BUS 41204 Review Session 2

Examples using k-NN and Trees

Chaoxing Dai
chaoxingdai@chicagobooth.edu

01/13/2018

Plan

- Classification using k-NN
- Solve Homework 1 Q2 using
  - Linear and Polynomial Regression
  - k-NN
  - Regression Tree (with cross-validation on a different dataset)

Packages

```
#List the packages we need, install if missing, then load all of them
PackageList =c('MASS','data.table','tree','kknn','rpart','rpart.plot')
NewPackages=PackageList[!(PackageList %in%
                              installed.packages()[,"Package"])]
if(length(NewPackages)) install.packages(NewPackages)

lapply(PackageList,require,character.only=TRUE)#array function
```

```
## Loading required package: MASS

## Warning: package 'MASS' was built under R version 3.4.3

## Loading required package: data.table

## Loading required package: tree

## Loading required package: kknn

## Loading required package: rpart

## Loading required package: rpart.plot

## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
```

```
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
```

```
download.file("https://raw.githubusercontent.com/ChicagoBoothML/HelpR/master/docv.R", "docv.R")
source("docv.R") #this has docvknn used below

set.seed(2018) #Always set the seed for reproducibility
```

Utility Function

```
#Cross-Validation Function
docvlm = function(x,y,set,nfold=10,doran=TRUE,verbose=TRUE,...)
{
  #a little error checking
  x = as.matrix(x)
  set = matrix(set, ncol = 1)
  if(!(is.matrix(x) | is.data.frame(x))) {cat('error in docv: x is not a matrix or data frame\n'); retu
  if(!(is.vector(y))) {cat('error in docv: y is not a vector\n'); return(0)}
  if(!(length(y)==nrow(x))) {cat('error in docv: length(y) != nrow(x)\n'); return(0)}

  nset = nrow(set);
  n=length(y) #get dimensions

  if(n==nfold) doran=FALSE #no need to shuffle if you are doing them all.
  cat('in docv: nset,n,nfold: ',nset,n,nfold,'\n')
  lossv = rep(0,nset) #return values
  if(doran) {ii = sample(1:n,n); y=y[ii]; x=x[ii,,drop=FALSE]} #shuffle rows

  fs = round(n/nfold) # fold size
  for(i in 1:nfold) { #fold loop
    bot = (i-1)*fs+1;
    top = ifelse(i==nfold,n,i*fs);
    ii = bot:top
    if(verbose) cat('on fold: ',i,', range: ',bot,':',top,'\n')
    xin = x[-ii,,drop=FALSE];
    yin=y[-ii];
    xout=x[ii,,drop=FALSE];
    yout=y[ii]
    xin = as.vector(xin)
    xout = as.vector(xout)
    datain = data.frame(x = xin, y = yin)
    dataout = data.frame(x = xout, y = yout)
    for(k in 1:nset)
    { #setting loop
      fit = lm(y ~ poly(x, set[k,]), data = datain)
      yhat = predict(fit,  newdata = dataout)
      lossv[k]=lossv[k]+mean((yout-yhat)^2)
    }
  }
```

```
  return(lossv)
}
```

Classification using K-NN

Data: measurements of Forensic Glass Fragments

```
data(fgl)
help(fgl) # For description
head(fgl,n=3)
```

```
##      RI    Na   Mg   Al    Si    K   Ca Ba Fe type
## 1  3.01 13.64 4.49 1.10 71.78 0.06 8.75  0  0 WinF
## 2 -0.39 13.89 3.60 1.36 72.73 0.48 7.83  0  0 WinF
## 3 -1.82 13.53 3.55 1.54 72.99 0.39 7.78  0  0 WinF
```

```
summary(fgl)
```

```
##        RI                Na              Mg              Al
##  Min.   :-6.8500   Min.   :10.73   Min.   :0.000   Min.   :0.290
##  1st Qu.:-1.4775   1st Qu.:12.91   1st Qu.:2.115   1st Qu.:1.190
##  Median :-0.3200   Median :13.30   Median :3.480   Median :1.360
##  Mean   : 0.3654   Mean   :13.41   Mean   :2.685   Mean   :1.445
##  3rd Qu.: 1.1575   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630
##  Max.   :15.9300   Max.   :17.38   Max.   :4.490   Max.   :3.500
##        Si              K                Ca               Ba
##  Min.   :69.81   Min.   :0.0000   Min.   : 5.430   Min.   :0.000
##  1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.000
##  Median :72.79   Median :0.5550   Median : 8.600   Median :0.000
##  Mean   :72.65   Mean   :0.4971   Mean   : 8.957   Mean   :0.175
##  3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172   3rd Qu.:0.000
##  Max.   :75.41   Max.   :6.2100   Max.   :16.190   Max.   :3.150
##        Fe              type
##  Min.   :0.00000   WinF :70
##  1st Qu.:0.00000   WinNF:76
##  Median :0.00000   Veh  :17
##  Mean   :0.05701   Con  :13
##  3rd Qu.:0.10000   Tabl : 9
##  Max.   :0.51000   Head :29
```

Classification using K-NN

Code the 7 types into 3 main categories

```
n=nrow(fgl)
y = rep(3,n)
y[fgl$type=="WinF"]=1
y[fgl$type=="WinNF"]=2
y = as.factor(y)
levels(y) = c("WinF","WinNF","Other")
print(table(y,fgl$type))
```

```
##
## y       WinF WinNF Veh Con Tabl Head
##   WinF    70     0   0   0    0    0
##   WinNF    0    76   0   0    0    0
##   Other    0     0  17  13    9   29
```

```
x = cbind(fgl$RI,fgl$Na,fgl$Al)
colnames(x) = c("RI","Na","Al") # Refractive Index, Sodium, Aluminium

ddf=data.frame(type=y,x)
```

Classification using K-NN

Run the model.

```
fit0 = kknn(type~.,
            ddf,
            ddf,
            k=10,
            scale=TRUE,
            kernel = "rectangular")
print(table(fit0$fitted,ddf$type))
```

Predicted probabilities

```
fitdf = data.frame(type=ddf$type,fit0$prob)
names(fitdf)[2:4] = c("ProbWinF","ProbWinNF","ProbOther")
head(fitdf,n=3)
par(mfrow=c(1,3))
plot(ProbWinF~type,fitdf,col=c(grey(.5),2:3),cex.lab=1.4)
plot(ProbWinNF~type,fitdf,col=c(grey(.5),2:3),cex.lab=1.4)
plot(ProbOther~type,fitdf,col=c(grey(.5),2:3),cex.lab=1.4)
```

Boston Housing Data

The Boston Housing data is contained in “MASS” package, which is “Boston”

```
data(Boston) #Load data
help(Boston) #For description
sapply(Boston, class) #Check variables Type
```

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

```
summary(Boston) #Summary statistics
```

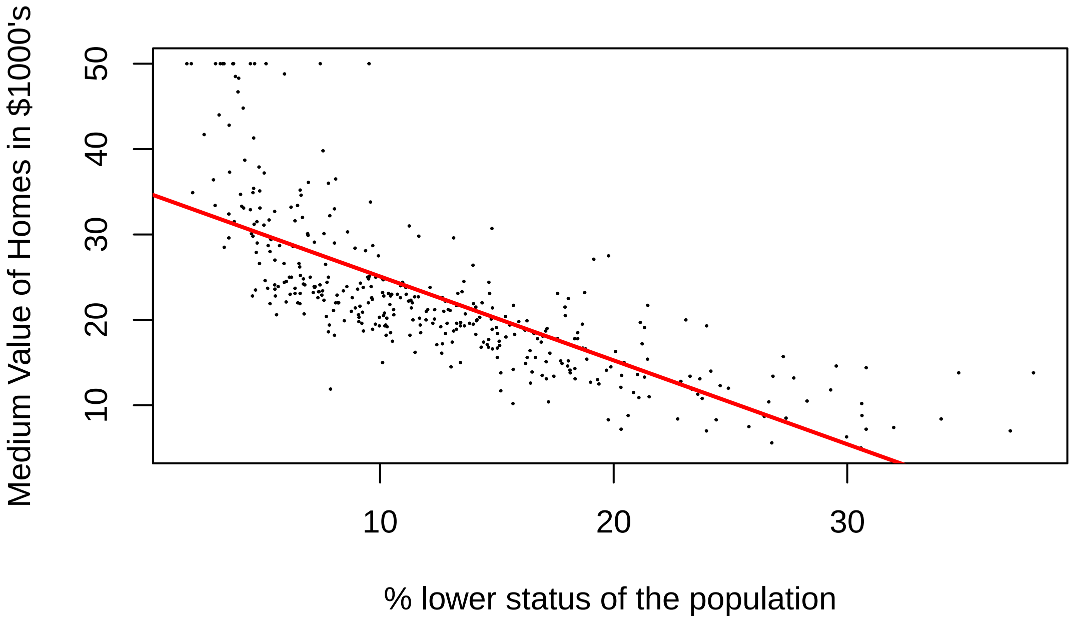
```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
```

```
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

```r
N=dim(Boston)[1]
p=dim(Boston)[2]-1  #One of the columns is response, price


train_indices = sample(N, size = N * 0.75, replace = FALSE) #random partition

boston_train= Boston[train_indices,]
boston_test = Boston[-train_indices,]
```

Simple Linear Regression

```r
fit = lm(medv ~ lstat, data = boston_train)
plot(boston_train$lstat, boston_train$medv, xlab = "% lower status of the population", ylab = "Medium Va
abline(fit, col = "red", lwd = 2)
```




```
cat('RMSE is ',sqrt(mean((boston_test$medv -predict(fit, boston_test))^2)),'\n')
```

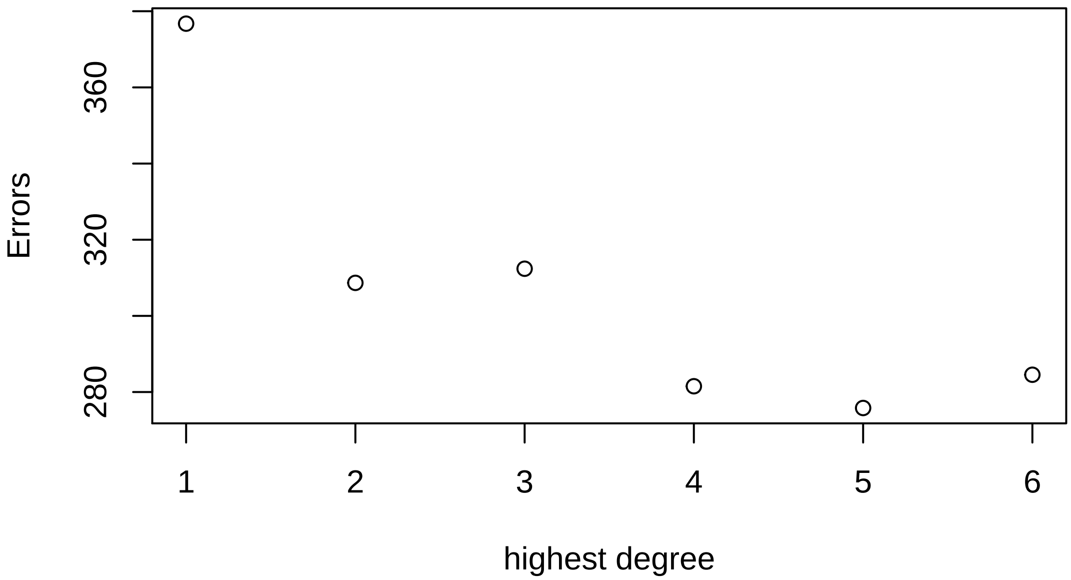
```
## RMSE is  6.543447
```

Polynomial Regression with Cross-Validation

```
D = seq(6)
errors = docvlm(boston_train$lstat, boston_train$medv, D, nfold = 10)
```

```
## in docv: nset,n,nfold:  6 379 10
## on fold:  1 , range:  1 : 38
## on fold:  2 , range:  39 : 76
## on fold:  3 , range:  77 : 114
## on fold:  4 , range:  115 : 152
## on fold:  5 , range:  153 : 190
## on fold:  6 , range:  191 : 228
## on fold:  7 , range:  229 : 266
## on fold:  8 , range:  267 : 304
## on fold:  9 , range:  305 : 342
## on fold:  10 , range:  343 : 379
```

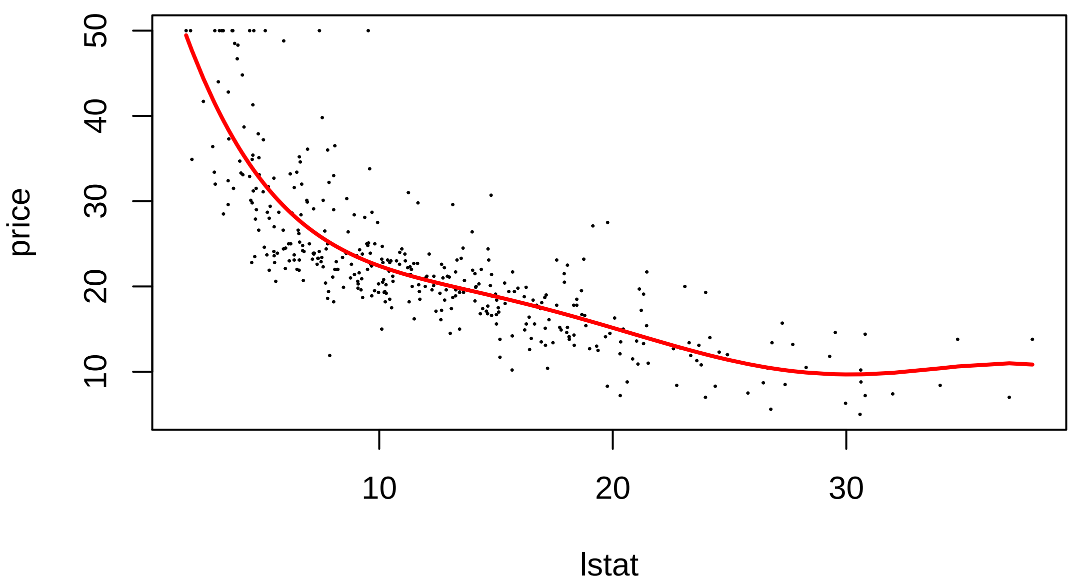
```
plot(D, errors, xlab = "highest degree", ylab = "Errors")
```




```
bestdegree=D[which.min(errors)] # find the best degree
cat('The best degree is : ',bestdegree,'\n')
```

```
## The best degree is :  5
```

```
fit2 = lm(medv ~ poly(lstat, bestdegree), data = boston_train)


fitted = predict(fit2, newdata = data.frame(lstat = sort(boston_train$lstat)))
plot(boston_train$lstat, boston_train$medv, xlab = "lstat", ylab = "price", pch = 1, cex = 0.1)
lines(sort(boston_train$lstat),fitted,col="red",lwd=2, cex.lab=2)
```




```
cat('RMSE is ',sqrt(mean((boston_test$medv -predict(fit2, boston_test))^2)),'\n')
```

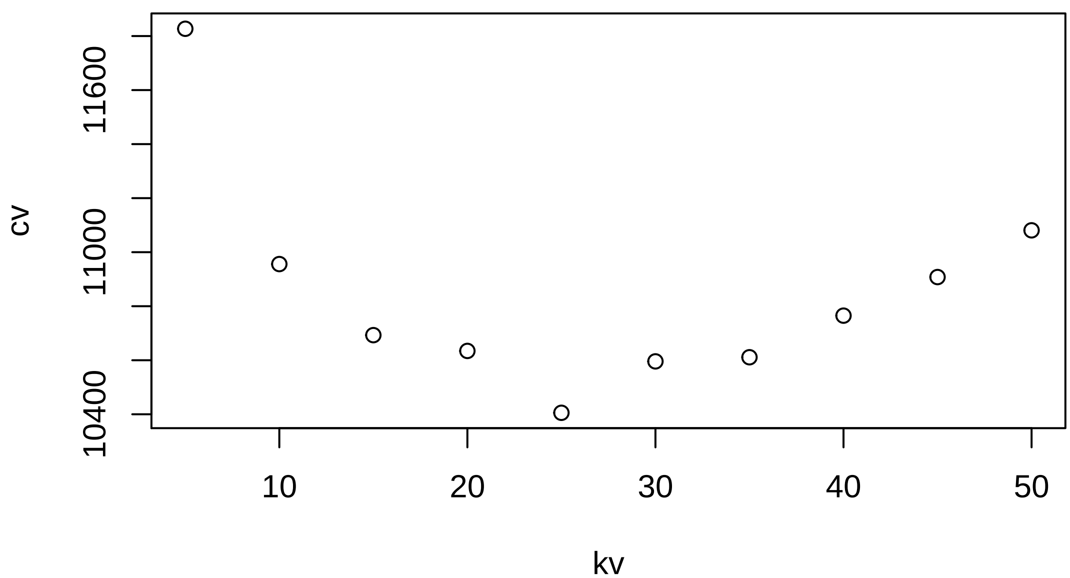
```
## RMSE is  5.309444
```

k-NN

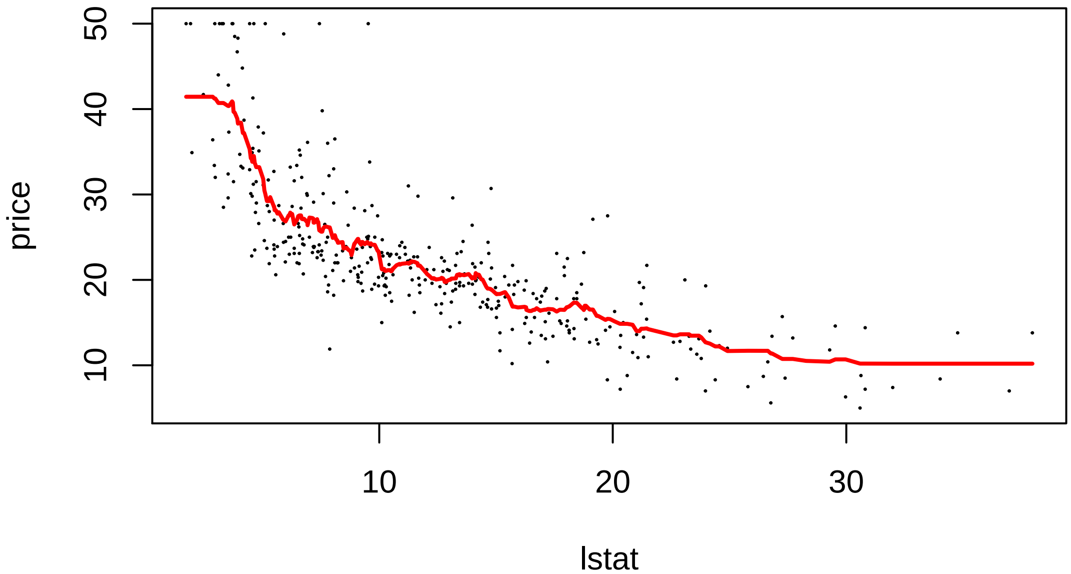
```r
kv=seq(5,50,5)

cv = docvknn(matrix(boston_train$lstat,ncol=1),boston_train$medv,kv,nfold=10)
```

```
## in docv: nset,n,nfold:  10 379 10
## on fold:  1 , range:  1 : 38
## on fold:  2 , range:  39 : 76
## on fold:  3 , range:  77 : 114
## on fold:  4 , range:  115 : 152
## on fold:  5 , range:  153 : 190
## on fold:  6 , range:  191 : 228
## on fold:  7 , range:  229 : 266
## on fold:  8 , range:  267 : 304
## on fold:  9 , range:  305 : 342
## on fold:  10 , range:  343 : 379
```

```r
plot(kv, cv)
```




```r
kbest = kv[which.min(cv)]


# Re-Fit the model using best k
fit3 = kknn(medv~lstat,    # Formula
            boston_train, # Training set
            data.frame(lstat=sort(boston_train$lstat)),# Test set
            k=kbest,      #Number of neighbors considered
            scale=TRUE,   # Scale variable to have equal sd.
            kernel = "rectangular") # Standard unweighted knn


plot(boston_train$lstat, boston_train$medv, xlab = "lstat", ylab = "price", pch = 1, cex = 0.1)
lines(sort(boston_train$lstat),fit3$fitted,col="red",lwd=2, cex.lab=2)
```




```
fit3T = kknn(medv~lstat,boston_train,boston_test,k=kbest,kernel = "rectangular")
cat('RMSE is ',sqrt(mean((boston_test$medv -fit3T$fitted.values)^2)),'\n')
```

```
## RMSE is  5.213411
```

Regresssion Trees

```
fit4 = rpart(medv~lstat,       #Formula
             data=boston_train,#Data
              control=rpart.control(minsplit=5,#the minimum number of observations that must exist in a
                                    cp=0.0001, #complexity, the lower, the larger the tree is
                                    xval=10    #number of cross validations
                                    ))

nbig = length(unique(fit4$where))
cat('size of big tree: ',nbig,'\n')
```

```
## size of big tree:  116
```

```
(cptable = printcp(fit4))
```

```
##
## Regression tree:
## rpart(formula = medv ~ lstat, data = boston_train, control = rpart.control(minsplit = 5,
##     cp = 1e-04, xval = 10))
##
## Variables actually used in tree construction:
## [1] lstat
##
## Root node error: 31984/379 = 84.391
##
## n= 379
##
##             CP nsplit rel error  xerror     xstd
## 1   0.43384272      0   1.00000 1.00521 0.097332
```

```
## 2  0.11370221      1   0.56616 0.66497 0.064711
## 3  0.10921944      2   0.45246 0.52174 0.055673
## 4  0.02170866      3   0.34324 0.42148 0.049458
## 5  0.01335821      4   0.32153 0.38970 0.046570
## 6  0.00983740      5   0.30817 0.37998 0.046259
## 7  0.00837290      6   0.29833 0.37675 0.046141
## 8  0.00710330      7   0.28996 0.39539 0.048211
## 9  0.00616155      9   0.27575 0.41603 0.050155
## 10 0.00400609     13   0.25111 0.40121 0.048405
## 11 0.00366481     15   0.24309 0.41736 0.050353
## 12 0.00349552     16   0.23943 0.41709 0.050621
## 13 0.00318344     17   0.23593 0.41793 0.050558
## 14 0.00264601     19   0.22957 0.43763 0.051350
## 15 0.00231529     21   0.22427 0.45479 0.053304
## 16 0.00224561     23   0.21964 0.45919 0.053395
## 17 0.00214163     25   0.21515 0.46205 0.053420
## 18 0.00186038     29   0.20659 0.45767 0.052823
## 19 0.00174444     32   0.20100 0.45740 0.052814
## 20 0.00162840     33   0.19926 0.45850 0.052402
## 21 0.00159229     34   0.19763 0.46323 0.052519
## 22 0.00155478     35   0.19604 0.46323 0.052519
## 23 0.00145963     36   0.19448 0.46365 0.052511
## 24 0.00130812     38   0.19157 0.46250 0.052514
## 25 0.00126273     40   0.18895 0.46522 0.052542
## 26 0.00122031     42   0.18642 0.46631 0.052543
## 27 0.00119063     43   0.18520 0.46593 0.051864
## 28 0.00112624     44   0.18401 0.47177 0.051968
## 29 0.00104391     45   0.18289 0.47208 0.052316
## 30 0.00103063     46   0.18184 0.47219 0.051706
## 31 0.00099807     47   0.18081 0.47381 0.051678
## 32 0.00098185     48   0.17981 0.47416 0.051672
## 33 0.00090111     49   0.17883 0.47231 0.050866
## 34 0.00089266     52   0.17613 0.46943 0.049706
## 35 0.00080706     55   0.17345 0.47047 0.049711
## 36 0.00075179     56   0.17264 0.47852 0.051317
## 37 0.00072859     57   0.17189 0.48008 0.051352
## 38 0.00072043     60   0.16971 0.48441 0.051407
## 39 0.00070072     65   0.16553 0.48422 0.051411
## 40 0.00065861     68   0.16342 0.48372 0.051420
## 41 0.00065036     69   0.16276 0.48643 0.051439
## 42 0.00062019     70   0.16211 0.48639 0.051441
## 43 0.00052292     72   0.16087 0.48979 0.051497
## 44 0.00049865     73   0.16034 0.49498 0.051930
## 45 0.00047728     75   0.15935 0.49453 0.051920
## 46 0.00046043     76   0.15887 0.49465 0.051910
## 47 0.00044732     77   0.15841 0.49447 0.051886
## 48 0.00042774     79   0.15751 0.49486 0.051887
## 49 0.00042522     80   0.15709 0.49559 0.052036
## 50 0.00038540     81   0.15666 0.49799 0.052013
## 51 0.00037358     82   0.15628 0.50065 0.052115
## 52 0.00036098     83   0.15590 0.49844 0.051873
## 53 0.00035772     85   0.15518 0.49751 0.051876
## 54 0.00034837     86   0.15482 0.49751 0.051876
## 55 0.00030832     88   0.15413 0.49971 0.051917
```

```
## 56 0.00029069     89   0.15382 0.50063 0.052255
## 57 0.00028790     91   0.15324 0.50091 0.052259
## 58 0.00028697     93   0.15266 0.50059 0.052264
## 59 0.00028458     95   0.15209 0.50059 0.052264
## 60 0.00027486     96   0.15180 0.50062 0.052263
## 61 0.00025615     98   0.15125 0.50050 0.052265
## 62 0.00025200     99   0.15100 0.49971 0.052271
## 63 0.00025038    100   0.15074 0.50021 0.052271
## 64 0.00024312    101   0.15049 0.49958 0.052279
## 65 0.00023763    102   0.15025 0.49992 0.052292
## 66 0.00022828    103   0.15001 0.49915 0.052303
## 67 0.00021618    104   0.14978 0.49948 0.052243
## 68 0.00021104    105   0.14957 0.49965 0.052240
## 69 0.00019814    106   0.14936 0.50034 0.052231
## 70 0.00013604    107   0.14916 0.49985 0.052355
## 71 0.00013554    109   0.14889 0.50002 0.052331
## 72 0.00013486    110   0.14875 0.50002 0.052331
## 73 0.00012382    113   0.14835 0.50002 0.052331
## 74 0.00010214    114   0.14822 0.50075 0.052361
## 75 0.00010000    115   0.14812 0.50158 0.052878
```

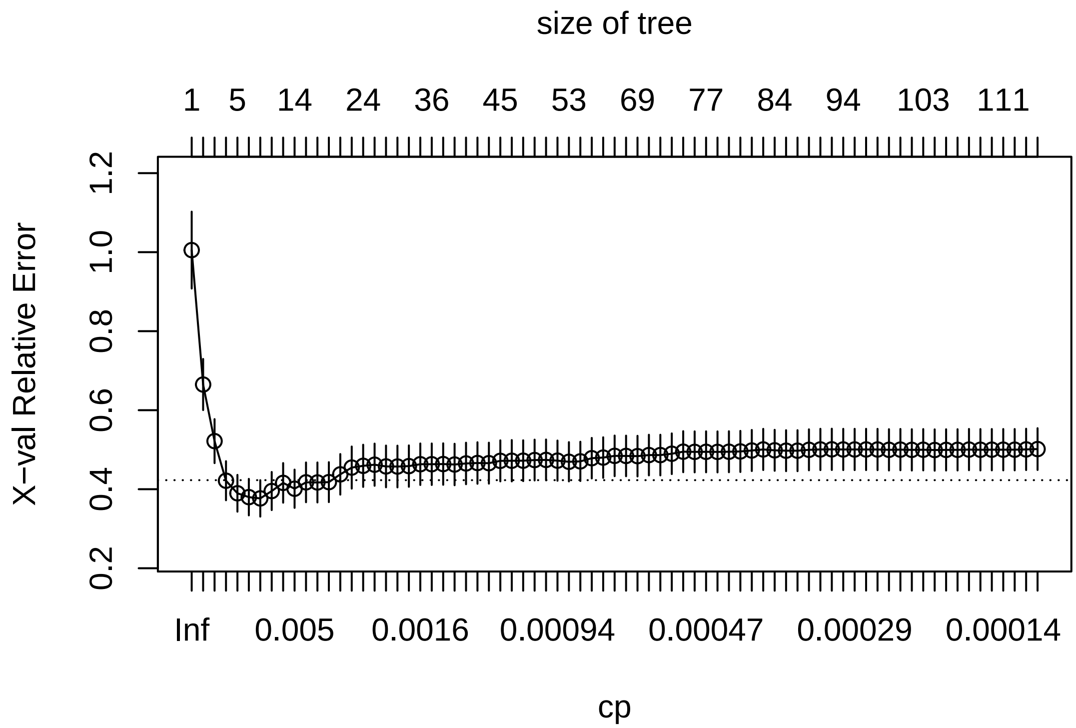
```
##              CP nsplit rel error    xerror       xstd
## 1  0.4338427205      0 1.0000000 1.0052093 0.09733218
## 2  0.1137022065      1 0.5661573 0.6649685 0.06471075
## 3  0.1092194385      2 0.4524551 0.5217373 0.05567270
## 4  0.0217086556      3 0.3432356 0.4214757 0.04945801
## 5  0.0133582131      4 0.3215270 0.3897022 0.04657038
## 6  0.0098374025      5 0.3081688 0.3799794 0.04625885
## 7  0.0083728987      6 0.2983314 0.3767527 0.04614067
## 8  0.0071032969      7 0.2899585 0.3953940 0.04821148
## 9  0.0061615516      9 0.2757519 0.4160291 0.05015547
## 10 0.0040060906     13 0.2511057 0.4012147 0.04840451
## 11 0.0036648096     15 0.2430935 0.4173645 0.05035293
## 12 0.0034955182     16 0.2394287 0.4170924 0.05062072
## 13 0.0031834365     17 0.2359332 0.4179256 0.05055809
## 14 0.0026460135     19 0.2295663 0.4376314 0.05135022
## 15 0.0023152852     21 0.2242743 0.4547863 0.05330378
## 16 0.0022456051     23 0.2196437 0.4591873 0.05339543
## 17 0.0021416348     25 0.2151525 0.4620525 0.05342020
## 18 0.0018603767     29 0.2065859 0.4576694 0.05282318
## 19 0.0017444445     32 0.2010048 0.4573983 0.05281395
## 20 0.0016284036     33 0.1992604 0.4584981 0.05240179
## 21 0.0015922906     34 0.1976320 0.4632343 0.05251941
## 22 0.0015547812     35 0.1960397 0.4632343 0.05251941
## 23 0.0014596291     36 0.1944849 0.4636532 0.05251123
## 24 0.0013081219     38 0.1915656 0.4625034 0.05251388
## 25 0.0012627293     40 0.1889494 0.4652174 0.05254184
## 26 0.0012203079     42 0.1864239 0.4663096 0.05254295
## 27 0.0011906262     43 0.1852036 0.4659259 0.05186445
## 28 0.0011262358     44 0.1840130 0.4717703 0.05196823
## 29 0.0010439054     45 0.1828868 0.4720755 0.05231551
## 30 0.0010306302     46 0.1818428 0.4721949 0.05170641
## 31 0.0009980681     47 0.1808122 0.4738121 0.05167831
## 32 0.0009818494     48 0.1798142 0.4741573 0.05167161
```

```
## 33 0.0009011127     49 0.1788323 0.4723110 0.05086552
## 34 0.0008926578     52 0.1761290 0.4694267 0.04970573
## 35 0.0008070629     55 0.1734510 0.4704682 0.04971088
## 36 0.0007517895     56 0.1726439 0.4785204 0.05131744
## 37 0.0007285933     57 0.1718921 0.4800786 0.05135248
## 38 0.0007204296     60 0.1697064 0.4844098 0.05140665
## 39 0.0007007230     65 0.1655342 0.4842156 0.05141076
## 40 0.0006586090     68 0.1634154 0.4837179 0.05142011
## 41 0.0006503583     69 0.1627568 0.4864348 0.05143934
## 42 0.0006201873     70 0.1621065 0.4863883 0.05144127
## 43 0.0005229234     72 0.1608661 0.4897927 0.05149669
## 44 0.0004986511     73 0.1603432 0.4949793 0.05192955
## 45 0.0004772760     75 0.1593459 0.4945348 0.05192030
## 46 0.0004604344     76 0.1588686 0.4946489 0.05190976
## 47 0.0004473174     77 0.1584082 0.4944657 0.05188616
## 48 0.0004277435     79 0.1575135 0.4948563 0.05188654
## 49 0.0004252236     80 0.1570858 0.4955860 0.05203609
## 50 0.0003853975     81 0.1566605 0.4979876 0.05201281
## 51 0.0003735785     82 0.1562752 0.5006540 0.05211459
## 52 0.0003609780     83 0.1559016 0.4984414 0.05187272
## 53 0.0003577158     85 0.1551796 0.4975104 0.05187609
## 54 0.0003483742     86 0.1548219 0.4975104 0.05187609
## 55 0.0003083180     88 0.1541252 0.4997105 0.05191703
## 56 0.0002906859     89 0.1538168 0.5006296 0.05225490
## 57 0.0002879018     91 0.1532355 0.5009100 0.05225881
## 58 0.0002869673     93 0.1526597 0.5005885 0.05226418
## 59 0.0002845842     95 0.1520857 0.5005885 0.05226418
## 60 0.0002748556     96 0.1518011 0.5006218 0.05226339
## 61 0.0002561544     98 0.1512514 0.5004962 0.05226548
## 62 0.0002520013     99 0.1509953 0.4997091 0.05227058
## 63 0.0002503833    100 0.1507433 0.5002074 0.05227075
## 64 0.0002431194    101 0.1504929 0.4995789 0.05227937
## 65 0.0002376271    102 0.1502498 0.4999249 0.05229188
## 66 0.0002282787    103 0.1500121 0.4991504 0.05230261
## 67 0.0002161776    104 0.1497839 0.4994823 0.05224278
## 68 0.0002110411    105 0.1495677 0.4996527 0.05223984
## 69 0.0001981442    106 0.1493566 0.5003422 0.05223145
## 70 0.0001360407    107 0.1491585 0.4998474 0.05235483
## 71 0.0001355353    109 0.1488864 0.5000231 0.05233098
## 72 0.0001348595    110 0.1487509 0.5000231 0.05233098
## 73 0.0001238212    113 0.1483463 0.5000231 0.05233098
## 74 0.0001021439    114 0.1482225 0.5007545 0.05236086
## 75 0.0001000000    115 0.1481203 0.5015809 0.05287844
```

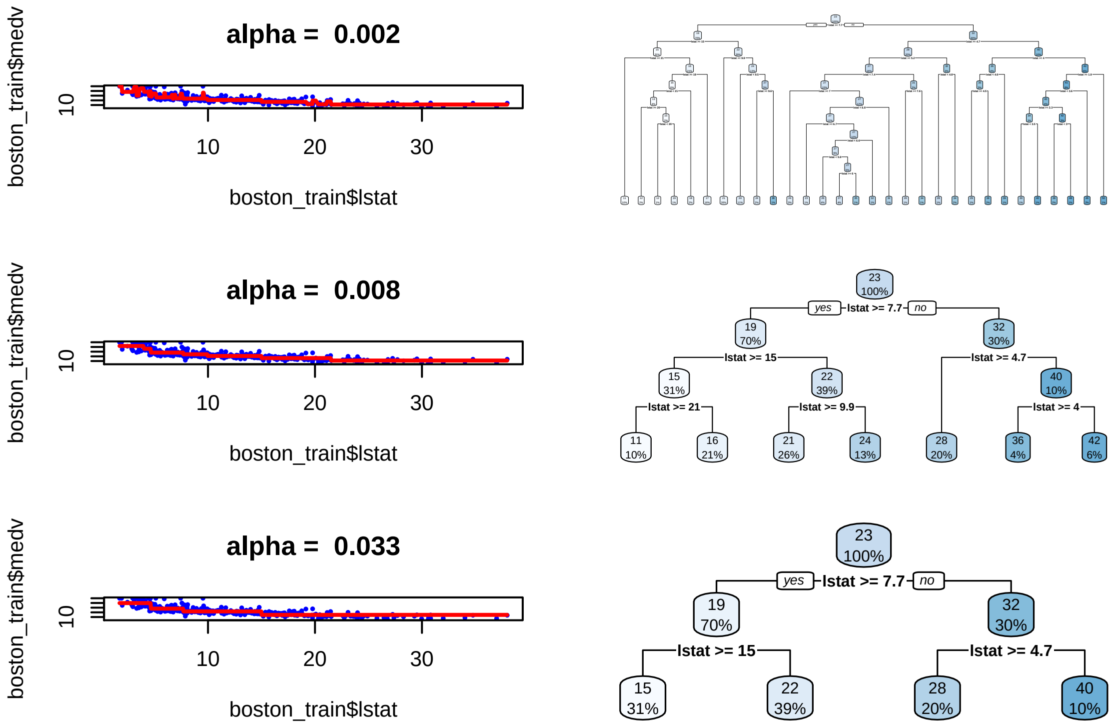
```r
(bestcp = cptable[ which.min(cptable[,"xerror"]), "CP" ])   # this is the optimal cp parameter
```

```
## [1] 0.008372899
```

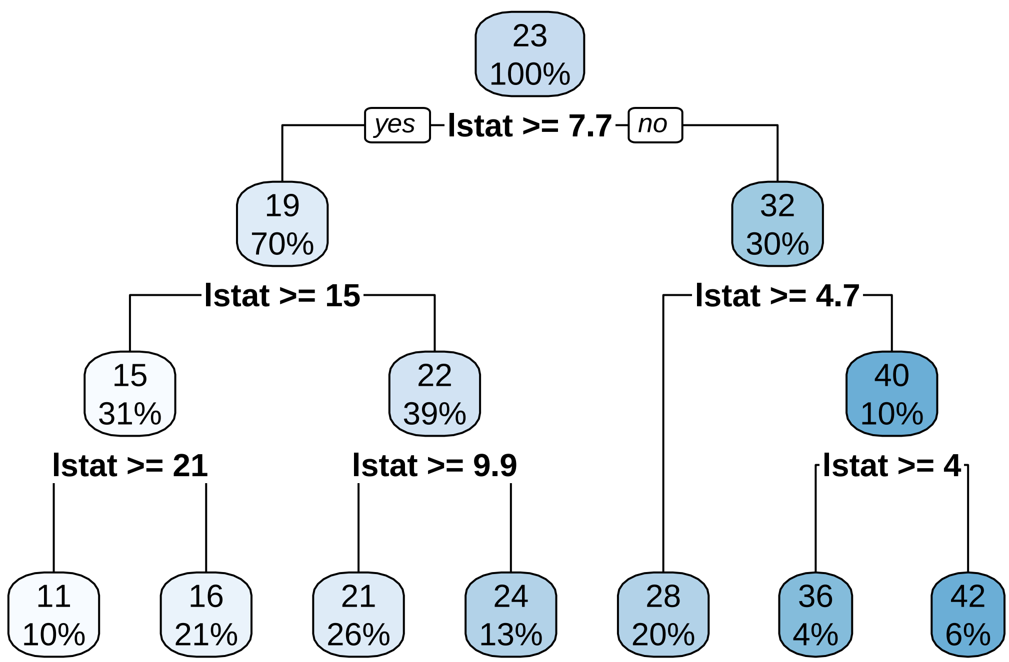
```r
plotcp(fit4) # plot results
```



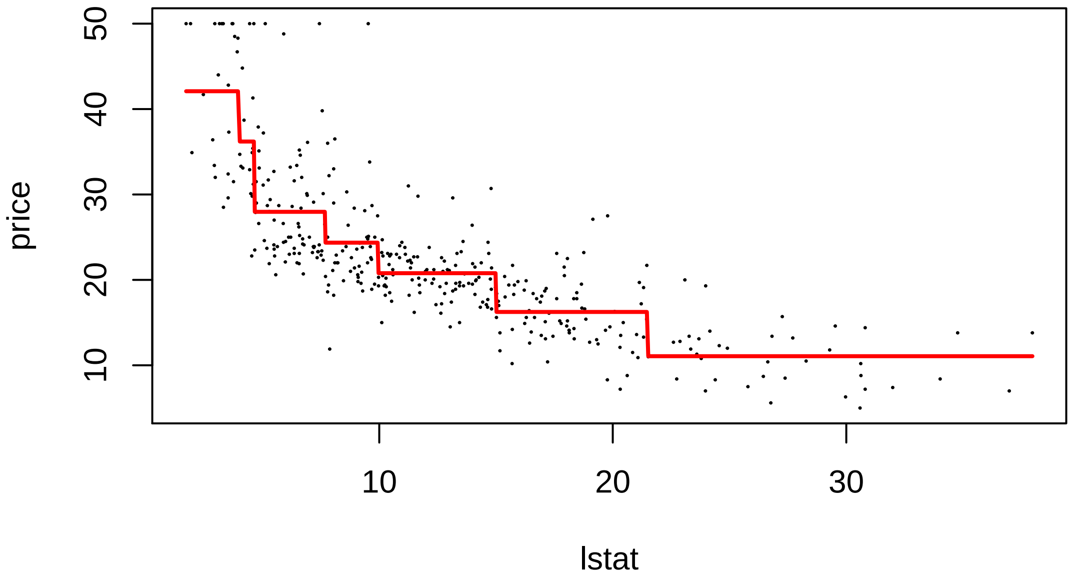

```
# show fit from some trees
oo = order(boston_train$lstat)
cpvec = c(bestcp / 4, bestcp,bestcp*4)

par(mfrow=c(3,2))
for(i in 1:3) {
  plot(boston_train$lstat,boston_train$medv,pch=16,col='blue',cex=.5)
  ptree = prune(fit4,cp=cpvec[i])
  pfit = predict(ptree)
  lines(boston_train$lstat[oo],pfit[oo],col='red',lwd=2)
  title(paste('alpha = ',round(cpvec[i],3)))
  rpart.plot(ptree)
}
```




```r
par(mfrow=c(1,1))
fit4B = prune(fit4,cp=bestcp)
rpart.plot(fit4B)
```




```r
plot(boston_train$lstat, boston_train$medv, xlab = "lstat", ylab = "price", pch = 1, cex = 0.1)
lines(sort(boston_train$lstat),predict(fit4B)[oo],col="red",lwd=2,cex.lab=2)
```




```r
# error
cat('RMSE is ',sqrt(mean((boston_test$medv -predict(fit4B, boston_test))^2)),'\n')
```

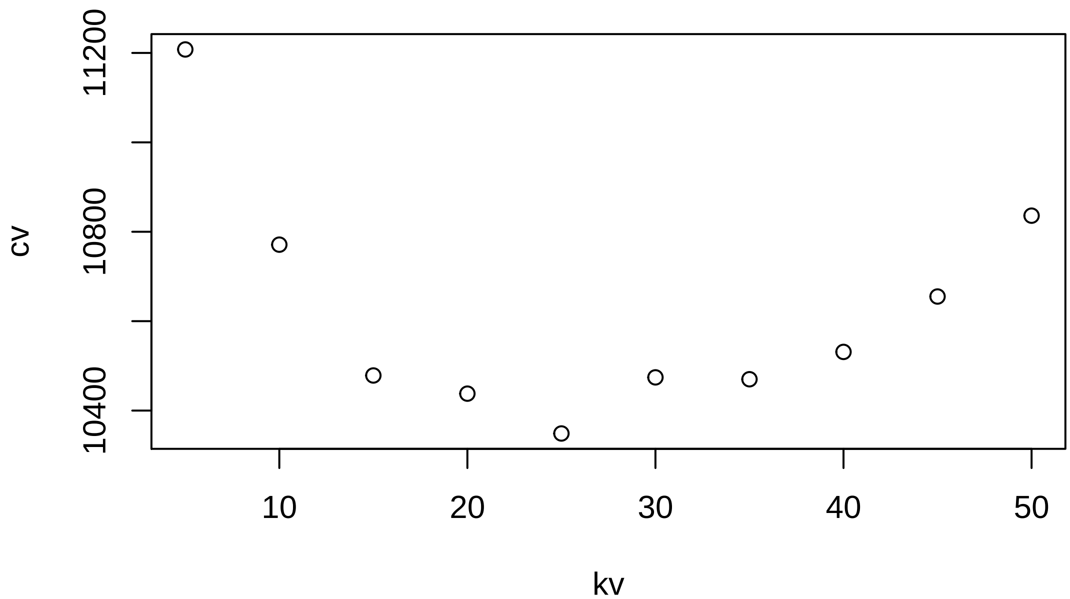
```
## RMSE is  5.266127
```

Adding Variables: k-NN

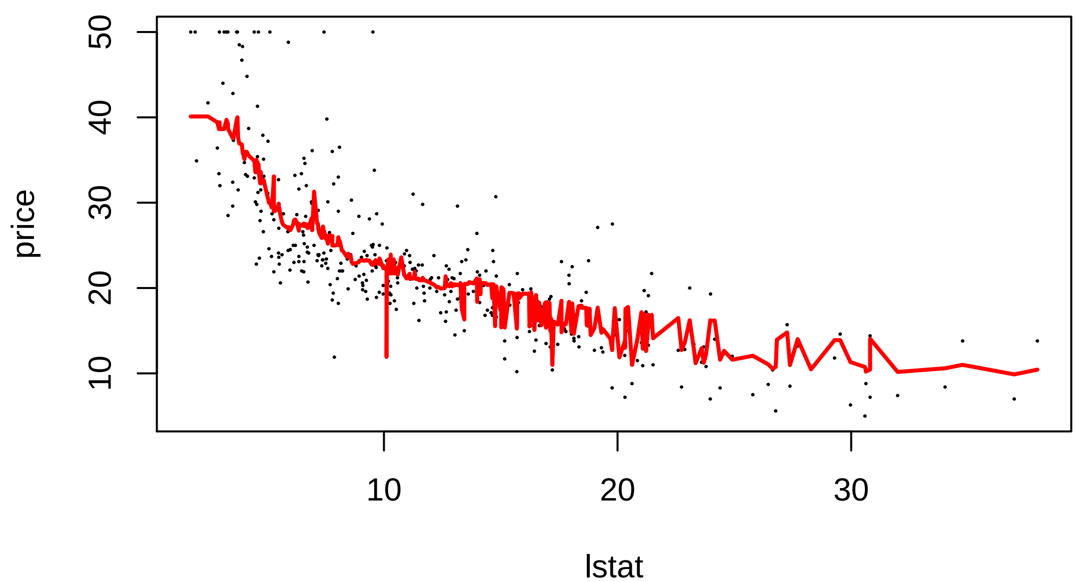
```r
kv=seq(5,50,5)

cv = docvknn(as.matrix(cbind(boston_train$lstat, boston_train$crim)),boston_train$medv,kv,nfold=10)
```

```
## in docv: nset,n,nfold:  10 379 10
## on fold:  1 , range:  1 : 38
## on fold:  2 , range:  39 : 76
## on fold:  3 , range:  77 : 114
## on fold:  4 , range:  115 : 152
## on fold:  5 , range:  153 : 190
## on fold:  6 , range:  191 : 228
## on fold:  7 , range:  229 : 266
## on fold:  8 , range:  267 : 304
## on fold:  9 , range:  305 : 342
## on fold:  10 , range:  343 : 379
```

```r
plot(kv, cv)
```




```
kbest = kv[which.min(cv)]


# Re-Fit the model using best k
fit5 = kknn(medv~lstat+crim,    #Formula
            train=boston_train, # Training Set
            test=boston_train,#test=data.frame(lstat=sort(boston_train$lstat)),# Test set
            k=kbest,       # Number of neighbors considered
            scale=TRUE,
            kernel = "rectangular") # Standard unweighted knn


plot(boston_train$lstat, boston_train$medv, xlab = "lstat", ylab = "price", pch = 1, cex = 0.1)
ord=order(boston_train$lstat)
lines(boston_train$lstat[ord],fit5$fitted[ord],col="red",lwd=2, cex.lab=2)
```



```r
fit5T = kknn(medv~lstat+crim,boston_train,boston_test,k=kbest,kernel = "rectangular")
cat('RMSE is ',sqrt(mean((boston_test$medv -fit5T$fitted.values)^2)),'\n')
```

```
## RMSE is  5.021405
```

Adding Variables: Regression Trees

```r
fit6 = rpart(medv~lstat+crim,       #Formula
             data=boston_train,#Data
              control=rpart.control(minsplit=5,#the minimum number of observations that must exist in a
                                    cp=0.0001, #complexity, the lower, the larger the tree is
                                    xval=10   #number of cross validations
                                    ))

nbig = length(unique(fit6$where))
cat('size of big tree: ',nbig,'\n')
```

```
## size of big tree:  118
```

```r
(cptable = printcp(fit6))
```

```
##
## Regression tree:
## rpart(formula = medv ~ lstat + crim, data = boston_train, control = rpart.control(minsplit = 5,
##     cp = 1e-04, xval = 10))
##
## Variables actually used in tree construction:
## [1] crim  lstat
##
## Root node error: 31984/379 = 84.391
##
## n= 379
##
##            CP nsplit rel error  xerror     xstd
## 1  0.43384272      0  1.000000 1.00438 0.096791
## 2  0.11370221      1  0.566157 0.64298 0.062749
## 3  0.10921944      2  0.452455 0.45715 0.049021
## 4  0.03092312      3  0.343236 0.40798 0.047752
## 5  0.02512200      4  0.312313 0.36979 0.045669
## 6  0.01635459      5  0.287191 0.36402 0.046001
## 7  0.01469593      8  0.238127 0.33470 0.039979
## 8  0.00863611     10  0.208735 0.33139 0.041877
## 9  0.00831637     11  0.200099 0.33429 0.038484
## 10 0.00683488     12  0.191782 0.32725 0.037991
## 11 0.00446285     14  0.178113 0.32531 0.037766
## 12 0.00429931     15  0.173650 0.33299 0.038459
## 13 0.00383545     16  0.169350 0.33723 0.038581
## 14 0.00376802     17  0.165515 0.34242 0.038720
## 15 0.00371956     19  0.157979 0.34929 0.039216
## 16 0.00335749     20  0.154259 0.35100 0.039283
## 17 0.00302694     22  0.147544 0.35639 0.039448
## 18 0.00292567     23  0.144518 0.36439 0.040480
## 19 0.00252183     25  0.138666 0.36192 0.040456
```

```
## 20 0.00238619      26   0.136144 0.36259 0.040660
## 21 0.00231321      27   0.133758 0.36314 0.040743
## 22 0.00213714      28   0.131445 0.35819 0.039076
## 23 0.00210833      29   0.129308 0.36239 0.040414
## 24 0.00205339      30   0.127199 0.36222 0.040425
## 25 0.00199024      32   0.123093 0.36193 0.039860
## 26 0.00194546      33   0.121102 0.36194 0.039867
## 27 0.00190099      34   0.119157 0.36359 0.039901
## 28 0.00182229      35   0.117256 0.36353 0.039927
## 29 0.00181589      36   0.115434 0.36322 0.039932
## 30 0.00175351      37   0.113618 0.37271 0.040213
## 31 0.00173263      38   0.111864 0.37108 0.040186
## 32 0.00172179      41   0.106666 0.37112 0.040185
## 33 0.00154779      42   0.104945 0.38241 0.043433
## 34 0.00145422      44   0.101849 0.38779 0.043539
## 35 0.00138416      45   0.100395 0.38821 0.043541
## 36 0.00133504      47   0.097627 0.39026 0.043567
## 37 0.00131901      52   0.090951 0.39016 0.043558
## 38 0.00118163      53   0.089632 0.39576 0.044025
## 39 0.00115566      54   0.088451 0.39788 0.044247
## 40 0.00109525      55   0.087295 0.39908 0.044264
## 41 0.00101830      57   0.085105 0.40403 0.044377
## 42 0.00097953      59   0.083068 0.40558 0.044460
## 43 0.00093083      60   0.082088 0.40446 0.044465
## 44 0.00090138      61   0.081158 0.40732 0.044475
## 45 0.00074038      62   0.080256 0.40620 0.044447
## 46 0.00072894      63   0.079516 0.40457 0.044254
## 47 0.00066997      64   0.078787 0.40535 0.044245
## 48 0.00065432      66   0.077447 0.40746 0.044231
## 49 0.00064847      69   0.075484 0.40737 0.044233
## 50 0.00050955      72   0.073539 0.41014 0.044312
## 51 0.00048273      73   0.073029 0.42177 0.044884
## 52 0.00048100      74   0.072546 0.42044 0.044879
## 53 0.00044935      76   0.071584 0.42109 0.044882
## 54 0.00043027      77   0.071135 0.42000 0.044876
## 55 0.00042690      78   0.070705 0.41895 0.044817
## 56 0.00042688      81   0.069424 0.41968 0.044841
## 57 0.00039805      82   0.068997 0.42015 0.044835
## 58 0.00038146      86   0.067405 0.42039 0.044847
## 59 0.00034943      87   0.067023 0.42014 0.044848
## 60 0.00033970      90   0.065975 0.42061 0.044852
## 61 0.00033767      91   0.065635 0.42084 0.044852
## 62 0.00032524      92   0.065298 0.42230 0.044873
## 63 0.00031087      94   0.064647 0.42240 0.044871
## 64 0.00030832      95   0.064336 0.42263 0.044863
## 65 0.00026017      96   0.064028 0.42404 0.044890
## 66 0.00025409      97   0.063768 0.42517 0.044879
## 67 0.00024120      98   0.063514 0.42583 0.044873
## 68 0.00022215      99   0.063273 0.42607 0.044871
## 69 0.00021852     100   0.063050 0.42639 0.044868
## 70 0.00018916     101   0.062832 0.42698 0.044865
## 71 0.00018520     102   0.062643 0.42772 0.044853
## 72 0.00014266     103   0.062458 0.42811 0.044835
## 73 0.00013335     104   0.062315 0.43000 0.045030
```

```
## 74 0.00013269    106  0.062048 0.42919 0.044921
## 75 0.00012767    108  0.061783 0.42953 0.044916
## 76 0.00012048    109  0.061655 0.42951 0.044915
## 77 0.00012048    110  0.061535 0.43003 0.044914
## 78 0.00011843    111  0.061414 0.43003 0.044914
## 79 0.00011081    113  0.061177 0.43004 0.044914
## 80 0.00011011    115  0.060956 0.43018 0.044911
## 81 0.00010422    116  0.060846 0.43055 0.044930
## 82 0.00010000    117  0.060741 0.43056 0.044932
```

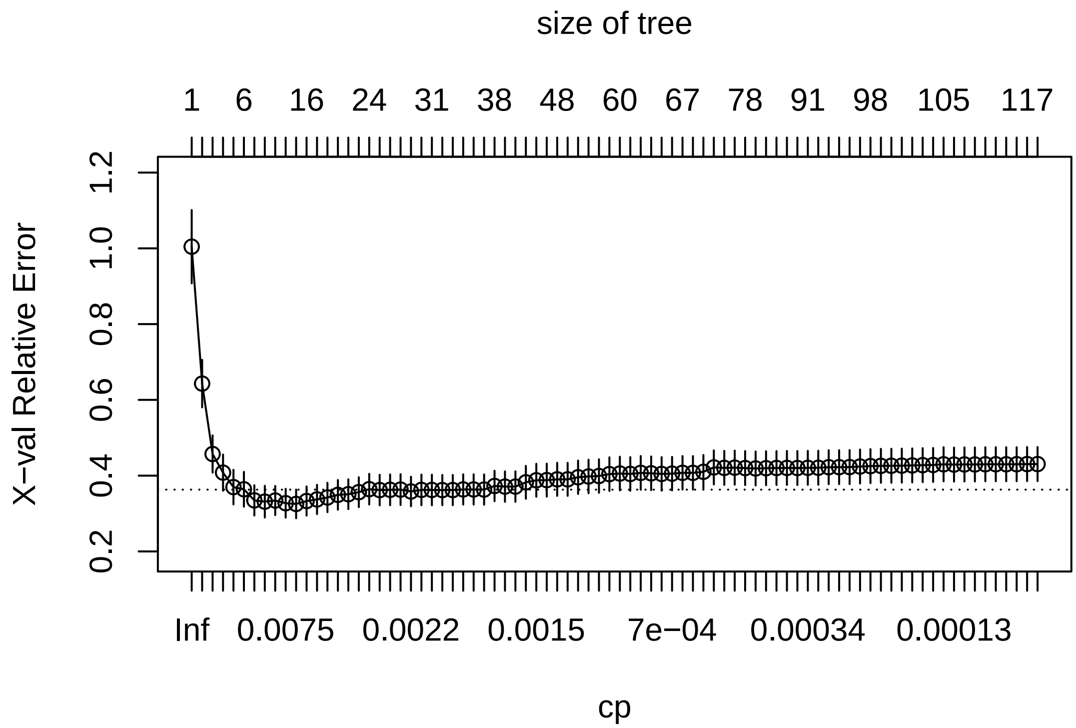
```
##               CP nsplit  rel error    xerror       xstd
## 1  0.4338427205      0 1.00000000 1.0043836 0.09679059
## 2  0.1137022065      1 0.56615728 0.6429755 0.06274874
## 3  0.1092194385      2 0.45245507 0.4571457 0.04902140
## 4  0.0309231230      3 0.34323563 0.4079795 0.04775210
## 5  0.0251219988      4 0.31231251 0.3697931 0.04566883
## 6  0.0163545936      5 0.28719051 0.3640225 0.04600072
## 7  0.0146959315      8 0.23812673 0.3347031 0.03997910
## 8  0.0086361067     10 0.20873487 0.3313861 0.04187657
## 9  0.0083163671     11 0.20009876 0.3342900 0.03848423
## 10 0.0068348774     12 0.19178240 0.3272471 0.03799081
## 11 0.0044628495     14 0.17811264 0.3253058 0.03776590
## 12 0.0042993111     15 0.17364979 0.3329930 0.03845938
## 13 0.0038354525     16 0.16935048 0.3372270 0.03858107
## 14 0.0037680164     17 0.16551503 0.3424194 0.03872048
## 15 0.0037195601     19 0.15797899 0.3492908 0.03921627
## 16 0.0033574882     20 0.15425943 0.3510033 0.03928266
## 17 0.0030269406     22 0.14754446 0.3563889 0.03944750
## 18 0.0029256674     23 0.14451752 0.3643876 0.04047964
## 19 0.0025218275     25 0.13866618 0.3619194 0.04045585
## 20 0.0023861949     26 0.13614436 0.3625868 0.04066047
## 21 0.0023132105     27 0.13375816 0.3631435 0.04074303
## 22 0.0021371414     28 0.13144495 0.3581877 0.03907647
## 23 0.0021083267     29 0.12930781 0.3623897 0.04041394
## 24 0.0020533853     30 0.12719948 0.3622165 0.04042519
## 25 0.0019902375     32 0.12309271 0.3619340 0.03986043
## 26 0.0019454638     33 0.12110247 0.3619448 0.03986685
## 27 0.0019009853     34 0.11915701 0.3635874 0.03990098
## 28 0.0018222895     35 0.11725602 0.3635267 0.03992676
## 29 0.0018158915     36 0.11543373 0.3632246 0.03993202
## 30 0.0017535088     37 0.11361784 0.3727059 0.04021285
## 31 0.0017326289     38 0.11186433 0.3710799 0.04018634
## 32 0.0017217923     41 0.10666645 0.3711167 0.04018547
## 33 0.0015477874     42 0.10494466 0.3824072 0.04343322
## 34 0.0014542174     44 0.10184908 0.3877894 0.04353926
## 35 0.0013841580     45 0.10039486 0.3882137 0.04354083
## 36 0.0013350447     47 0.09762655 0.3902597 0.04356658
## 37 0.0013190069     52 0.09095132 0.3901640 0.04355850
## 38 0.0011816274     53 0.08963232 0.3957584 0.04402511
## 39 0.0011556611     54 0.08845069 0.3978753 0.04424747
## 40 0.0010952453     55 0.08729503 0.3990842 0.04426395
## 41 0.0010182976     57 0.08510454 0.4040313 0.04437689
## 42 0.0009795322     59 0.08306794 0.4055784 0.04446023
## 43 0.0009308301     60 0.08208841 0.4044639 0.04446484
```

```
## 44 0.0009013800     61 0.08115758 0.4073192 0.04447452
## 45 0.0007403786     62 0.08025620 0.4061951 0.04444655
## 46 0.0007289392     63 0.07951582 0.4045695 0.04425427
## 47 0.0006699741     64 0.07878688 0.4053529 0.04424498
## 48 0.0006543207     66 0.07744693 0.4074585 0.04423145
## 49 0.0006484697     69 0.07548397 0.4073741 0.04423342
## 50 0.0005095508     72 0.07353856 0.4101357 0.04431228
## 51 0.0004827297     73 0.07302901 0.4217743 0.04488388
## 52 0.0004809980     74 0.07254628 0.4204377 0.04487900
## 53 0.0004493492     76 0.07158429 0.4210928 0.04488167
## 54 0.0004302708     77 0.07113494 0.4200005 0.04487627
## 55 0.0004269035     78 0.07070467 0.4189526 0.04481658
## 56 0.0004268762     81 0.06942396 0.4196761 0.04484072
## 57 0.0003980523     82 0.06899708 0.4201492 0.04483519
## 58 0.0003814633     86 0.06740487 0.4203934 0.04484713
## 59 0.0003494265     87 0.06702341 0.4201390 0.04484785
## 60 0.0003397005     90 0.06597513 0.4206143 0.04485236
## 61 0.0003376658     91 0.06563543 0.4208423 0.04485153
## 62 0.0003252449     92 0.06529776 0.4223037 0.04487297
## 63 0.0003108706     94 0.06464727 0.4223990 0.04487070
## 64 0.0003083180     95 0.06433640 0.4226290 0.04486266
## 65 0.0002601694     96 0.06402808 0.4240420 0.04489042
## 66 0.0002540930     97 0.06376791 0.4251675 0.04487863
## 67 0.0002411959     98 0.06351382 0.4258303 0.04487309
## 68 0.0002221507     99 0.06327262 0.4260699 0.04487142
## 69 0.0002185164    100 0.06305047 0.4263854 0.04486787
## 70 0.0001891554    101 0.06283196 0.4269843 0.04486548
## 71 0.0001852009    102 0.06264280 0.4277193 0.04485290
## 72 0.0001426638    103 0.06245760 0.4281117 0.04483486
## 73 0.0001333501    104 0.06231494 0.4299952 0.04503003
## 74 0.0001326895    106 0.06204824 0.4291940 0.04492062
## 75 0.0001276668    108 0.06178286 0.4295259 0.04491642
## 76 0.0001204758    109 0.06165519 0.4295104 0.04491544
## 77 0.0001204758    110 0.06153472 0.4300253 0.04491429
## 78 0.0001184262    111 0.06141424 0.4300253 0.04491429
## 79 0.0001108120    113 0.06117739 0.4300359 0.04491402
## 80 0.0001101136    115 0.06095576 0.4301765 0.04491075
## 81 0.0001042178    116 0.06084565 0.4305498 0.04493041
## 82 0.0001000000    117 0.06074143 0.4305577 0.04493240
```

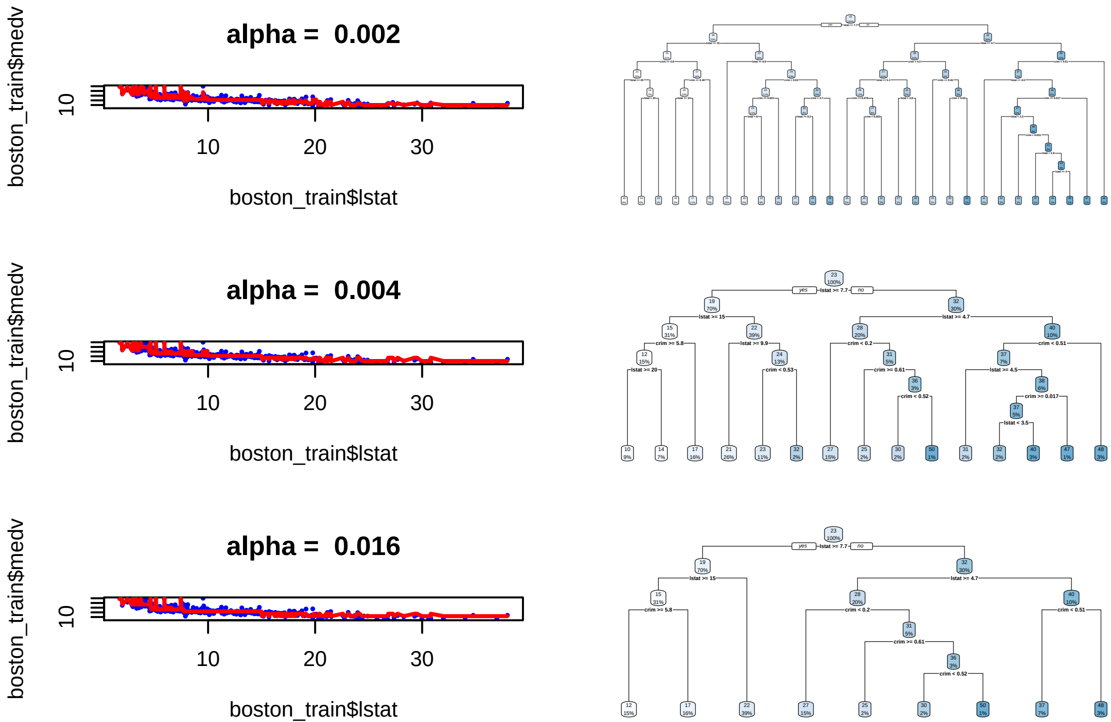
```r
(bestcp = cptable[ which.min(cptable[,"xerror"]), "CP" ])   # this is the optimal cp parameter
```

```
## [1] 0.004462849
```

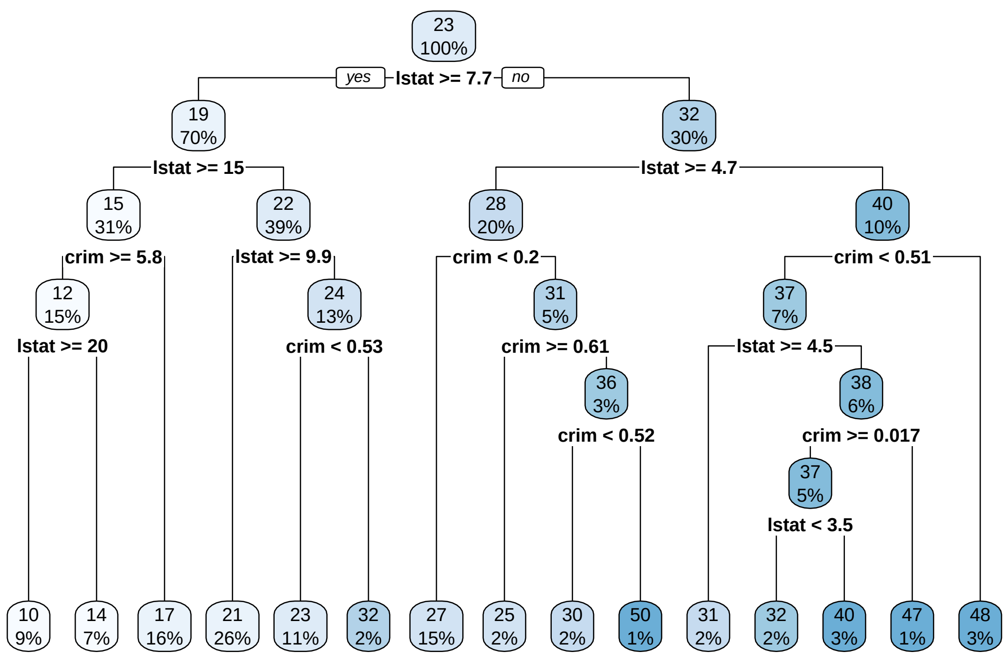
```r
plotcp(fit6) # plot results
```



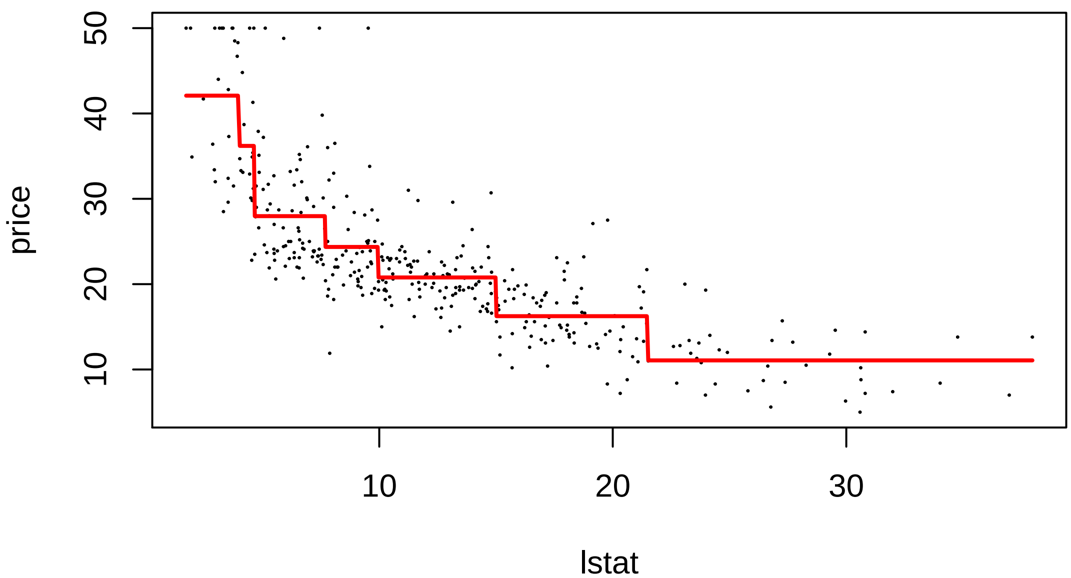

```
# show fit from some trees
cpvec = c(bestcp / 2, bestcp,.0157)

par(mfrow=c(3,2))
for(i in 1:3) {
  plot(boston_train$lstat,boston_train$medv,pch=16,col='blue',cex=.5)
  ptree = prune(fit6,cp=cpvec[i])
  pfit = predict(ptree)
  lines(boston_train$lstat[oo],pfit[oo],col='red',lwd=2)
  title(paste('alpha = ',round(cpvec[i],3)))
  rpart.plot(ptree)
}
```




```r
par(mfrow=c(1,1))
fit6B = prune(fit6,cp=bestcp)
rpart.plot(fit6B)
```




```r
plot(boston_train$lstat, boston_train$medv, xlab = "lstat", ylab = "price", pch = 1, cex = 0.1)
lines(sort(boston_train$lstat),predict(fit4B)[oo],col="red",lwd=2,cex.lab=2)
```




```
# error
cat('RMSE is ',sqrt(mean((boston_test$medv -predict(fit6B, boston_test))^2)),'\n')
```

```
## RMSE is  5.504513
```

Adding All Variables: Regression Trees

```
fit7 = rpart(medv~.,        #Formula
             data=boston_train,#Data
              control=rpart.control(minsplit=5,#the minimum number of observations that must exist in a
                                    cp=0.0001, #complexity, the lower, the larger the tree is
                                    xval=10    #number of cross validations
                                    ))
```

References

Data Description:

https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

Package manuals:

https://cran.r-project.org/web/packages/kknn/kknn.pdf https://cran.r-project.org/web/packages/rpart/rpart.pdf