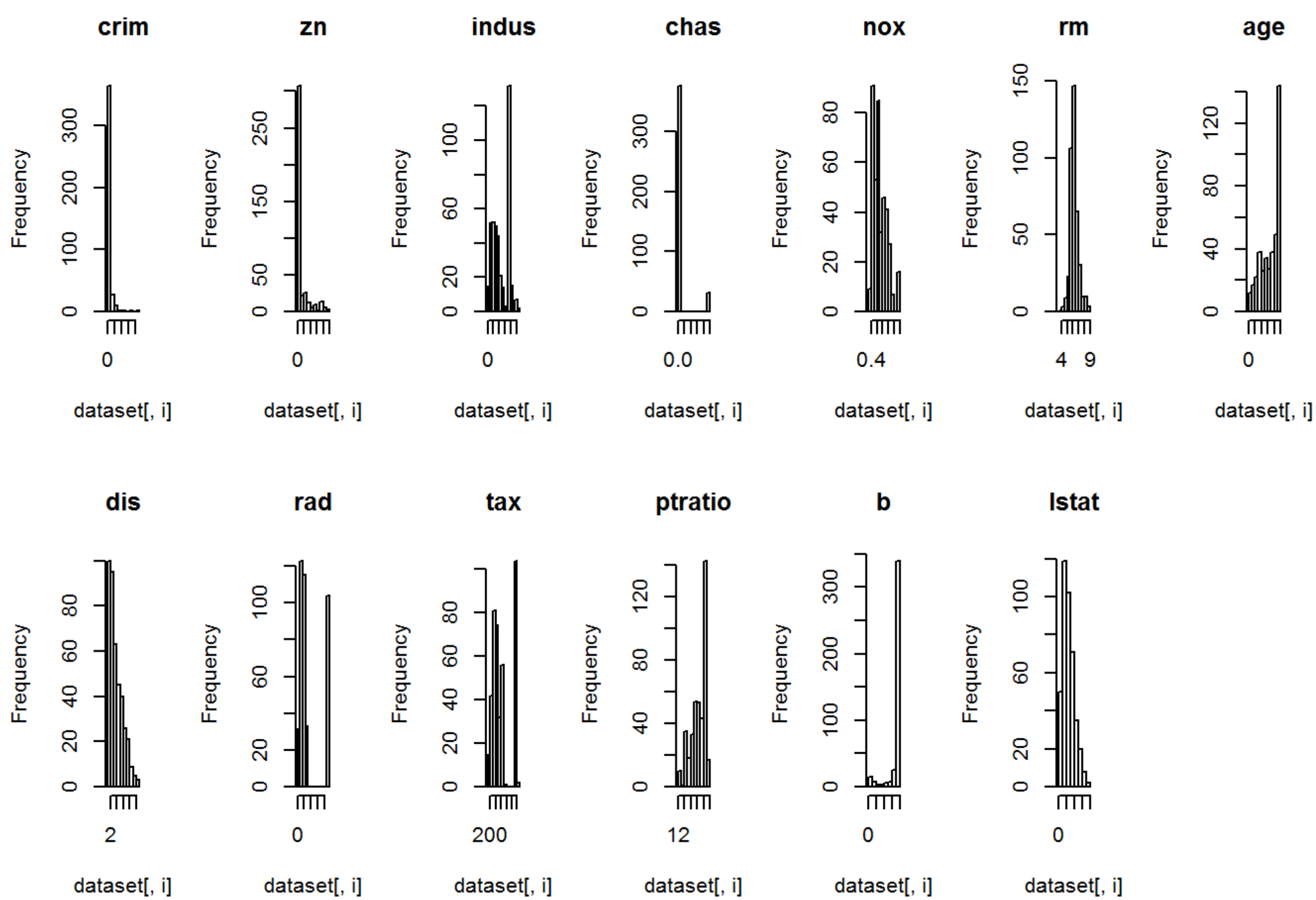
```
cor(dataset[,1:13])
```

```
##                crim          zn       indus        chas        nox
## crim     1.00000000 -0.19790631  0.40597009 -0.05713065  0.4232413
## zn      -0.19790631  1.00000000 -0.51895069 -0.04843477 -0.5058512
## indus    0.40597009 -0.51895069  1.00000000  0.08003629  0.7665481
## chas    -0.05713065 -0.04843477  0.08003629  1.00000000  0.1027366
## nox      0.42324132 -0.50585121  0.76654811  0.10273656  1.0000000
## rm      -0.21513269  0.28942883 -0.37673408  0.08252441 -0.2988506
## age      0.35438190 -0.57070265  0.65858310  0.10938121  0.7238371
## dis     -0.39050970  0.65618742 -0.72305885 -0.11142420 -0.7708680
## rad      0.64240501 -0.29952976  0.56774365 -0.00901245  0.5851676
## tax      0.60622608 -0.28791668  0.68070916 -0.02779018  0.6521787
## ptratio  0.28929828 -0.35341215  0.32920610 -0.13554380  0.1416616
## b       -0.30211854  0.16927489 -0.33597951  0.04724420 -0.3620791
## lstat    0.47537617 -0.39712686  0.59212718 -0.04569239  0.5819645
##                  rm        age        dis         rad         tax
## crim    -0.21513269  0.3543819 -0.3905097  0.64240501  0.60622608
## zn       0.28942883 -0.5707027  0.6561874 -0.29952976 -0.28791668
## indus   -0.37673408  0.6585831 -0.7230588  0.56774365  0.68070916
## chas     0.08252441  0.1093812 -0.1114242 -0.00901245 -0.02779018
## nox     -0.29885055  0.7238371 -0.7708680  0.58516760  0.65217875
## rm       1.00000000 -0.2325359  0.1952159 -0.19149122 -0.26794733
## age     -0.23253586  1.0000000 -0.7503321  0.45235421  0.50164657
## dis      0.19521590 -0.7503321  1.0000000 -0.49382744 -0.52649325
## rad     -0.19149122  0.4523542 -0.4938274  1.00000000  0.92137876
## tax     -0.26794733  0.5016466 -0.5264932  0.92137876  1.00000000
## ptratio -0.32000372  0.2564318 -0.2021897  0.45312318  0.44192428
## b        0.15539923 -0.2512574  0.2826819 -0.41033069 -0.41848779
## lstat   -0.62038075  0.5932128 -0.4957302  0.47306604  0.52339243
##             ptratio          b       lstat
## crim     0.2892983 -0.3021185  0.47537617
## zn      -0.3534121  0.1692749 -0.39712686
## indus    0.3292061 -0.3359795  0.59212718
## chas    -0.1355438  0.0472442 -0.04569239
## nox      0.1416616 -0.3620791  0.58196447
## rm      -0.3200037  0.1553992 -0.62038075
## age      0.2564318 -0.2512574  0.59321281
## dis     -0.2021897  0.2826819 -0.49573024
## rad      0.4531232 -0.4103307  0.47306604
## tax      0.4419243 -0.4184878  0.52339243
## ptratio  1.0000000 -0.1495283  0.35375936
```

```
## b       -0.1495283  1.0000000 -0.37661571
## lstat    0.3537594 -0.3766157  1.00000000
```

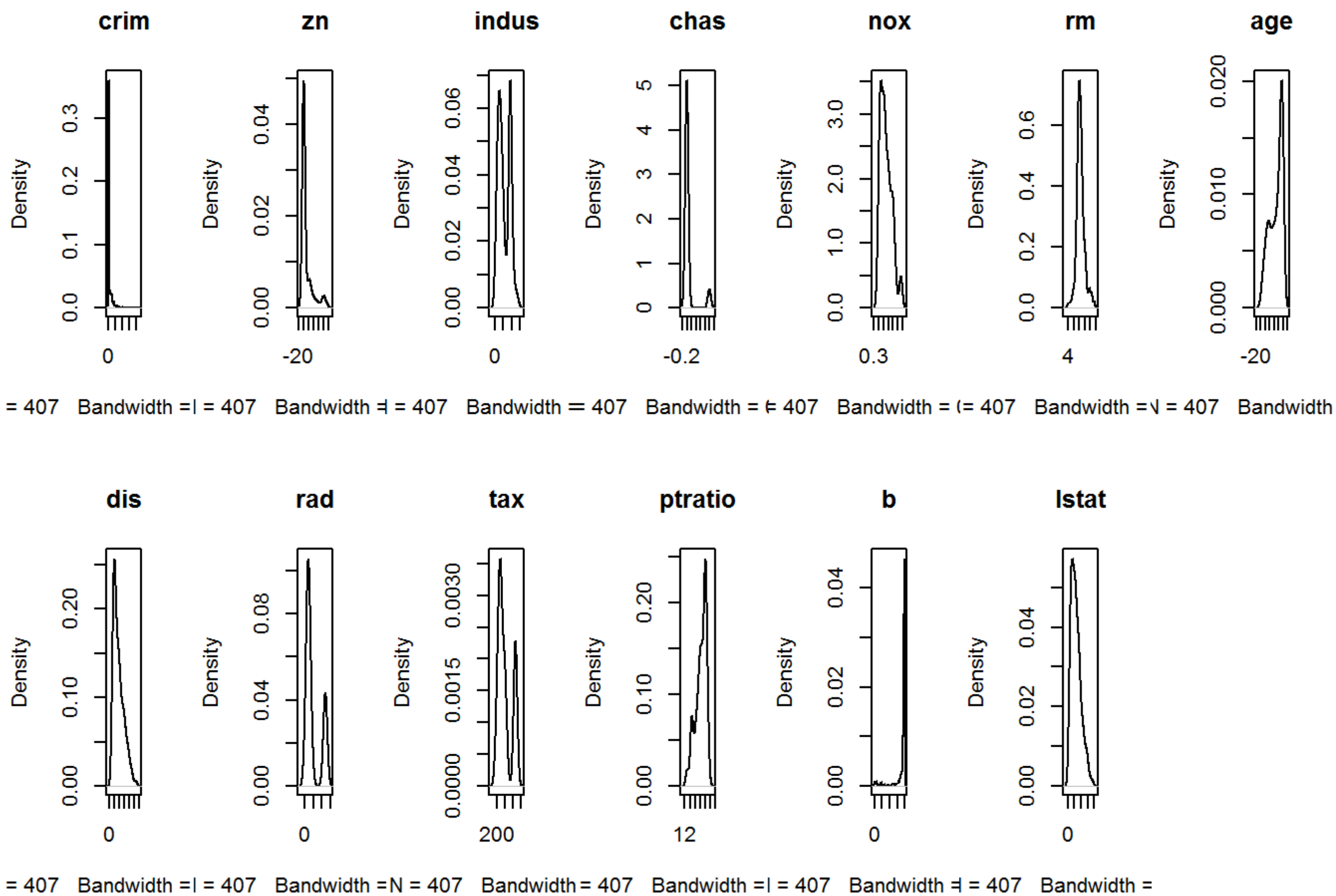
b. Data visualizations histograms for each attribute

```
par(mfrow=c(2,7))
for(i in 1:13) {
    hist(dataset[,i], main=names(dataset)[i])
}
```



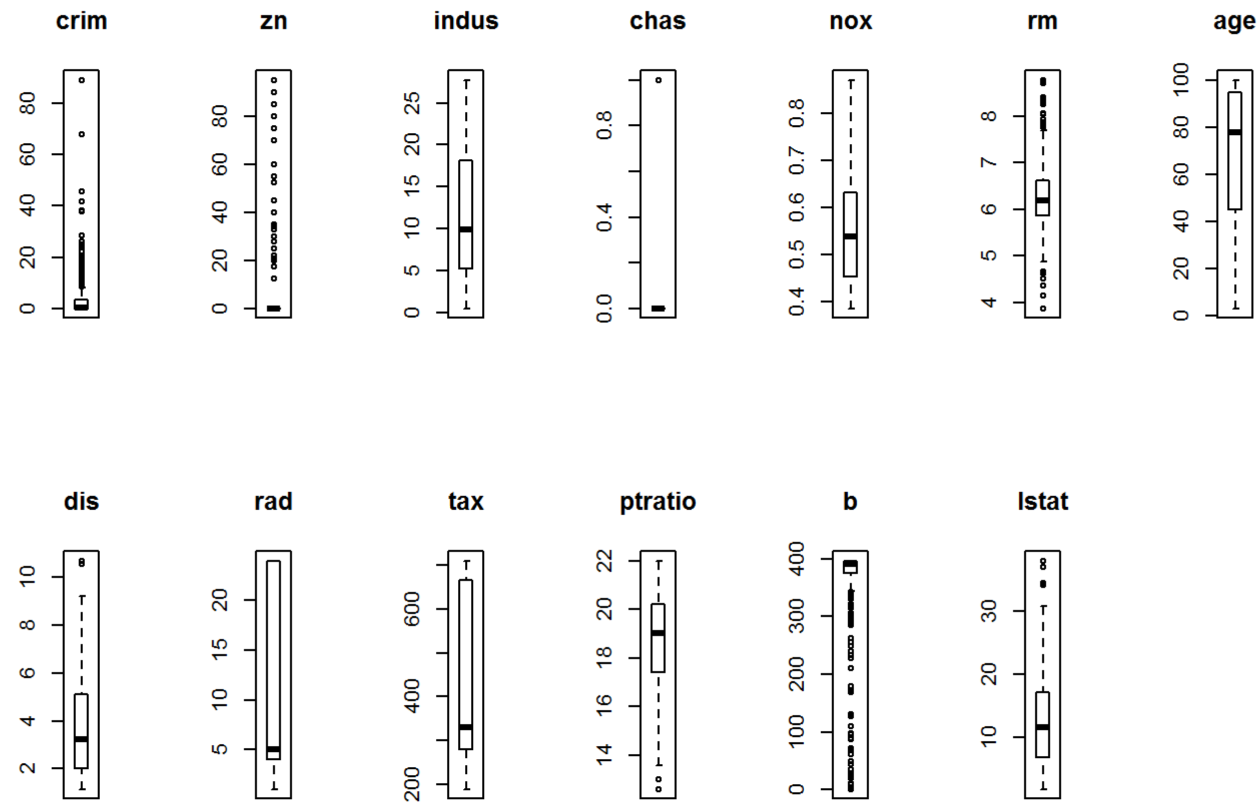
density plot for each attribute

```r
par(mfrow=c(2,7))
for(i in 1:13) {
    plot(density(dataset[,i]), main=names(dataset)[i])
}
```



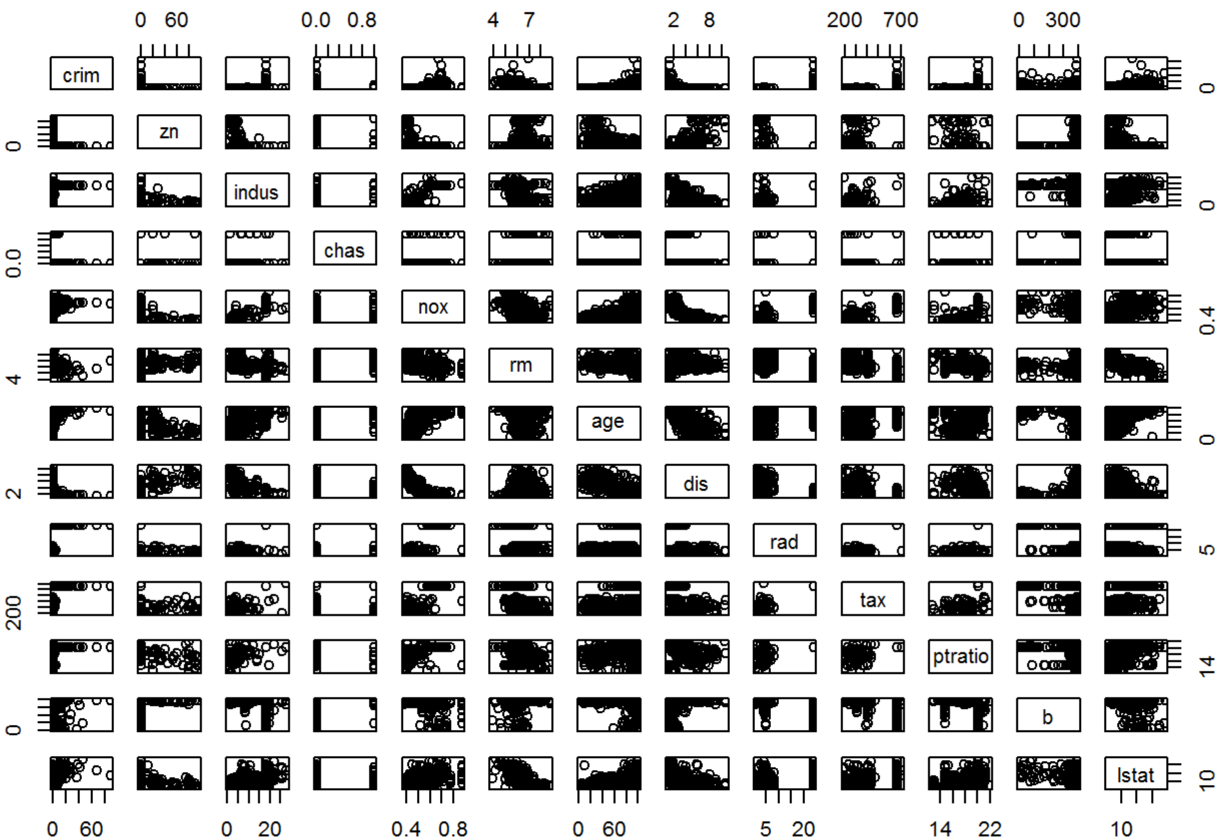
boxplots for each attribute

```r
par(mfrow=c(2,7))
for(i in 1:13) {
    boxplot(dataset[,i], main=names(dataset)[i])
}
```



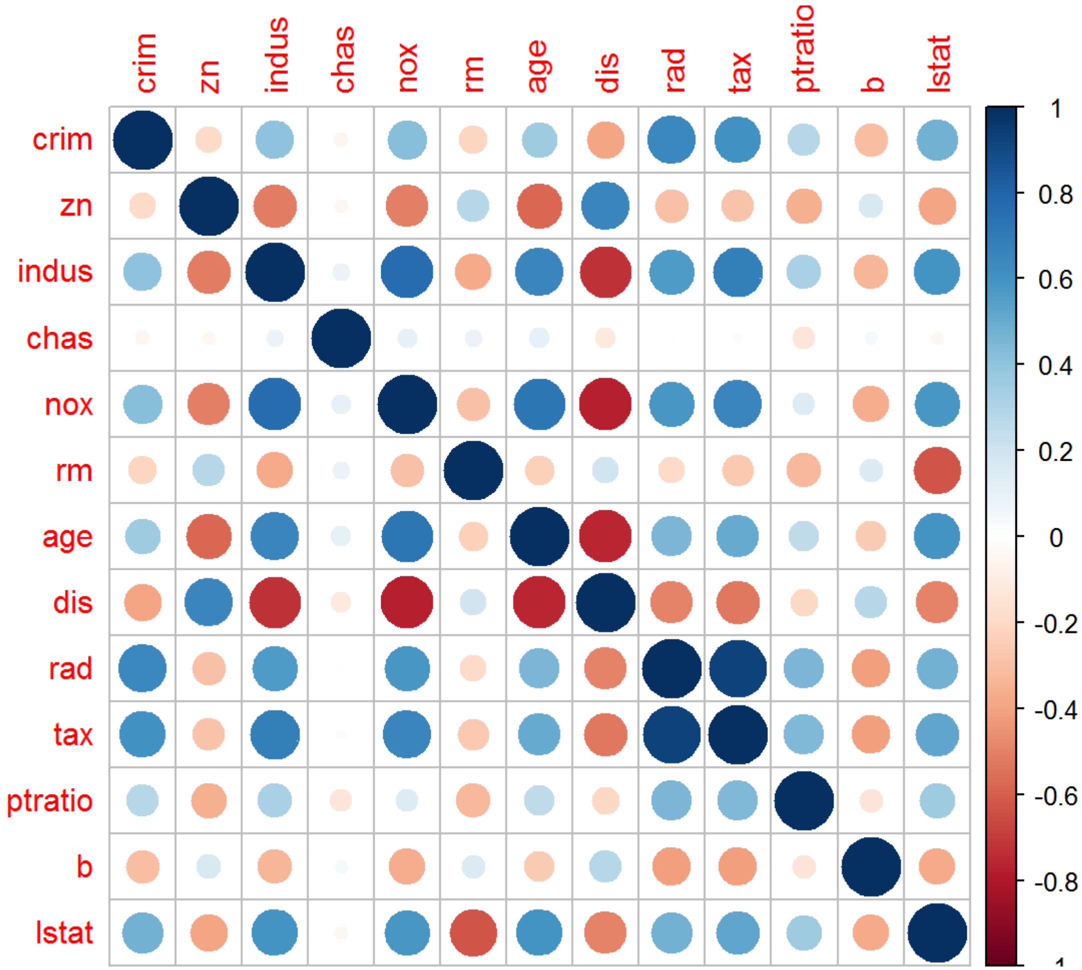
Multivariate Visualizations scatterplot matrix

```
pairs(dataset[,1:13])
```



correlation plot

```
correlations <- cor(dataset[,1:13])
corrplot(correlations, method="circle")
```



3. Evaluate Algorithms a.Run algorithms using 10-fold cross validation

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
```

Fit Linear model

```
set.seed(7)
fit.lm <- train(medv~., data=dataset, method="lm", metric=metric, preProc=c("center", "scale"), trControl=control)
```

Fit Generalized Linear Model

```
set.seed(7)
fit.glm <- train(medv~., data=dataset, method="glm", metric=metric, preProc=c("center", "scale"), trControl=control)
```

Fit GLMNET

```
set.seed(7)
fit.glmnet <- train(medv~., data=dataset, method="glmnet", metric=metric, preProc=c("center", "scale"), trControl=control)
```

```
## Loading required package: glmnet
```

```
## Warning: package 'glmnet' was built under R version 3.3.3
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.3.3
```

```
## Loaded glmnet 2.0-10
```

Fit SVM

```
set.seed(7)
fit.svm <- train(medv~., data=dataset, method="svmRadial", metric=metric, preProc=c("center", "scale"), trControl=control)
```

```
## Loading required package: kernlab
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

Fit CART

```
set.seed(7)
grid <- expand.grid(.cp=c(0, 0.05, 0.1))
fit.cart <- train(medv~., data=dataset, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center", "scale"), trContro
l=control)
```

```
## Loading required package: rpart
```

Fit kNN

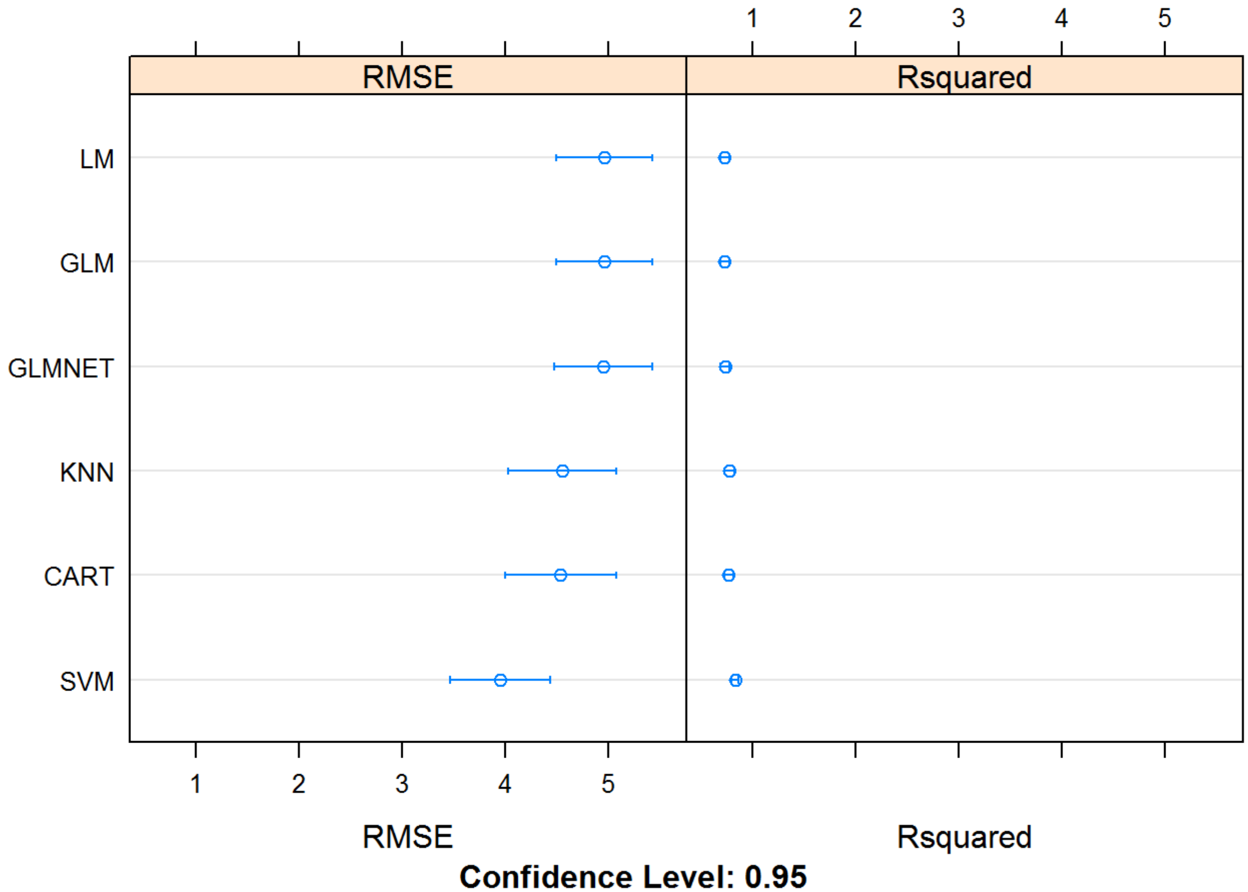
```
set.seed(7)
fit.knn <- train(medv~., data=dataset, method="knn", metric=metric, preProc=c("center", "scale"), trControl=control)
```

Finally Compare algorithms

```
results <- resamples(list(LM=fit.lm, GLM=fit.glm, GLMNET=fit.glmnet, SVM=fit.svm, CART=fit.cart, KNN=fit.knn))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: LM, GLM, GLMNET, SVM, CART, KNN
## Number of resamples: 30
##
## RMSE
##         Min. 1st Qu. Median  Mean 3rd Qu.  Max. NA's
## LM     3.514   4.056  4.773 4.963   5.529 9.448    0
## GLM    3.514   4.056  4.773 4.963   5.529 9.448    0
## GLMNET 3.484   4.017  4.767 4.955   5.520 9.506    0
## SVM    2.377   3.010  3.750 3.952   4.463 8.177    0
## CART   2.797   3.434  4.272 4.541   5.437 9.248    0
## KNN    2.394   3.509  4.471 4.555   5.089 8.757    0
##
## Rsquared
##          Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## LM     0.3169  0.6682 0.7428 0.7293  0.7984 0.8882    0
## GLM    0.3169  0.6682 0.7428 0.7293  0.7984 0.8882    0
## GLMNET 0.3092  0.6670 0.7437 0.7296  0.7989 0.8921    0
## SVM    0.5229  0.7803 0.8513 0.8290  0.8820 0.9418    0
## CART   0.3614  0.6733 0.8197 0.7680  0.8613 0.9026    0
## KNN    0.4313  0.7168 0.8024 0.7732  0.8588 0.9146    0
```

```
dotplot(results)
```



b.Feature Selection

The highly correlated attributes must be excluded (set cut-off = 0.7) to avoid loss of accuracy due to correlated predictors.

```
set.seed(7)
cutoff <- 0.70
correlations <- cor(dataset[,1:13])
highlyCorrelated <- findCorrelation(correlations, cutoff=cutoff)
for (value in highlyCorrelated) {
    print(names(dataset)[value])
}
```

```
## [1] "indus"
## [1] "nox"
## [1] "tax"
## [1] "dis"
```

create a new dataset without highly corrected features.

```
dataset_features <- dataset[,-highlyCorrelated]
dim(dataset_features)
```

```
## [1] 407  10
```

Run algorithms using 10-fold cross validation on dataset after removing correlated predictors.

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
```

Fit lm

```
set.seed(7)
fit.lm <- train(medv~., data=dataset_features, method="lm", metric=metric, preProc=c("center", "scale"), trControl=control)
```

Fit GLM

```
set.seed(7)
fit.glm <- train(medv~., data=dataset_features, method="glm", metric=metric, preProc=c("center", "scale"),
trControl=control)
```

Fit GLMNET

```
set.seed(7)
fit.glmnet <- train(medv~., data=dataset_features, method="glmnet", metric=metric, preProc=c("center", "scale"), trControl=c
ontrol)
```

Fit SVM

```
set.seed(7)
fit.svm <- train(medv~., data=dataset_features, method="svmRadial", metric=metric, preProc=c("center", "scale"), trControl=c
ontrol)
```

Fit CART

```
set.seed(7)
grid <- expand.grid(.cp=c(0, 0.05, 0.1))
fit.cart <- train(medv~., data=dataset_features, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center", "scale"),
 trControl=control)
```

Fit kNN

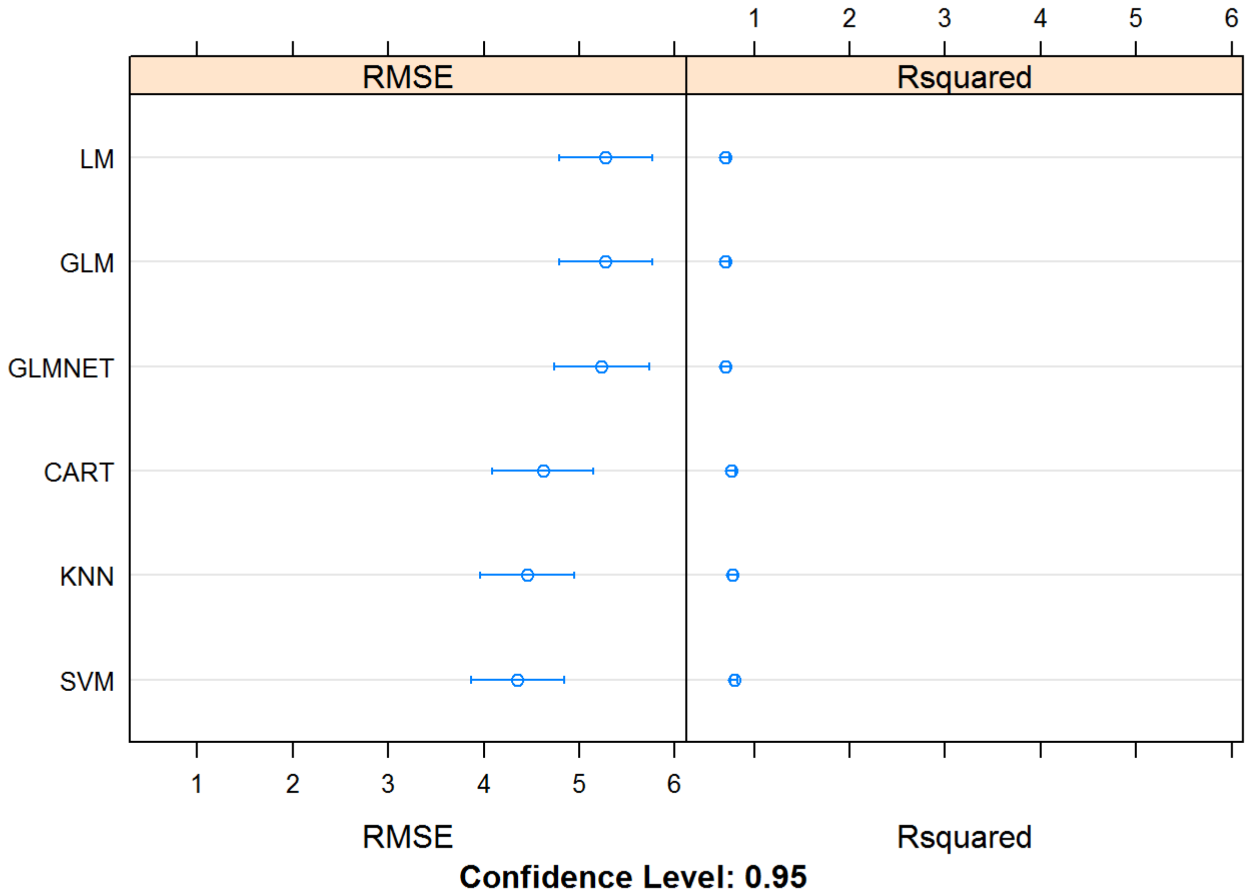
```
set.seed(7)
fit.knn <- train(medv~., data=dataset_features, method="knn", metric=metric, preProc=c("center", "scale"),
trControl=control)
```

Compare algorithms

```
feature_results <- resamples(list(LM=fit.lm, GLM=fit.glm, GLMNET=fit.glmnet, SVM=fit.svm, CART=fit.cart, KNN=fit.knn))
summary(feature_results)
```

```
##
## Call:
## summary.resamples(object = feature_results)
##
## Models: LM, GLM, GLMNET, SVM, CART, KNN
## Number of resamples: 30
##
## RMSE
##         Min. 1st Qu. Median  Mean 3rd Qu.  Max. NA's
## LM     3.431   4.439  4.908 5.277   5.998 9.982    0
## GLM    3.431   4.439  4.908 5.277   5.998 9.982    0
## GLMNET 3.283   4.330  4.950 5.236   5.895 9.869    0
## SVM    2.726   3.337  4.100 4.352   5.036 8.503    0
## CART   2.661   3.550  4.462 4.618   5.246 9.558    0
## KNN    2.488   3.377  4.467 4.453   5.051 8.889    0
##
## Rsquared
##          Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## LM     0.2505  0.6271 0.7125 0.6955  0.7797 0.8877    0
## GLM    0.2505  0.6271 0.7125 0.6955  0.7797 0.8877    0
## GLMNET 0.2581  0.6274 0.7174 0.7027  0.7783 0.8905    0
## SVM    0.4866  0.7522 0.8185 0.7883  0.8673 0.9168    0
## CART   0.3310  0.7067 0.7987 0.7607  0.8363 0.9360    0
## KNN    0.4105  0.7147 0.7981 0.7759  0.8648 0.9117    0
```

```
dotplot(feature_results)
```



c. Apply tranformations and refit models.

Run algorithms using 10-fold cross validation on dataset after removing correlated predictors.

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
```

Fit lm

```
set.seed(7)
fit.lm <- train(medv~., data=dataset_features, method="lm", metric=metric, preProc=c("center", "scale","BoxCox"),
trControl=control)
```

Fit GLM

```
set.seed(7)
fit.glm <- train(medv~., data=dataset_features, method="glm", metric=metric, preProc=c("center", "scale","BoxCox"), trContro
l=control)
```

Fit GLMNET

```
set.seed(7)
fit.glmnet <- train(medv~., data=dataset_features, method="glmnet", metric=metric, preProc=c("center", "scale","BoxCox"), tr
Control=control)
```

Fit SVM

```
set.seed(7)
fit.svm <- train(medv~., data=dataset_features, method="svmRadial", metric=metric, preProc=c("center", "scale","BoxCox"), tr
Control=control)
```

Fit CART

```
set.seed(7)
grid <- expand.grid(.cp=c(0, 0.05, 0.1))
fit.cart <- train(medv~., data=dataset_features, method="rpart", metric=metric, tuneGrid=grid, preProc=c("center",
"scale","BoxCox"), trControl=control)
```

Fit kNN

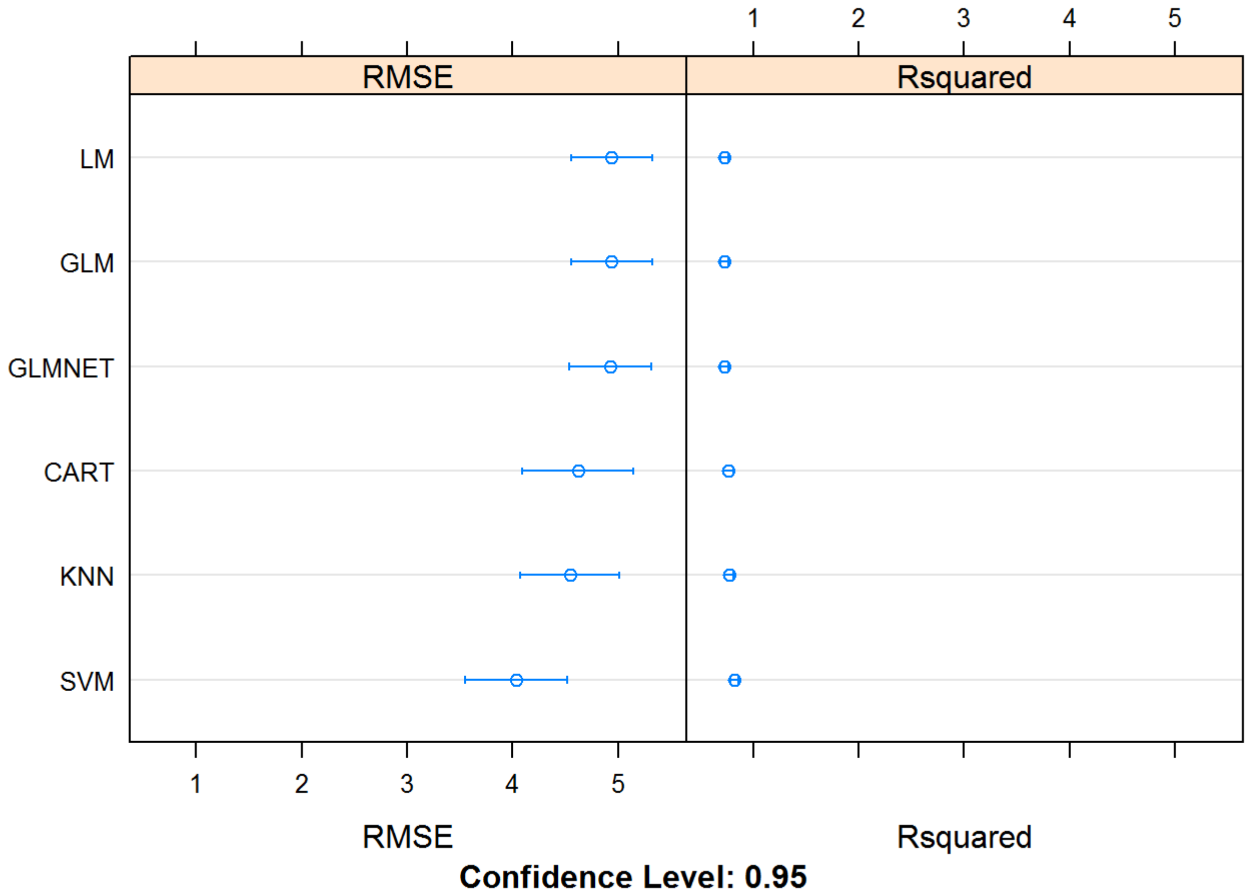
```
set.seed(7)
fit.knn <- train(medv~., data=dataset_features, method="knn", metric=metric, preProc=c("center", "scale","BoxCox"), trContro
l=control)
```

Compare algorithms

```
feature_results <- resamples(list(LM=fit.lm, GLM=fit.glm, GLMNET=fit.glmnet, SVM=fit.svm, CART=fit.cart, KNN=fit.knn))
summary(feature_results)
```

```
##
## Call:
## summary.resamples(object = feature_results)
##
## Models: LM, GLM, GLMNET, SVM, CART, KNN
## Number of resamples: 30
##
## RMSE
##         Min. 1st Qu. Median  Mean 3rd Qu.  Max. NA's
## LM     3.460   4.267  4.727 4.937   5.394 8.382    0
## GLM    3.460   4.267  4.727 4.937   5.394 8.382    0
## GLMNET 3.390   4.202  4.712 4.926   5.546 8.547    0
## SVM    2.489   3.123  3.837 4.030   4.744 8.059    0
## CART   2.661   3.550  4.462 4.619   5.246 9.558    0
## KNN    2.916   3.772  4.419 4.543   5.054 8.914    0
##
## Rsquared
##          Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## LM     0.4635  0.6817 0.7524 0.7283  0.7884 0.8963    0
## GLM    0.4635  0.6817 0.7524 0.7283  0.7884 0.8963    0
## GLMNET 0.4433  0.6773 0.7546 0.7298  0.7882 0.8959    0
## SVM    0.5394  0.7834 0.8424 0.8205  0.8828 0.9366    0
## CART   0.3310  0.7072 0.7987 0.7607  0.8363 0.9360    0
## KNN    0.4040  0.7158 0.8076 0.7724  0.8531 0.9047    0
```

```
dotplot(feature_results)
```



4. Improve Accuracy

a. Algorithm Tuning

Look at parameters

```
print(fit.svm)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 407 samples
##   9 predictor
##
## Pre-processing: centered (9), scaled (9), Box-Cox transformation (7)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 366, 367, 366, 366, 367, 367, ...
## Resampling results across tuning parameters:
##
##   C     RMSE      Rsquared
##   0.25  4.688959  0.7784489
##   0.50  4.284445  0.8036791
##   1.00  4.029789  0.8204519
##
## Tuning parameter 'sigma' was held constant at a value of 0.1304129
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were sigma = 0.1304129 and C = 1.
```

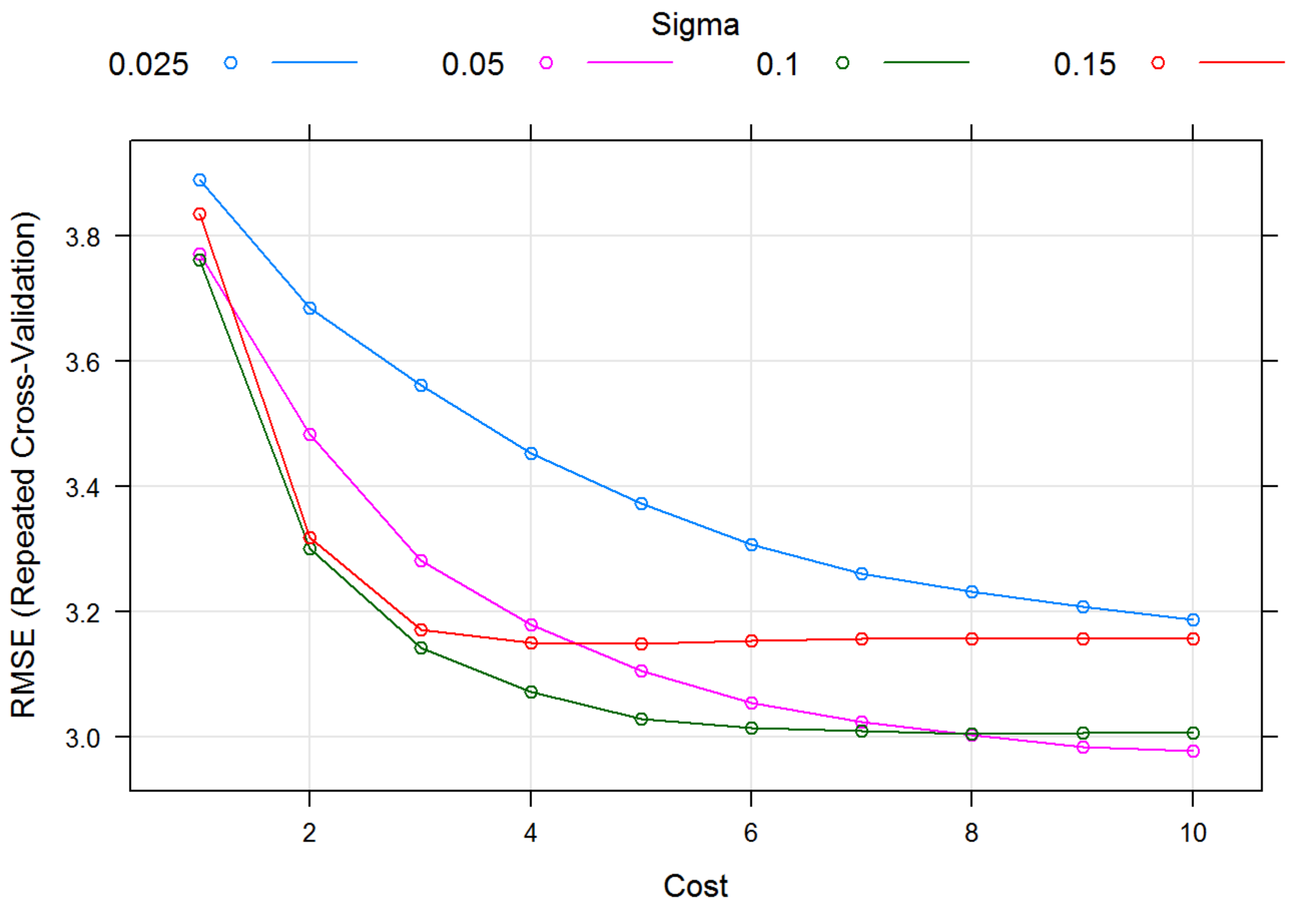
Tune SVM sigma and C parameters

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
set.seed(7)
grid <- expand.grid(.sigma=c(0.025, 0.05, 0.1, 0.15), .C=seq(1, 10, by=1))
fit.svm <- train(medv~., data=dataset, method="svmRadial", metric=metric, tuneGrid=grid, preProc=c("BoxCox"), trControl=cont
rol)
print(fit.svm)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 407 samples
##  13 predictor
##
## Pre-processing: Box-Cox transformation (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 366, 367, 366, 366, 367, 367, ...
## Resampling results across tuning parameters:
##
##   sigma  C   RMSE      Rsquared
##   0.025   1  3.889703  0.8335201
##   0.025   2  3.685009  0.8470869
##   0.025   3  3.562851  0.8553298
##   0.025   4  3.453041  0.8628558
##   0.025   5  3.372501  0.8686287
##   0.025   6  3.306693  0.8731149
##   0.025   7  3.261471  0.8761873
##   0.025   8  3.232191  0.8780827
##   0.025   9  3.208426  0.8797434
##   0.025  10  3.186740  0.8812147
##   0.050   1  3.771428  0.8438368
##   0.050   2  3.484116  0.8634056
##   0.050   3  3.282230  0.8768963
##   0.050   4  3.179856  0.8829293
##   0.050   5  3.105290  0.8873315
##   0.050   6  3.054516  0.8907211
##   0.050   7  3.024010  0.8925927
##   0.050   8  3.003371  0.8936101
##   0.050   9  2.984457  0.8944677
##   0.050  10  2.977085  0.8948000
##   0.100   1  3.762027  0.8453751
##   0.100   2  3.300432  0.8747723
##   0.100   3  3.142907  0.8825268
##   0.100   4  3.071231  0.8862783
##   0.100   5  3.028898  0.8890841
##   0.100   6  3.015042  0.8900253
##   0.100   7  3.009815  0.8904964
##   0.100   8  3.005077  0.8909034
##   0.100   9  3.006147  0.8908668
```

```
##   0.100  10  3.006943  0.8908635
##   0.150   1  3.835849  0.8408209
##   0.150   2  3.318208  0.8716379
##   0.150   3  3.171005  0.8793969
##   0.150   4  3.151071  0.8809872
##   0.150   5  3.149461  0.8811425
##   0.150   6  3.154374  0.8807765
##   0.150   7  3.156741  0.8806358
##   0.150   8  3.157200  0.8806536
##   0.150   9  3.156256  0.8807690
##   0.150  10  3.156134  0.8807506
##
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were sigma = 0.05 and C = 10.
```

```
plot(fit.svm)
```



b. Ensemble methods

train emsemble models

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
```

Fit Random Forest

```
set.seed(7)
fit.rf <- train(medv~., data=dataset, method="rf", metric=metric, preProc=c("BoxCox"), trControl=control)
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Fit Stochastic Gradient Boosting model

```
set.seed(7)
fit.gbm <- train(medv~., data=dataset, method="gbm", metric=metric, preProc=c("BoxCox"), trControl=control, verbose=FALSE)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.3.3
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
## Loading required package: plyr
```

Fit Cubist

```
set.seed(7)
fit.cubist <- train(medv~., data=dataset, method="cubist", metric=metric, preProc=c("BoxCox"), trControl=control)
```

```
## Loading required package: Cubist
```

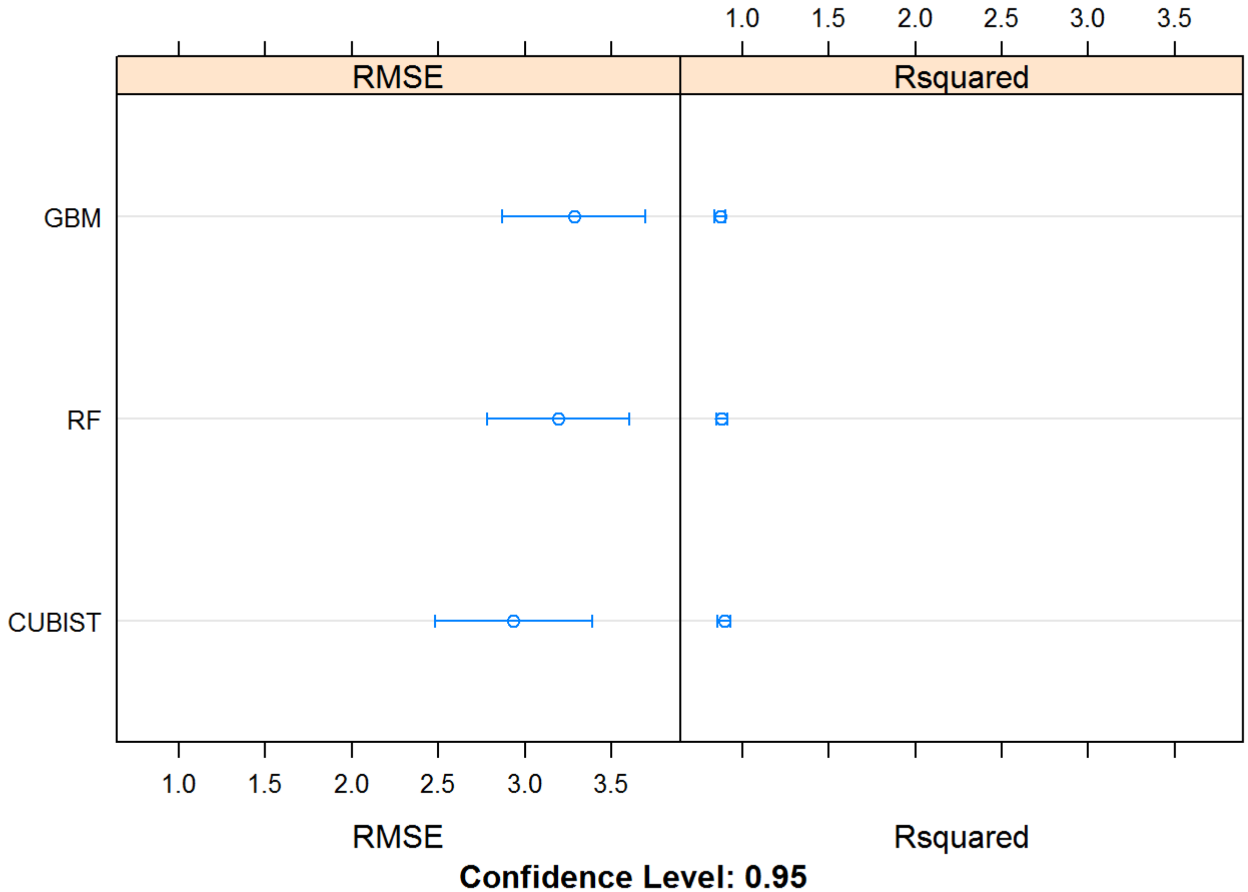
```
## Warning: package 'Cubist' was built under R version 3.3.3
```

Compare algorithms

```
ensemble_results <- resamples(list(RF=fit.rf, GBM=fit.gbm, CUBIST=fit.cubist))
summary(ensemble_results)
```

```
##
## Call:
## summary.resamples(object = ensemble_results)
##
## Models: RF, GBM, CUBIST
## Number of resamples: 30
##
## RMSE
##         Min. 1st Qu. Median  Mean 3rd Qu.  Max. NA's
## RF     2.034   2.416  2.868 3.194   3.637 7.439    0
## GBM    2.194   2.631  2.861 3.283   3.724 7.457    0
## CUBIST 1.671   2.325  2.598 2.935   2.862 7.894    0
##
## Rsquared
##          Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## RF     0.5905  0.8702 0.9176 0.8827  0.9349 0.9565    0
## GBM    0.5934  0.8491 0.9088 0.8720  0.9301 0.9561    0
## CUBIST 0.5312  0.9051 0.9272 0.8945  0.9401 0.9700    0
```

```
dotplot(ensemble_results)
```



Tune Cubist

```
print(fit.cubist)
```

```
## Cubist
##
## 407 samples
##  13 predictor
##
## Pre-processing: Box-Cox transformation (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 366, 367, 366, 366, 367, 367, ...
## Resampling results across tuning parameters:
##
##   committees  neighbors  RMSE      Rsquared
##    1          0          3.805611  0.8291291
##    1          5          3.372092  0.8607419
##    1          9          3.478679  0.8544866
##   10          0          3.321898  0.8684445
##   10          5          3.014602  0.8880220
##   10          9          3.087316  0.8836591
##   20          0          3.248094  0.8747071
##   20          5          2.934577  0.8944885
##   20          9          3.011090  0.8899419
##
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were committees = 20 and neighbors = 5.
```

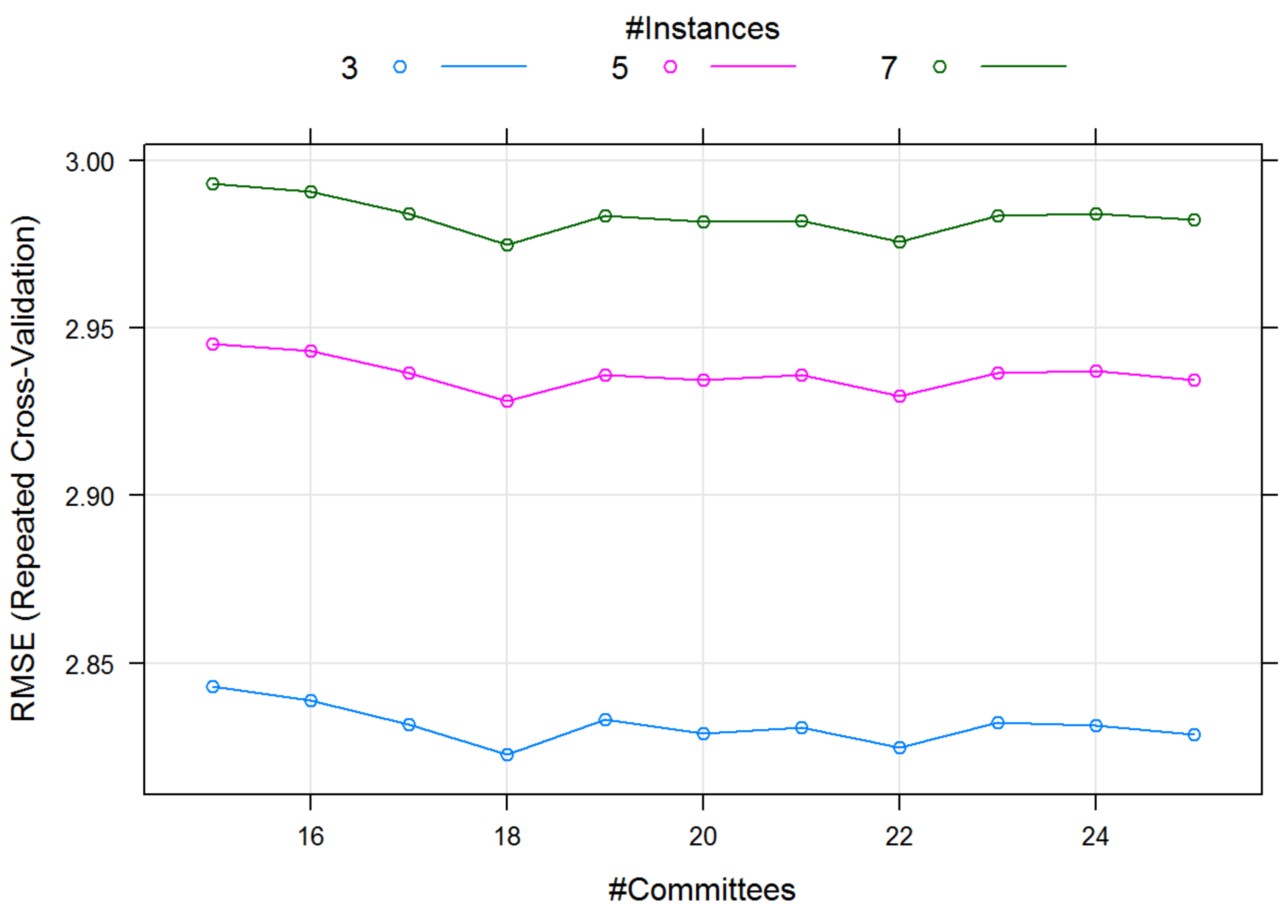
Tune the Cubist algorithm and fine tune it.

```
set.seed(7)
grid <- expand.grid(.committees=seq(15, 25, by=1), .neighbors=c(3, 5, 7))
tune.cubist <- train(medv~., data=dataset, method="cubist", metric=metric, preProc=c("BoxCox"), tuneGrid=grid, trControl=con
trol)
print(tune.cubist)
```

```
## Cubist
##
## 407 samples
##  13 predictor
##
## Pre-processing: Box-Cox transformation (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 366, 367, 366, 366, 367, 367, ...
## Resampling results across tuning parameters:
##
##   committees  neighbors  RMSE      Rsquared
##   15          3          2.843135  0.9009984
##   15          5          2.945379  0.8942976
##   15          7          2.992984  0.8913018
##   16          3          2.838901  0.9006522
##   16          5          2.943103  0.8937805
##   16          7          2.990762  0.8907565
##   17          3          2.831608  0.9014030
##   17          5          2.936655  0.8944879
##   17          7          2.984208  0.8915462
##   18          3          2.822586  0.9018685
##   18          5          2.928190  0.8949810
##   18          7          2.974838  0.8920938
##   19          3          2.833172  0.9015738
##   19          5          2.935970  0.8947952
##   19          7          2.983656  0.8918658
##   20          3          2.828846  0.9014772
##   20          5          2.934577  0.8944885
##   20          7          2.981788  0.8915894
##   21          3          2.830738  0.9015562
##   21          5          2.935917  0.8946249
##   21          7          2.982084  0.8917713
##   22          3          2.824865  0.9017161
##   22          5          2.929831  0.8948180
##   22          7          2.975859  0.8919612
##   23          3          2.832242  0.9014543
##   23          5          2.936506  0.8945625
##   23          7          2.983658  0.8916340
##   24          3          2.831439  0.9012894
##   24          5          2.937095  0.8943113
```

```
##    24          7          2.984098  0.8914109
##    25          3          2.828737  0.9018392
##    25          5          2.934371  0.8948494
##    25          7          2.982422  0.8918831
##
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were committees = 18 and neighbors = 3.
```

```
plot(tune.cubist)
```



5. Finalize Model

a.Apply data transform on training dataset

```
set.seed(7)
x <- dataset[,1:13]
y <- dataset[,14]
preprocessParams <- preProcess(x, method=c("BoxCox"))
trans_x <- predict(preprocessParams, x)
```

b. Train the final model

```
finalModel <- cubist(x=trans_x, y=y, committees=18)
summary(finalModel)
```

```
##
## Call:
## cubist.default(x = trans_x, y = y, committees = 18)
##
##
## Cubist [Release 2.07 GPL Edition]  Tue Aug 08 00:19:16 2017
## ---------------------------------
##
##     Target attribute `outcome'
##
## Read 407 cases (14 attributes) from undefined.data
##
## Model 1:
##
##   Rule 1/1: [84 cases, mean 13.75, range 5 to 25, est err 1.85]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 28.8 - 3.48 lstat + 8.2 nox - 1.41 crim + 5.3 dis + 3e-005 b
##
##   Rule 1/2: [166 cases, mean 19.48, range 7 to 31, est err 2.05]
##
##     if
##  nox <= -0.497006
##  lstat > 2.858393
##     then
##  outcome = 158.68 - 2.35 lstat + 1.8 rad - 75 tax - 2.6 dis
##            - 0.037 ptratio + 10 rm - 0.0075 age + 2.3e-005 b + 1.6 chas
##
##   Rule 1/3: [107 cases, mean 25.59, range 18.6 to 37.2, est err 1.86]
##
##     if
##  rm <= 1.954587
##  dis > 0.5868974
##  lstat <= 2.858393
##     then
##  outcome = 17.94 + 49.3 rm - 4.3 dis - 1.71 lstat - 0.014 age
##            - 0.46 indus - 0.023 ptratio - 36 tax - 0.5 nox + 0.1 rad
##
```

```
##   Rule 1/4: [4 cases, mean 31.15, range 23.3 to 50, est err 1.69]
##
##     if
##  dis <= 0.5868974
##  b <= 66469.73
##  lstat <= 2.858393
##     then
##  outcome = 71.04 - 60.5 dis - 4.99 lstat
##
##   Rule 1/5: [9 cases, mean 38.21, range 17.8 to 50, est err 2.17]
##
##     if
##  rm > 1.954587
##  dis > 0.5868974
##  tax > 1.894297
##     then
##  outcome = 2234.56 - 1152 tax - 11.9 dis - 0.9 lstat + 7.6 rm
##            - 0.012 ptratio
##
##   Rule 1/6: [38 cases, mean 40.13, range 31 to 50, est err 2.55]
##
##     if
##  rm > 1.954587
##  tax <= 1.894297
##     then
##  outcome = 391.64 - 0.000497 b + 80.8 rm - 246 tax - 0.0294 age
##            - 0.047 ptratio - 1.52 lstat - 2.4 dis
##
##   Rule 1/7: [4 cases, mean 50.00, range 50 to 50, est err 0.00]
##
##     if
##  dis <= 0.5868974
##  b > 66469.73
##  lstat <= 2.858393
##     then
##  outcome = 50
##
## Model 2:
##
##   Rule 2/1: [10 cases, mean 7.92, range 5 to 12.3, est err 3.49]
##
```

```
##     if
##  nox > -0.4727541
##  dis <= 0.462979
##  ptratio > 149.145
##     then
##  outcome = 244.69 + 544.4 nox - 0.3 dis - 0.16 lstat
##
##   Rule 2/2: [9 cases, mean 12.56, range 10.2 to 15, est err 3.06]
##
##     if
##  nox <= -0.4727541
##  dis <= 0.462979
##  b > 67032.41
##  lstat > 1.931448
##     then
##  outcome = 31.3 - 0.48 lstat - 0.3 dis - 8 tax + 1.4 rm - 0.004 ptratio
##            - 0.06 indus
##
##   Rule 2/3: [145 cases, mean 18.41, range 7 to 35.2, est err 2.38]
##
##     if
##  dis > 0.462979
##  ptratio > 149.145
##  b <= 77263.3
##  lstat > 1.931448
##     then
##  outcome = 42.95 - 4.01 lstat - 0.042 ptratio + 5.7e-005 b + 5 rm
##            - 0.0055 age - 0.4 dis - 7 tax - 0.06 indus
##
##   Rule 2/4: [116 cases, mean 22.59, range 8.3 to 43.8, est err 2.25]
##
##     if
##  dis > 0.462979
##  tax > 1.857866
##  ptratio > 149.145
##  b > 77263.3
##     then
##  outcome = 120.83 - 0.001783 b + 36.4 rm - 0.026 age - 0.52 lstat
##            - 0.5 dis - 0.11 indus - 10 tax
##
##   Rule 2/5: [74 cases, mean 23.58, range 11.8 to 50, est err 2.37]
```

```
##
##     if
##  ptratio <= 149.145
##  lstat > 1.931448
##     then
##  outcome = -65.34 + 53.6 rm - 0.112 ptratio + 7.7e-005 b - 0.32 lstat
##            - 0.3 dis
##
##   Rule 2/6: [11 cases, mean 23.62, range 6.3 to 50, est err 7.87]
##
##     if
##  dis <= 0.462979
##  ptratio > 149.145
##  b <= 67032.41
##     then
##  outcome = 72.68 - 117.8 dis - 37.4 nox - 4.66 lstat - 0.000222 b
##
##   Rule 2/7: [11 cases, mean 24.03, range 15.7 to 39.8, est err 5.34]
##
##     if
##  tax <= 1.857866
##  lstat > 1.931448
##     then
##  outcome = 130.83 - 0.477 ptratio - 4.17 lstat - 0.0344 age - 0.2 dis
##
##   Rule 2/8: [9 cases, mean 28.52, range 22.5 to 50, est err 7.28]
##
##     if
##  rm <= 1.895669
##  lstat <= 1.931448
##     then
##  outcome = 77.55 - 33.16 lstat + 3.4 rm - 0.7 dis
##
##   Rule 2/9: [61 cases, mean 36.05, range 21 to 50, est err 3.01]
##
##     if
##  rm > 1.895669
##  tax <= 1.894297
##     then
##  outcome = 398.49 + 98.7 rm - 288 tax - 0.0433 age - 7.5 dis - 0.28 lstat
##
```

```
##   Rule 2/10: [13 cases, mean 42.53, range 30.5 to 50, est err 2.91]
##
##     if
##  tax > 1.894297
##  lstat <= 1.931448
##     then
##  outcome = -2.05 + 0.032 age + 21.8 rm - 3.2 dis
##
## Model 3:
##
##   Rule 3/1: [68 cases, mean 13.12, range 5 to 25, est err 2.21]
##
##     if
##  nox > -0.497006
##  ptratio > 109.02
##     then
##  outcome = 51.26 + 41.9 nox - 2.06 crim - 4.11 lstat
##
##   Rule 3/2: [171 cases, mean 19.94, range 7 to 50, est err 2.60]
##
##     if
##  nox <= -0.497006
##  rm <= 1.831781
##  lstat > 1.739722
##     then
##  outcome = 76.77 + 25.7 rm - 0.063 ptratio - 4.2 dis + 0.56 crim
##            - 1.15 lstat - 0.01 age - 42 tax - 0.37 indus + 3e-005 b
##            + 0.012 zn
##
##   Rule 3/3: [35 cases, mean 24.17, range 11.8 to 50, est err 3.79]
##
##     if
##  ptratio <= 109.02
##  lstat > 1.739722
##     then
##  outcome = 113.14 - 0.352 ptratio - 5.85 lstat - 1.3 dis + 5.3 rm
##            - 1.5 nox - 24 tax + 0.0024 age
##
##   Rule 3/4: [118 cases, mean 27.40, range 16.1 to 50, est err 2.09]
##
##     if
```

```
##  nox <= -0.497006
##  rm > 1.831781
##  lstat > 1.739722
##     then
##  outcome = 67.96 + 56.8 rm - 0.05 ptratio - 1.89 lstat - 71 tax - 1.3 dis
##            + 0.24 crim - 0.16 indus - 0.0032 age + 1.3e-005 b + 0.005 zn
##
##   Rule 3/5: [25 cases, mean 38.41, range 24.8 to 50, est err 3.51]
##
##     if
##  dis > 1.162901
##  lstat <= 1.739722
##     then
##  outcome = 283.78 + 96.9 rm - 0.0574 age - 219 tax - 7.6 dis
##            - 0.067 ptratio
##
##   Rule 3/6: [9 cases, mean 49.42, range 44.8 to 50, est err 3.35]
##
##     if
##  dis <= 1.162901
##  lstat <= 1.739722
##     then
##  outcome = 56.35 - 9 dis
##
## Model 4:
##
##   Rule 4/1: [90 cases, mean 13.70, range 5 to 27.9, est err 3.46]
##
##     if
##  dis <= 0.7244036
##  lstat > 2.858393
##     then
##  outcome = 203.21 - 19.6 dis - 7.53 lstat + 0.0679 age - 6.3 nox - 80 tax
##            - 5.1e-005 b - 8 rm
##
##   Rule 4/2: [247 cases, mean 17.48, range 5 to 31, est err 2.91]
##
##     if
##  lstat > 2.858393
##     then
##  outcome = -2.29 + 21.8 rm - 0.0239 age - 2.18 lstat - 0.034 ptratio
```

```
##           + 5.2e-005 b + 2.5 nox - 0.1 crim - 0.3 dis
##
##   Rule 4/3: [41 cases, mean 20.49, range 14.4 to 29.6, est err 2.58]
##
##     if
##  nox <= -1.04499
##  lstat > 2.858393
##     then
##  outcome = 62.87 - 0.000266 b + 10.1 nox - 5.9 dis
##
##   Rule 4/4: [103 cases, mean 25.88, range 18.6 to 50, est err 2.66]
##
##     if
##  rm <= 1.937158
##  lstat <= 2.858393
##     then
##  outcome = -38.51 + 52.8 rm - 4.7 dis - 1.39 lstat - 0.012 age - 0.8 nox
##           + 0.12 crim - 12 tax - 0.006 ptratio - 0.09 indus + 5e-006 b
##
##   Rule 4/5: [47 cases, mean 38.35, range 23.6 to 50, est err 3.20]
##
##     if
##  rm > 1.937158
##  tax <= 1.896025
##     then
##  outcome = 697.02 - 0.000827 b + 102.5 rm - 418 tax - 0.0401 age
##           - 4.8 dis - 0.038 ptratio
##
##   Rule 4/6: [11 cases, mean 38.45, range 21.9 to 50, est err 6.32]
##
##     if
##  rm > 1.937158
##  dis > 0.5868974
##  tax > 1.896025
##  lstat <= 2.858393
##     then
##  outcome = -78.2 + 65 rm - 0.087 ptratio
##
##   Rule 4/7: [8 cases, mean 40.58, range 23.3 to 50, est err 22.46]
##
##     if
```

```
##  dis <= 0.5868974
##  lstat <= 2.858393
##     then
##  outcome = 58.89 - 104.3 dis + 11.72 lstat
##
## Model 5:
##
##   Rule 5/1: [84 cases, mean 13.75, range 5 to 25, est err 3.17]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 40.91 + 12.2 dis - 2.6 crim + 8.3 nox - 2.85 lstat - 10.2 rm
##            + 3e-005 b
##
##   Rule 5/2: [12 cases, mean 15.95, range 13.2 to 23.7, est err 3.47]
##
##     if
##  nox <= -0.497006
##  lstat > 4.439301
##     then
##  outcome = 17.92
##
##   Rule 5/3: [156 cases, mean 19.81, range 10.2 to 29.6, est err 1.91]
##
##     if
##  nox <= -0.497006
##  rm <= 1.831301
##  dis > 0.5626968
##     then
##  outcome = 139.74 + 23.4 rm - 79 tax + 1.8 rad - 0.041 ptratio - 2.8 dis
##            - 1.24 lstat - 0.0069 age + 2.3e-005 b + 1.7 chas + 0.016 zn
##
##   Rule 5/4: [124 cases, mean 23.08, range 7.2 to 50, est err 3.41]
##
##     if
##  rm > 1.831301
##  lstat > 1.987397
##     then
##  outcome = 102.95 + 43.6 rm - 0.053 ptratio - 79 tax - 1.02 lstat
##            - 1.1 dis + 0.5 rad - 0.9 nox + 0.0019 age + 0.008 zn
```

```
##
##   Rule 5/5: [10 cases, mean 26.73, range 7 to 50, est err 9.26]
##
##     if
##  nox <= -0.497006
##  rm <= 1.831301
##  dis <= 0.5626968
##  lstat <= 4.439301
##     then
##  outcome = 116.88 - 24.41 lstat - 3.1 dis
##
##   Rule 5/6: [62 cases, mean 36.86, range 21.9 to 50, est err 4.83]
##
##     if
##  lstat <= 1.987397
##     then
##  outcome = -111.16 + 78.2 rm - 6.3 dis - 1.52 crim - 0.43 lstat
##
##   Rule 5/7: [13 cases, mean 43.86, range 21.9 to 50, est err 9.72]
##
##     if
##  age > 229.474
##  lstat <= 1.987397
##     then
##  outcome = -60.3 + 0.4787 age - 1.09 lstat - 1 dis - 1.2 nox - 14 tax
##            - 0.006 ptratio + 1.6 rm + 0.2 rad
##
## Model 6:
##
##   Rule 6/1: [29 cases, mean 13.33, range 7 to 27.5, est err 4.31]
##
##     if
##  b <= 16084.5
##     then
##  outcome = 19.31 + 0.000779 b - 0.38 lstat - 0.4 dis + 1 rm
##            - 0.003 ptratio - 5 tax
##
##   Rule 6/2: [247 cases, mean 17.48, range 5 to 31, est err 2.54]
##
##     if
##  lstat > 2.858393
```

```
##    then
##  outcome = 67.94 - 3.35 lstat - 0.68 crim - 0.0142 age + 3.3 nox
##            - 0.015 ptratio + 2.6e-005 b + 0.4 rad - 17 tax
##
##   Rule 6/3: [9 cases, mean 21.33, range 15.7 to 29.6, est err 3.41]
##
##    if
##  tax <= 1.857866
##  lstat > 2.858393
##    then
##  outcome = 4.61 + 32.7 dis - 0.36 lstat - 0.003 ptratio + 0.9 rm - 5 tax
##
##   Rule 6/4: [114 cases, mean 28.38, range 18.6 to 50, est err 2.17]
##
##    if
##  dis > 1.230868
##  lstat <= 2.858393
##    then
##  outcome = -84.69 + 73.9 rm - 5.8 dis - 0.0285 age - 0.84 indus
##            - 0.64 lstat + 0.13 crim - 0.7 nox - 0.004 ptratio - 6 tax
##
##   Rule 6/5: [29 cases, mean 33.81, range 20.6 to 50, est err 3.83]
##
##    if
##  dis <= 1.230868
##  b > 73935.01
##  lstat <= 2.858393
##    then
##  outcome = -82.01 - 14.9 dis + 69.6 rm + 0.92 crim - 1.6 lstat - 3.2 nox
##            - 0.48 indus - 0.0095 age
##
##   Rule 6/6: [12 cases, mean 39.08, range 21.9 to 50, est err 8.85]
##
##    if
##  dis <= 0.6604174
##  lstat <= 2.858393
##    then
##  outcome = -5.71 - 62.2 dis + 0.001034 b - 0.21 lstat
##
##   Rule 6/7: [8 cases, mean 41.85, range 25 to 50, est err 11.11]
##
```

```
##     if
##  dis > 0.6604174
##  dis <= 1.230868
##  b <= 73935.01
##  lstat <= 2.858393
##     then
##  outcome = -98.17 + 74.9 rm - 11.2 dis + 0.000142 b + 0.68 crim
##            - 1.17 lstat - 2.3 nox - 0.35 indus - 0.0069 age
##
## Model 7:
##
##   Rule 7/1: [84 cases, mean 13.75, range 5 to 25, est err 2.49]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 25.08 + 11 dis - 2.31 crim - 3.33 lstat + 6.9 nox + 2.8e-005 b
##
##   Rule 7/2: [59 cases, mean 14.73, range 5 to 50, est err 7.01]
##
##     if
##  dis <= 0.5626968
##  lstat > 1.931448
##     then
##  outcome = 16.83 + 0.06 crim + 1.1 rm - 0.003 ptratio - 0.2 dis
##
##   Rule 7/3: [236 cases, mean 21.99, range 10.2 to 50, est err 2.22]
##
##     if
##  nox <= -0.497006
##  dis > 0.5626968
##  tax > 1.865769
##  lstat > 1.931448
##     then
##  outcome = -18.15 + 35.4 rm - 0.0248 age - 0.045 ptratio + 0.58 crim
##            - 2.2 dis + 5.8e-005 b + 1.2 lstat - 1 nox - 9 tax
##
##   Rule 7/4: [10 cases, mean 25.41, range 7 to 50, est err 12.49]
##
##     if
##  nox <= -0.497006
```

```
##  dis <= 0.5626968
##  b <= 67032.41
##     then
##  outcome = 81.16 - 115.7 dis - 0.000217 b - 0.49 lstat
##
##   Rule 7/5: [58 cases, mean 25.55, range 16.5 to 50, est err 2.33]
##
##     if
##  nox <= -0.497006
##  ptratio <= 149.145
##  lstat > 1.931448
##     then
##  outcome = -22.5 + 47.6 rm - 0.089 ptratio + 3.1 nox - 0.66 lstat
##            - 0.6 dis - 12 tax + 0.0019 age + 0.08 crim
##
##   Rule 7/6: [20 cases, mean 29.26, range 15.7 to 50, est err 4.96]
##
##     if
##  tax <= 1.865769
##  ptratio > 149.145
##     then
##  outcome = 53.7 - 0.394 ptratio + 12 dis + 17.6 nox + 2.64 crim + 33.6 rm
##            - 2.74 lstat
##
##   Rule 7/7: [6 cases, mean 29.48, range 22.5 to 50, est err 6.03]
##
##     if
##  rm <= 1.882057
##  lstat <= 1.931448
##     then
##  outcome = 220.69 - 106 rm
##
##   Rule 7/8: [37 cases, mean 38.17, range 22.8 to 50, est err 3.89]
##
##     if
##  rm > 1.882057
##  tax <= 1.894297
##  lstat <= 1.931448
##     then
##  outcome = 767.63 + 111.8 rm - 506 tax - 0.0255 age - 0.33 lstat
##            - 0.5 nox - 0.3 dis - 0.003 ptratio
```

```
##
##   Rule 7/9: [12 cases, mean 41.91, range 30.5 to 50, est err 2.56]
##
##     if
##  rm > 1.882057
##  tax > 1.894297
##  lstat <= 1.931448
##     then
##  outcome = 46.61 + 0.0476 age + 10.8 rm - 1.11 lstat - 1.6 nox - 1 dis
##            - 0.009 ptratio - 16 tax + 0.1 crim
##
## Model 8:
##
##   Rule 8/1: [39 cases, mean 12.55, range 5 to 27.9, est err 3.96]
##
##     if
##  crim > 2.382708
##     then
##  outcome = 54.82 - 5.35 crim - 6.25 lstat
##
##   Rule 8/2: [141 cases, mean 17.25, range 6.3 to 31, est err 2.50]
##
##     if
##  crim <= 2.382708
##  nox > -0.8796992
##  lstat > 2.858393
##     then
##  outcome = 60.02 - 5.63 lstat + 7.7 nox - 0.97 crim + 1.4 dis
##            - 0.0078 age + 3.2 rm + 1.7e-005 b - 0.008 ptratio - 12 tax
##
##   Rule 8/3: [67 cases, mean 20.84, range 14.4 to 29.6, est err 2.09]
##
##     if
##  nox <= -0.8796992
##  lstat > 2.858393
##     then
##  outcome = 48.44 + 8.1 nox - 1.47 lstat + 2.6 rm - 0.5 dis
##            - 0.007 ptratio - 0.11 crim - 9 tax
##
##   Rule 8/4: [160 cases, mean 30.52, range 18.6 to 50, est err 3.26]
##
```

```
##     if
##  lstat <= 2.858393
##     then
##  outcome = -31.85 + 60.1 rm - 6.3 dis - 2.81 lstat - 0.0153 age
##            + 0.27 crim - 18 tax + 0.015 zn - 0.009 ptratio
##
##   Rule 8/5: [8 cases, mean 40.58, range 23.3 to 50, est err 9.14]
##
##     if
##  dis <= 0.5868974
##  lstat <= 2.858393
##     then
##  outcome = 75.2 - 87.2 dis
##
## Model 9:
##
##   Rule 9/1: [83 cases, mean 13.65, range 5 to 25, est err 2.25]
##
##     if
##  nox > -0.497006
##  lstat > 2.150069
##     then
##  outcome = 22.54 + 11.7 dis + 10.8 nox - 1.57 crim + 6.3e-005 b
##            - 0.15 lstat - 6 tax
##
##   Rule 9/2: [29 cases, mean 19.16, range 7 to 50, est err 7.45]
##
##     if
##  nox <= -0.497006
##  rm <= 1.85661
##  dis <= 0.7285143
##  lstat > 2.150069
##     then
##  outcome = 387.41 - 44.9 dis + 2.15 crim - 4.7 lstat - 199 tax + 32 rm
##
##   Rule 9/3: [167 cases, mean 20.49, range 12.7 to 29.6, est err 2.00]
##
##     if
##  nox <= -0.497006
##  rm <= 1.85661
##  dis > 0.7285143
```

```
##     then
##  outcome = 132.11 + 25.2 rm - 0.057 ptratio - 0.0213 age - 3.3 dis
##            - 75 tax + 1.4 rad + 0.49 crim + 0.21 indus + 0.44 lstat
##
##   Rule 9/4: [60 cases, mean 26.86, range 16.1 to 50, est err 2.12]
##
##     if
##  nox <= -0.497006
##  rm > 1.85661
##  lstat > 2.150069
##     then
##  outcome = 117.46 + 69.1 rm - 113 tax - 0.053 ptratio - 0.7 dis
##            - 0.0034 age + 0.14 crim + 0.3 rad
##
##   Rule 9/5: [76 cases, mean 35.06, range 20.6 to 50, est err 5.99]
##
##     if
##  lstat <= 2.150069
##     then
##  outcome = 147.59 + 35.6 rm - 93 tax - 0.046 ptratio - 1.41 lstat
##            + 2.2 nox - 1.4 dis + 0.033 zn + 0.5 rad + 0.0031 age
##            + 0.6 chas
##
##   Rule 9/6: [26 cases, mean 38.57, range 23.6 to 50, est err 4.54]
##
##     if
##  nox <= -0.6286645
##  rm > 1.914272
##  tax > 1.877141
##  lstat <= 2.150069
##     then
##  outcome = -1249.52 + 111.9 rm + 580 tax + 20.9 nox - 0.0495 age
##            - 1.29 lstat - 0.014 ptratio + 0.005 zn
##
##   Rule 9/7: [19 cases, mean 38.87, range 28.5 to 50, est err 4.77]
##
##     if
##  nox <= -0.6286645
##  rm > 1.914272
##  tax <= 1.877141
##  lstat <= 2.150069
```

```
##     then
##  outcome = -191.22 + 159.9 rm + 3.92 lstat + 2.7 nox - 45 tax
##            - 0.022 ptratio + 0.019 zn
##
##   Rule 9/8: [6 cases, mean 45.12, range 21.9 to 50, est err 25.03]
##
##     if
##  nox > -0.6286645
##  lstat <= 2.150069
##     then
##  outcome = -119.53 - 313.9 nox - 9.3 chas
##
## Model 10:
##
##   Rule 10/1: [221 cases, mean 18.34, range 5 to 50, est err 3.01]
##
##     if
##  rm <= 1.831301
##  lstat > 1.931448
##     then
##  outcome = 98.79 - 4.59 lstat - 0.76 crim + 10 rm - 1.9 dis - 41 tax
##            - 0.36 indus + 0.8 rad - 0.017 ptratio
##
##   Rule 10/2: [185 cases, mean 27.55, range 7.2 to 50, est err 3.26]
##
##     if
##  rm > 1.831301
##     then
##  outcome = 143.21 - 6.01 lstat + 34.3 rm - 3.5 dis - 84 tax - 0.33 indus
##            - 0.016 ptratio - 0.11 crim + 0.2 rad
##
##   Rule 10/3: [9 cases, mean 28.52, range 22.5 to 50, est err 5.73]
##
##     if
##  rm <= 1.895669
##  lstat <= 1.931448
##     then
##  outcome = 201.43 - 91.7 rm - 2.4 dis
##
##   Rule 10/4: [40 cases, mean 35.86, range 22.5 to 50, est err 3.20]
##
```

```
##     if
##  crim <= -1.051538
##  lstat <= 1.931448
##     then
##  outcome = -136.13 + 96.1 rm - 7.9 dis - 0.0206 age
##
##   Rule 10/5: [12 cases, mean 46.13, range 31.5 to 50, est err 8.21]
##
##     if
##  crim > -1.051538
##  rm > 1.895669
##  lstat <= 1.931448
##     then
##  outcome = 4.13 + 3.95 crim - 7.6 dis + 28.7 rm - 0.0197 age
##
## Model 11:
##
##   Rule 11/1: [84 cases, mean 13.75, range 5 to 25, est err 2.87]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 41.07 + 15.3 dis + 13.5 nox - 2.3 crim - 1.64 lstat - 13.3 rm
##            + 6.1e-005 b
##
##   Rule 11/2: [169 cases, mean 19.61, range 7 to 33.8, est err 3.09]
##
##     if
##  nox <= -0.497006
##  lstat > 2.848535
##     then
##  outcome = 36.73 - 0.069 ptratio + 0.84 crim - 3.2 dis - 0.0122 age
##            + 4.8e-005 b - 0.33 lstat - 0.4 nox + 1 rm
##
##   Rule 11/3: [157 cases, mean 30.59, range 18.6 to 50, est err 4.09]
##
##     if
##  lstat <= 2.848535
##     then
##  outcome = 54.82 + 87.9 rm - 0.026 age - 0.062 ptratio + 1.01 crim
##            - 91 tax + 4.8 nox - 1.1 dis
```

```
##
##   Rule 11/4: [11 cases, mean 39.32, range 21.9 to 50, est err 28.68]
##
##     if
##  dis <= 0.6492998
##  lstat <= 2.848535
##     then
##  outcome = 58.77 - 179.7 dis - 16.74 crim + 9.48 lstat + 31.1 rm
##
## Model 12:
##
##   Rule 12/1: [300 cases, mean 19.06, range 5 to 50, est err 2.96]
##
##     if
##  rm <= 1.887978
##  lstat > 1.805082
##     then
##  outcome = 118.53 - 5.23 lstat + 13.1 rm - 0.59 indus - 2.2 dis - 53 tax
##
##   Rule 12/2: [67 cases, mean 28.08, range 7.5 to 50, est err 3.89]
##
##     if
##  rm > 1.887978
##  lstat > 1.805082
##     then
##  outcome = 243.66 + 32.5 rm - 143 tax - 4.5 dis - 2.36 lstat - 5 nox
##            - 0.53 indus
##
##   Rule 12/3: [31 cases, mean 38.23, range 22.8 to 50, est err 3.37]
##
##     if
##  tax <= 1.899749
##  lstat <= 1.805082
##     then
##  outcome = -126.48 + 82.5 rm - 3.09 crim - 8.4e-005 b - 0.81 lstat
##
##   Rule 12/4: [9 cases, mean 46.42, range 32.9 to 50, est err 8.91]
##
##     if
##  tax > 1.899749
##  lstat <= 1.805082
```

```
##     then
##  outcome = 53.55
##
## Model 13:
##
##   Rule 13/1: [84 cases, mean 13.75, range 5 to 25, est err 3.29]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 36.89 + 16.5 dis - 3.04 crim + 15.4 nox - 14.3 rm + 7.1e-005 b
##
##   Rule 13/2: [169 cases, mean 19.61, range 7 to 33.8, est err 3.36]
##
##     if
##  nox <= -0.497006
##  lstat > 2.848535
##     then
##  outcome = 215.56 + 3.5 rad - 0.074 ptratio - 98 tax - 3.4 dis
##            - 0.0119 age + 5.5e-005 b + 0.33 indus
##
##   Rule 13/3: [298 cases, mean 24.85, range 7 to 50, est err 3.84]
##
##     if
##  crim <= 0.9690653
##     then
##  outcome = -128.58 + 90.9 rm - 5.5 dis - 0.0176 age + 0.77 crim
##            - 0.028 ptratio
##
##   Rule 13/4: [135 cases, mean 29.21, range 18.6 to 50, est err 2.85]
##
##     if
##  dis > 0.9708925
##  lstat <= 2.848535
##     then
##  outcome = 159.83 + 79.8 rm + 1.65 crim - 0.0294 age - 138 tax - 4.7 dis
##            - 0.058 ptratio
##
##   Rule 13/5: [9 cases, mean 38.40, range 21.9 to 50, est err 11.82]
##
##     if
```

```
##  crim > 0.9690653
##  lstat <= 2.848535
##     then
##  outcome = -9.01 - 41.5 dis + 40.7 rm
##
## Model 14:
##
##   Rule 14/1: [218 cases, mean 18.28, range 5 to 50, est err 3.15]
##
##     if
##  rm <= 1.831301
##  lstat > 2.150069
##     then
##  outcome = 92.35 - 4.33 lstat - 0.7 crim - 35 tax + 5 rm - 0.23 indus
##
##   Rule 14/2: [112 cases, mean 22.42, range 7.2 to 50, est err 2.59]
##
##     if
##  rm > 1.831301
##  tax > 1.857866
##  lstat > 2.150069
##     then
##  outcome = 34.48 - 5.61 lstat + 40.2 rm - 36 tax - 0.33 indus - 0.9 dis
##
##   Rule 14/3: [10 cases, mean 23.18, range 15.7 to 39.8, est err 5.98]
##
##     if
##  tax <= 1.857866
##  lstat > 2.150069
##     then
##  outcome = 32.05 + 30.2 dis - 4.58 lstat + 1.8 rm - 10 tax - 0.09 indus
##
##   Rule 14/4: [8 cases, mean 26.65, range 20.6 to 50, est err 6.28]
##
##     if
##  rm <= 1.849714
##  lstat <= 2.150069
##     then
##  outcome = 260.66 - 127.5 rm - 0.56 lstat - 0.3 dis - 0.07 crim
##
##   Rule 14/5: [68 cases, mean 36.05, range 21.9 to 50, est err 5.20]
```

```
##
##     if
##  rm > 1.849714
##  lstat <= 2.150069
##     then
##  outcome = -143.78 + 95.9 rm - 2.43 crim - 6.6 dis - 0.94 lstat
##            - 0.008 ptratio
##
##   Rule 14/6: [20 cases, mean 40.28, range 21.9 to 50, est err 9.15]
##
##     if
##  rm > 1.849714
##  age > 195.4278
##  lstat <= 2.150069
##     then
##  outcome = 41.37 + 0.1192 age - 10.24 lstat - 0.162 ptratio + 3.1 rm
##
## Model 15:
##
##   Rule 15/1: [84 cases, mean 13.75, range 5 to 25, est err 2.84]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 41.69 + 13.1 dis + 15.4 nox - 2.35 crim - 2 lstat - 11.9 rm
##            + 6e-005 b
##
##   Rule 15/2: [166 cases, mean 19.48, range 7 to 31, est err 2.83]
##
##     if
##  nox <= -0.497006
##  lstat > 2.858393
##     then
##  outcome = 47.22 - 0.081 ptratio - 4.8 dis + 0.99 crim + 12.7 rm
##            - 0.0128 age + 5.1e-005 b - 0.67 lstat - 1 nox + 0.4 rad
##            - 15 tax
##
##   Rule 15/3: [160 cases, mean 30.52, range 18.6 to 50, est err 4.39]
##
##     if
##  lstat <= 2.858393
```

```
##     then
##  outcome = 85.06 + 81 rm + 1.46 crim - 4.8 dis - 0.0258 age
##            - 0.065 ptratio - 100 tax - 0.5 nox - 0.15 lstat + 0.1 rad
##
##   Rule 15/4: [11 cases, mean 39.32, range 21.9 to 50, est err 18.74]
##
##     if
##  dis <= 0.6492998
##  lstat <= 2.858393
##     then
##  outcome = 132.31 - 123.9 dis - 5.64 indus
##
## Model 16:
##
##   Rule 16/1: [62 cases, mean 16.43, range 5 to 50, est err 4.36]
##
##     if
##  dis <= 0.5626968
##     then
##  outcome = 71.14 - 35.8 dis - 10.27 lstat + 1 rm
##
##   Rule 16/2: [345 cases, mean 23.72, range 7.2 to 50, est err 3.33]
##
##     if
##  dis > 0.5626968
##     then
##  outcome = 67.61 + 33.1 rm - 2.21 lstat - 0.65 crim + 3.4 nox - 0.5 indus
##            - 0.0108 age - 51 tax + 4.9e-005 b
##
##   Rule 16/3: [15 cases, mean 29.29, range 15.7 to 50, est err 3.14]
##
##     if
##  tax <= 1.857866
##     then
##  outcome = 144.35 - 5.98 lstat + 18.4 rm - 0.78 indus - 2.8 dis - 67 tax
##
##   Rule 16/4: [6 cases, mean 30.28, range 22.8 to 50, est err 15.17]
##
##     if
##  rm <= 1.895669
##  lstat <= 1.805082
```

```
##     then
##  outcome = 294.96 - 134.8 rm - 0.000128 b - 0.96 lstat
##
##   Rule 16/5: [33 cases, mean 37.96, range 22.8 to 50, est err 4.98]
##
##     if
##  tax <= 1.900249
##  lstat <= 1.805082
##     then
##  outcome = -137.4 + 91.7 rm - 3.85 crim - 0.000181 b - 0.77 lstat
##
##   Rule 16/6: [6 cases, mean 50.00, range 50 to 50, est err 12.43]
##
##     if
##  rm > 1.895669
##  tax > 1.900249
##  lstat <= 1.805082
##     then
##  outcome = 1473.45 - 649 tax - 87.1 rm - 7.3e-005 b - 0.31 lstat
##
## Model 17:
##
##   Rule 17/1: [84 cases, mean 13.75, range 5 to 25, est err 2.83]
##
##     if
##  nox > -0.497006
##     then
##  outcome = 52.92 + 14.8 dis - 3.7 crim - 2.25 lstat - 17.7 rm
##
##   Rule 17/2: [9 cases, mean 13.79, range 7.5 to 21.9, est err 9.65]
##
##     if
##  nox > -0.5267175
##  rm > 1.898219
##     then
##  outcome = 19.26 - 5.03 crim
##
##   Rule 17/3: [243 cases, mean 21.17, range 7 to 50, est err 2.94]
##
##     if
##  nox <= -0.497006
```

```
##  rm <= 1.898219
##     then
##  outcome = 62.1 - 0.074 ptratio - 2.55 lstat - 4.2 dis + 1.8 rad
##            - 0.9 nox + 2.2 rm - 12 tax + 0.0017 age
##
##   Rule 17/4: [47 cases, mean 34.92, range 22 to 50, est err 5.09]
##
##     if
##  nox <= -0.5267175
##  rm > 1.898219
##  tax > 1.877141
##     then
##  outcome = -63.35 - 0.1167 age + 94.1 rm - 19.2 dis + 23.1 nox
##            - 0.111 ptratio - 0.28 lstat
##
##   Rule 17/5: [33 cases, mean 38.21, range 26.6 to 50, est err 4.06]
##
##     if
##  rm > 1.898219
##  tax <= 1.877141
##     then
##  outcome = -195.9 + 126.5 rm - 4.9 dis - 1.69 lstat - 0.0141 age
##            - 0.028 ptratio
##
## Model 18:
##
##   Rule 18/1: [250 cases, mean 17.59, range 5 to 33.8, est err 2.85]
##
##     if
##  lstat > 2.848535
##     then
##  outcome = 176.49 - 2.81 lstat - 92 tax + 15.1 rm - 0.0132 age
##            + 5.8e-005 b + 1 nox - 0.3 dis + 0.05 crim - 0.04 indus
##
##   Rule 18/2: [157 cases, mean 30.59, range 18.6 to 50, est err 4.38]
##
##     if
##  lstat <= 2.848535
##     then
##  outcome = 120.3 + 45 rm - 3.67 lstat - 91 tax - 3.2 nox - 0.6 dis
##            - 0.08 indus + 6e-006 b + 0.05 crim
```

```
##
##   Rule 18/3: [8 cases, mean 40.58, range 23.3 to 50, est err 18.43]
##
##     if
##  dis <= 0.5868974
##  lstat <= 2.848535
##     then
##  outcome = 90.38 - 114.5 dis
##
##
## Evaluation on training data (407 cases):
##
##     Average  |error|               1.72
##     Relative |error|               0.26
##     Correlation coefficient        0.97
##
##
##  Attribute usage:
##    Conds  Model
##
##     69%    85%    lstat
##     37%    52%    nox
##     35%    88%    rm
##     24%    87%    dis
##     12%    73%    tax
##      6%    66%    crim
##      6%    63%    ptratio
##      4%    50%    b
##            65%    age
##            38%    indus
##            27%    rad
##            10%    zn
##             5%    chas
##
##
## Time: 0.4 secs
```

c. Make predictions on validation dataset on compute RMSE

```
set.seed(7)
val_x <- validation[,1:13]
trans_val_x <- predict(preprocessParams, val_x)
val_y <- validation[,14]
predictions <- predict(finalModel, newdata=trans_val_x, neighbors=3)
rmse <- RMSE(predictions, val_y)
r2 <- R2(predictions, val_y)
print(rmse)
```

```
## [1] 2.666336
```

6. Conclusion.

Looking at the RMSE and Rsquared obtained from different sets of models, It is clear that emsemble models have higher predictive power than any individual models. Among the ensemble models, CUbist model appears to have resulted in higher accuracy on the validation dataset.