Logistic Regression & Tree Classification

Group 4 - Cullen McNamee, Josep Nueno, and WanQi Tay

8/25/2018

```
suppressWarnings(library(dplyr))
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
suppressWarnings(library(gains))
suppressWarnings(library(AUC))
```

```
## AUC 0.3.0
## Type AUCNews() to see the change log and ?AUC to get an overview.
```

```
suppressWarnings(library(rpart))
suppressWarnings(library(rpart.plot))
```

Data Preparation

```
melbourne <- read.csv('melbourne_edited.csv')
```

```
#Identified categorical variables by sorting by the number of unique values.

uniqueCounts <- apply(melbourne, 2, unique)
sort(unlist(lapply(uniqueCounts, length)))
```

```
##          Type    Regionname        Method         Rooms      Bathroom
##             3             8             9            12            12
##      Bedroom2           Car   CouncilArea          Date     YearBuilt
##            16            16            33            78           161
##      Postcode      Distance Propertycount        Suburb       SellerG
##           211           215           342           350           388
##  BuildingArea      Landsize         Price    Longtitude     Lattitude
##           741          1685          2872          5588         13403
##       Address
##         34006
```

```
#Exclude irrelevant or distinct fields (i.e. address and seller)

fieldsToExclude <- c('Address', 'SellerG', 'Postcode', 'Longtitude',
                     'Lattitude', 'Date', 'Suburb', 'CouncilArea')

melbourne <- melbourne[,!colnames(melbourne) %in% fieldsToExclude]
```

```
#Remove NAs in Price, since this will be our Response variable.

melbourne <- melbourne %>% filter(Price != 'NA')

#Remove variables with NA in other important fields.

melbourne <- melbourne %>% filter(is.na(Bedroom2) == FALSE,
                                  is.na(Bathroom) == FALSE,
                                  is.na(Car) == FALSE,
                                  is.na(Landsize) == FALSE,
                                  is.na(BuildingArea) == FALSE,
                                  is.na(YearBuilt) == FALSE)

#This leaves us with 8895 observations remaining

nrow(melbourne)
```

```
## [1] 8895
```

```
#Identify top quartile price, for use in logistic regression.
#Note: Top quartile here is an arbitrary cutoff point for classification.

summary(melbourne$Price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  131000  640500  900000 1092524 1345000 9000000
```

```
#Code top-quartile prices as 1's and others as 0s.

melbourne$Price <- 1*(melbourne$Price > 1345000)
```

Logistic Regression

```
#Split data set into numeric and categorical variables.

numericVariables <- c('Landsize', 'Rooms', 'Bathroom', 'Bedroom2',
                      'Car', 'Distance', 'Propertycount', 'BuildingArea', 'YearBuilt')

categoricalVariables <- c('Price', 'Type', 'Method', 'Regionname')

#Note: Price was transforming to a categorical variable: {top-quartile, not top-quartile}

melbourne[,categoricalVariables] <- lapply(melbourne[,categoricalVariables], factor)
melbourne[,numericVariables] <- apply(melbourne[,numericVariables], 2, as.numeric)
```

```
#Randomly split data set into training and holdout samples.

set.seed(10101)
randomIndex <- sample(1:nrow(melbourne), size = 0.632*nrow(melbourne))
melbourneTrain <- melbourne[randomIndex, ]
melbourneHoldout <- melbourne[-randomIndex, ]
```

```
#Run logistic regression on the training sample, using all variables.

logisticAll <- glm(Price ~ ., data = melbourneTrain, family=binomial(link=logit))
```

```r
#Use step to minimize the AIC of the model, using direction both.

stepAICModel <- step(logisticAll, direction = 'both', trace = FALSE)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
print(stepAICModel$call)
```

```
## glm(formula = Price ~ Rooms + Type + Distance + Bathroom + Car +
##     Landsize + BuildingArea + YearBuilt + Regionname + Propertycount,
##     family = binomial(link = logit), data = melbourneTrain)
```

```r
logisticStepped <- stepAICModel
summary(stepAICModel)
```

```
##
## Call:
## glm(formula = Price ~ Rooms + Type + Distance + Bathroom + Car +
##     Landsize + BuildingArea + YearBuilt + Regionname + Propertycount,
##     family = binomial(link = logit), data = melbourneTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0193  -0.3722  -0.1158   0.0000   3.7617
##
## Coefficients:
##                                         Estimate Std. Error z value
## (Intercept)                            2.299e+01  3.078e+00   7.470
## Rooms                                  6.901e-01  7.976e-02   8.653
## Typet                                 -1.553e+00  2.104e-01  -7.384
## Typeu                                 -4.357e+00  3.895e-01 -11.187
## Distance                              -3.187e-01  1.676e-02 -19.017
## Bathroom                               6.155e-01  8.959e-02   6.870
## Car                                    2.266e-01  5.320e-02   4.260
```

```
## Landsize                               1.895e-04  4.148e-05   4.569
## BuildingArea                           8.419e-03  8.457e-04   9.954
## YearBuilt                             -1.297e-02  1.614e-03  -8.036
## RegionnameEastern Victoria            -1.085e+01  6.030e+02  -0.018
## RegionnameNorthern Metropolitan       -2.109e+00  1.898e-01 -11.117
## RegionnameNorthern Victoria           -1.276e+01  4.499e+02  -0.028
## RegionnameSouth-Eastern Metropolitan   1.006e+00  3.107e-01   3.237
## RegionnameSouthern Metropolitan        1.068e+00  1.563e-01   6.831
## RegionnameWestern Metropolitan        -2.290e+00  1.814e-01 -12.626
## RegionnameWestern Victoria            -1.583e+01  7.652e+02  -0.021
## Propertycount                          2.955e-05  1.309e-05   2.258
##                                       Pr(>|z|)
## (Intercept)                           8.05e-14 ***
## Rooms                                  < 2e-16 ***
## Typet                                 1.53e-13 ***
## Typeu                                  < 2e-16 ***
## Distance                               < 2e-16 ***
## Bathroom                              6.41e-12 ***
## Car                                   2.04e-05 ***
## Landsize                              4.90e-06 ***
## BuildingArea                           < 2e-16 ***
## YearBuilt                             9.27e-16 ***
## RegionnameEastern Victoria             0.98564
## RegionnameNorthern Metropolitan        < 2e-16 ***
## RegionnameNorthern Victoria            0.97737
## RegionnameSouth-Eastern Metropolitan   0.00121 **
## RegionnameSouthern Metropolitan       8.45e-12 ***
## RegionnameWestern Metropolitan         < 2e-16 ***
## RegionnameWestern Victoria             0.98350
## Propertycount                          0.02395 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6270.2  on 5620  degrees of freedom
## Residual deviance: 2833.5  on 5603  degrees of freedom
## AIC: 2869.5
##
## Number of Fisher Scoring iterations: 16
```

```r
#Using probability cutoff of 0.5, predict 1s or 0s for Price.

fittedValues <- logisticStepped$fitted.values
fittedValues[fittedValues>=0.5]=1
fittedValues[fittedValues<0.5]=0
```

Confusion Matrix of Train

```r
table(melbourneTrain$Price,fittedValues)
```

```
##    fittedValues
##        0    1
##   0 4007  232
##   1  382 1000
```

```r
round(prop.table(table(melbourneTrain$Price,fittedValues),1),2)
```

```
##    fittedValues
##        0    1
##   0 0.95 0.05
##   1 0.28 0.72
```

```r
#Run predictions on holodut sample.
predictions.holdout=predict(logisticStepped, newdata=melbourneHoldout[,-3],type="response")
predictedValues <- predictions.holdout
predictedValues[predictedValues>=0.5]=1
predictedValues[predictedValues<0.5]=0
```

Confusion Matrix of Holdout

```r
table(melbourneHoldout$Price,predictedValues)
```

```
##    predictedValues
##        0    1
##   0 2295  140
##   1  235  604
```

```r
round(prop.table(table(melbourneHoldout$Price,predictedValues),1),2)
```

```
##    predictedValues
##        0    1
##   0 0.94 0.06
##   1 0.28 0.72
```

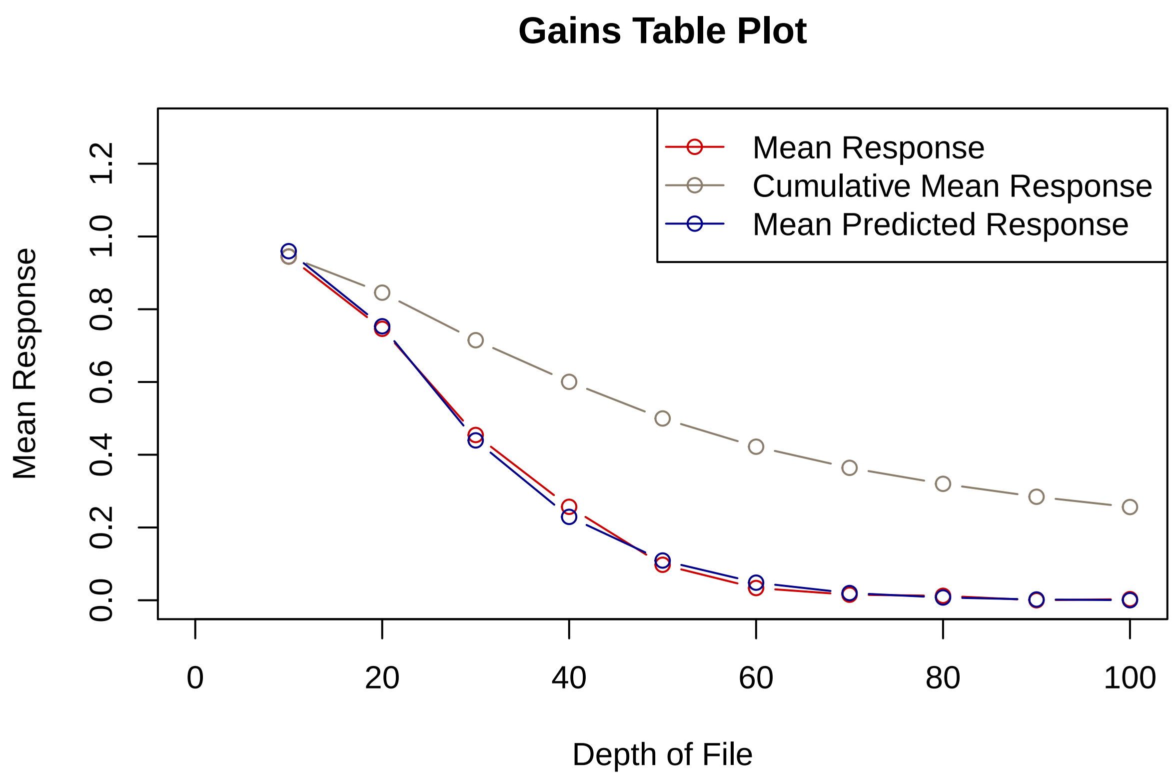
Gains Table

```r
gains(as.numeric(melbourneHoldout$Price)-1,predictions.holdout,10)
```

```
## Depth                            Cume   Cume Pct                     Mean
##  of           Cume     Mean      Mean   of Total    Lift   Cume     Model
## File     N      N      Resp      Resp      Resp     Index  Lift     Score
## -------------------------------------------------------------------------
##   10   327    327      0.94      0.94      36.8%     369    369      0.96
##   20   327    654      0.75      0.85      65.9%     291    330      0.75
##   30   328    982      0.45      0.71      83.7%     177    279      0.44
##   40   327   1309      0.26      0.60      93.7%     100    234      0.23
##   50   328   1637      0.10      0.50      97.5%      38    195      0.11
##   60   327   1964      0.03      0.42      98.8%      13    165      0.05
##   70   327   2291      0.02      0.36      99.4%       6    142      0.02
##   80   328   2619      0.01      0.32      99.9%       5    125      0.01
##   90   327   2946      0.00      0.28      99.9%       0    111      0.00
##  100   328   3274      0.00      0.26     100.0%       1    100      0.00
```

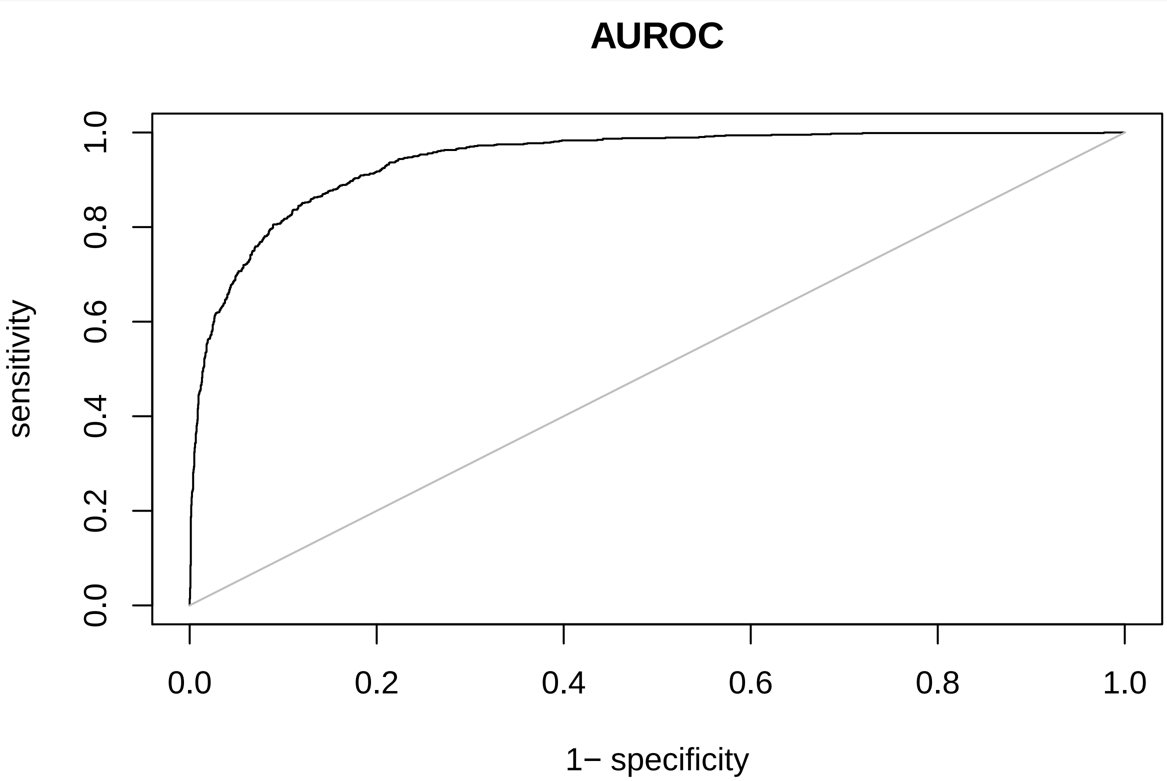
Gains Plot

```r
plot(gains(as.numeric(melbourneHoldout$Price)-1,
           predictions.holdout,10), ylim = c(0, 1.3),
           xlim = c(0, 100))
```



AUC Plot

```
plot(roc(predictions.holdout, melbourneHoldout$Price), main = "AUROC")
```



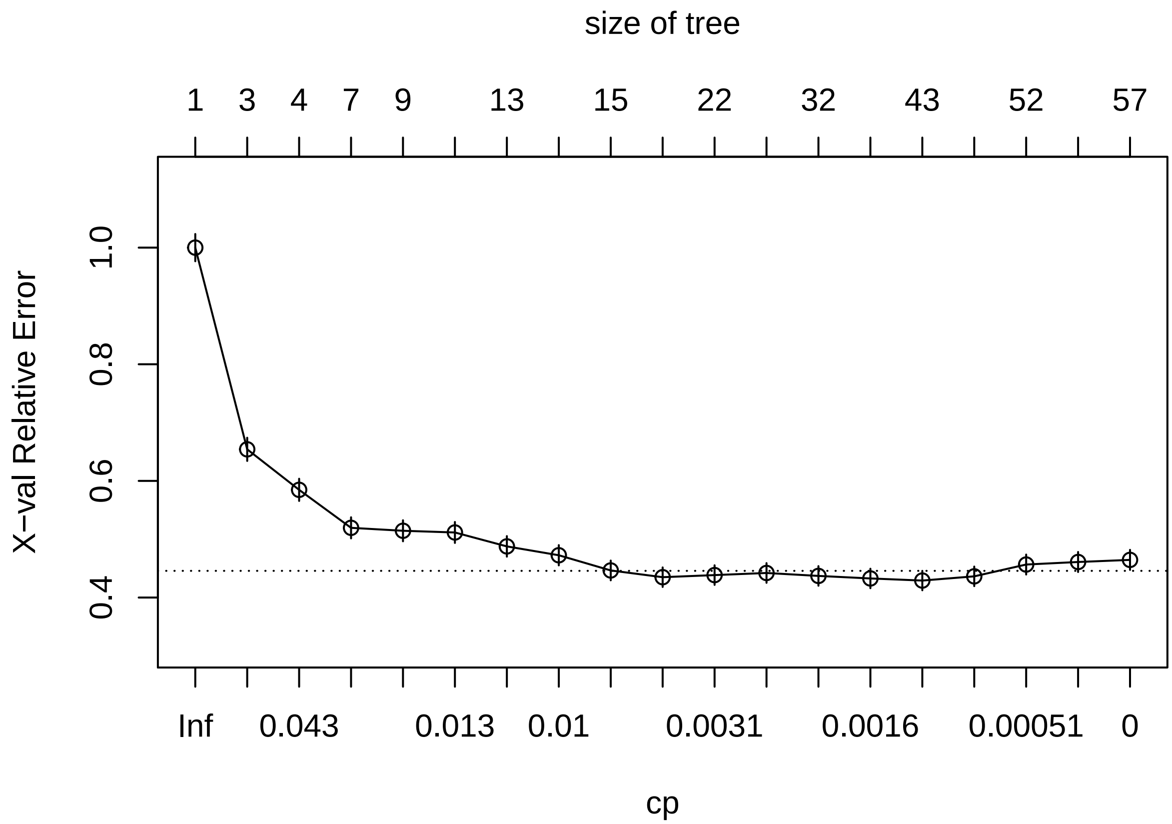
Tree Classification Model

```
#Fit tree model using rpart.
#Minimum split of 30.

classificationTrain <-
  rpart(Price ~.,data = melbourneTrain,
        control=rpart.control(cp=0,minsplit=30,xval=10, maxsurrogate=0))
printcp(classificationTrain)
```

```
##
## Classification tree:
## rpart(formula = Price ~ ., data = melbourneTrain, control = rpart.control(cp = 0,
##     minsplit = 30, xval = 10, maxsurrogate = 0))
##
## Variables actually used in tree construction:
##  [1] Bathroom      Bedroom2      BuildingArea  Car           Distance
##  [6] Landsize      Propertycount Regionname    Type          YearBuilt
##
## Root node error: 1382/5621 = 0.24586
##
## n= 5621
##
##             CP nsplit rel error  xerror     xstd
## 1  0.17329957      0   1.00000 1.00000 0.023360
## 2  0.07525326      2   0.65340 0.65412 0.019930
## 3  0.02484322      3   0.57815 0.58466 0.019033
## 4  0.01447178      6   0.50362 0.51954 0.018108
## 5  0.01338640      8   0.47467 0.51447 0.018033
## 6  0.01266281     10   0.44790 0.51158 0.017989
## 7  0.01085384     12   0.42258 0.48770 0.017623
## 8  0.01013025     13   0.41172 0.47250 0.017383
## 9  0.00361795     14   0.40159 0.44645 0.016958
## 10 0.00325615     15   0.39797 0.43488 0.016764
## 11 0.00289436     21   0.37410 0.43849 0.016825
## 12 0.00180897     25   0.36252 0.44211 0.016886
## 13 0.00168837     31   0.35166 0.43705 0.016801
## 14 0.00144718     34   0.34660 0.43271 0.016727
## 15 0.00108538     42   0.33213 0.42909 0.016665
## 16 0.00072359     48   0.32562 0.43632 0.016788
## 17 0.00036179     51   0.32344 0.45658 0.017126
## 18 0.00024120     53   0.32272 0.46093 0.017197
## 19 0.00000000     56   0.32200 0.46454 0.017255
```

```
plotcp(classificationTrain, minline = TRUE)
```




```r
#Find minimum error of CP.

cpResults <- printcp(classificationTrain)
```

```
##
## Classification tree:
## rpart(formula = Price ~ ., data = melbourneTrain, control = rpart.control(cp = 0,
##     minsplit = 30, xval = 10, maxsurrogate = 0))
##
## Variables actually used in tree construction:
##  [1] Bathroom      Bedroom2      BuildingArea  Car           Distance
##  [6] Landsize      Propertycount Regionname    Type          YearBuilt
##
## Root node error: 1382/5621 = 0.24586
##
## n= 5621
##
##            CP nsplit rel error  xerror     xstd
## 1  0.17329957      0   1.00000 1.00000 0.023360
## 2  0.07525326      2   0.65340 0.65412 0.019930
## 3  0.02484322      3   0.57815 0.58466 0.019033
## 4  0.01447178      6   0.50362 0.51954 0.018108
## 5  0.01338640      8   0.47467 0.51447 0.018033
## 6  0.01266281     10   0.44790 0.51158 0.017989
## 7  0.01085384     12   0.42258 0.48770 0.017623
## 8  0.01013025     13   0.41172 0.47250 0.017383
## 9  0.00361795     14   0.40159 0.44645 0.016958
## 10 0.00325615     15   0.39797 0.43488 0.016764
```

```
## 11 0.00289436     21   0.37410 0.43849 0.016825
## 12 0.00180897     25   0.36252 0.44211 0.016886
## 13 0.00168837     31   0.35166 0.43705 0.016801
## 14 0.00144718     34   0.34660 0.43271 0.016727
## 15 0.00108538     42   0.33213 0.42909 0.016665
## 16 0.00072359     48   0.32562 0.43632 0.016788
## 17 0.00036179     51   0.32344 0.45658 0.017126
## 18 0.00024120     53   0.32272 0.46093 0.017197
## 19 0.00000000     56   0.32200 0.46454 0.017255
```

```
minErrorCP <- cpResults[as.numeric(which.min(cpResults[,4])),1]
print(minErrorCP)
```

```
## [1] 0.001085384
```

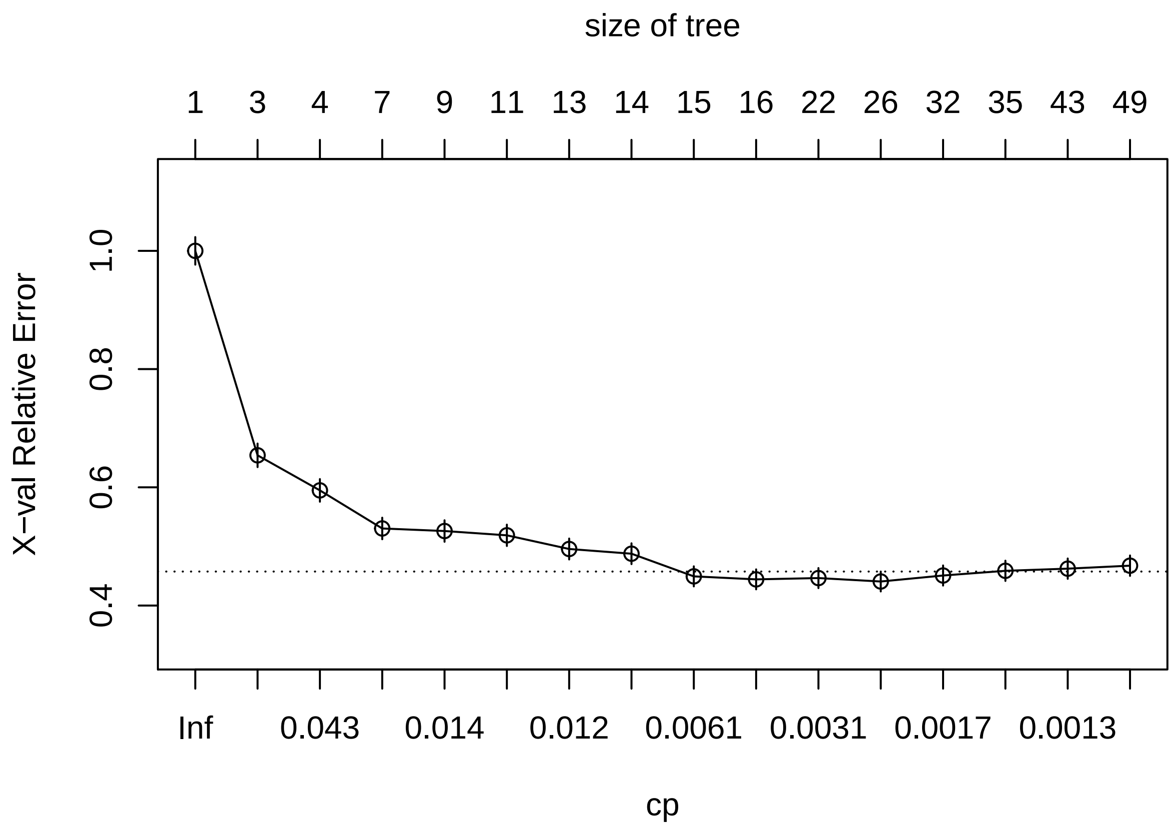
```
#Prune initial tree using above cp.

classificationTrainPruned <- rpart(Price ~., data = melbourneTrain,
                                   control=rpart.control(cp=minErrorCP,minsplit=30,
                                                         xval=10, maxsurrogate=0))
printcp(classificationTrainPruned)
```

```
##
## Classification tree:
## rpart(formula = Price ~ ., data = melbourneTrain, control = rpart.control(cp = minErrorCP,
##     minsplit = 30, xval = 10, maxsurrogate = 0))
##
## Variables actually used in tree construction:
##  [1] Bathroom      Bedroom2      BuildingArea  Car           Distance
##  [6] Landsize      Propertycount Regionname    Type          YearBuilt
##
## Root node error: 1382/5621 = 0.24586
##
## n= 5621
##
##           CP nsplit rel error  xerror     xstd
## 1  0.1732996      0   1.00000 1.00000 0.023360
## 2  0.0752533      2   0.65340 0.65412 0.019930
## 3  0.0248432      3   0.57815 0.59479 0.019169
## 4  0.0144718      6   0.50362 0.53039 0.018268
## 5  0.0133864      8   0.47467 0.52605 0.018205
## 6  0.0126628     10   0.44790 0.51881 0.018098
## 7  0.0108538     12   0.42258 0.49566 0.017747
## 8  0.0101302     13   0.41172 0.48770 0.017623
## 9  0.0036179     14   0.40159 0.44935 0.017007
## 10 0.0032562     15   0.39797 0.44428 0.016922
## 11 0.0028944     21   0.37410 0.44645 0.016958
## 12 0.0018090     25   0.36252 0.44067 0.016862
## 13 0.0016884     31   0.35166 0.45080 0.017030
## 14 0.0014472     34   0.34660 0.45876 0.017161
## 15 0.0010854     42   0.33213 0.46237 0.017220
## 16 0.0010854     48   0.32562 0.46744 0.017302
```

CP Plot

```r
plotcp(classificationTrainPruned, minline = TRUE)
```



Confusion Matrix of Un-Pruned Tree (Train)

```r
table(melbourneTrain$Price,predict(classificationTrain,type="class"))
```

```
##
##         0    1
##   0 4073  166
##   1  279 1103
```

```r
round(prop.table(table(melbourneTrain$Price,predict(classificationTrain,
                                                    type="class")),1),2)
```

```
##
##         0    1
##   0 0.96 0.04
##   1 0.20 0.80
```

Confusion Matrix of Pruned Tree (Train)

```r
table(melbourneTrain$Price,predict(classificationTrainPruned,type="class"))
```

```
##
##         0    1
##   0 4079  160
##   1  290 1092
```

```r
round(prop.table(table(melbourneTrain$Price,predict(classificationTrainPruned,
                                                    type="class")),1),2)
```

```
##
##         0    1
##   0 0.96 0.04
##   1 0.21 0.79
```

Confusion Matrix of Pruned Tree (Holdout)

```r
table(melbourneHoldout[,3],
      predict(classificationTrainPruned,newdata=melbourneHoldout[,-3],type="class"))
```

```
##
##         0    1
##   0 2268  167
##   1  209  630
```

```r
round(prop.table(table(melbourneHoldout[,3],
      predict(classificationTrainPruned,newdata=melbourneHoldout[,-3],
              type="class")),1),2)
```

```
##
##         0    1
##   0 0.93 0.07
##   1 0.25 0.75
```

Tree Plot

```r
#For this analysis CouncilArea is excluded.
#Min split of 500 is used.

melbourneTrainWithoutCouncil <- melbourneTrain[, !colnames(melbourneTrain) %in% 'CouncilArea']

classificationTrainPruned_ExMelbourne <-
  rpart(Price ~., data = melbourneTrainWithoutCouncil,
        control=rpart.control(cp=minErrorCP,minsplit=500, xval=10, maxsurrogate=0))

rpart.plot(classificationTrainPruned_ExMelbourne, type = 5)
```