

HOMEWORK 2

a) Dichotomize the variable “Price” at its 75th percentile and call this new variable “HighPrice”. Please note that you will not replace the “Price” variable but create a new variable “HighPrice”.

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains R code for a logistic regression model and a function to dichotomize the 'Price' variable.
- Environment:** Lists objects in the global environment, including 'obj', 'obj1', 'rfit', 'yhat', 'yhat.class', and 'yhat1'.
- Console:** Shows the output of the R code, including the results of the 'HighPrice' function and the 'sum' function.
- Documentation:** Displays the R documentation for 'Arithmetic Operators'.

```
61 # a). Dichotomize the variable "Price" at its 75th percentile and call this new
62 # variable "HighPrice". Please note that you will not replace the "Price"
63 # variable but create a new variable "HighPrice".
64
65 par(mfrow=c(1,1))
66 yhat1 = predict(obj, newdata = dat.test, type='response')
67 hist(yhat1)
68 dat.train = dat[id.train,]
69 dat.test = dat[id.test,]
70
71 HighPrice = function(yhat1, cutoff=.75) {
72   out = rep(0, length(yhat1))
73   out[yhat1 > cutoff] = 1
74   out
75 }
76
77 yhat.class = HighPrice(yhat1)
78
79 sum(yhat.class != dat.test$Price)/length(id.test)
```

Console output:

```
+ out
+ ]
> yhat.class = HighPrice(yhat1)
> sum(yhat.class != dat.test$Price)/length(id.test)
[1] 1
> sum(yhat.class != dat.test$HighPrice)/length(id.test)
[1] 0
> sum(yhat.class != dat.test$Price)/length(id.test)
[1] 1
>
```

Environment:

Object	Type	Value
obj	List of 12	
obj1	List of 14	
rfit	List of 15	
yhat	Named num [1:392]	19913 13689 21837 22110 1...
yhat.class	num [1:392]	1 1 1 1 1 1 1 1 1 ...
yhat1	Named num [1:392]	19913 13689 21837 22110 1...

Documentation: Arithmetic Operators

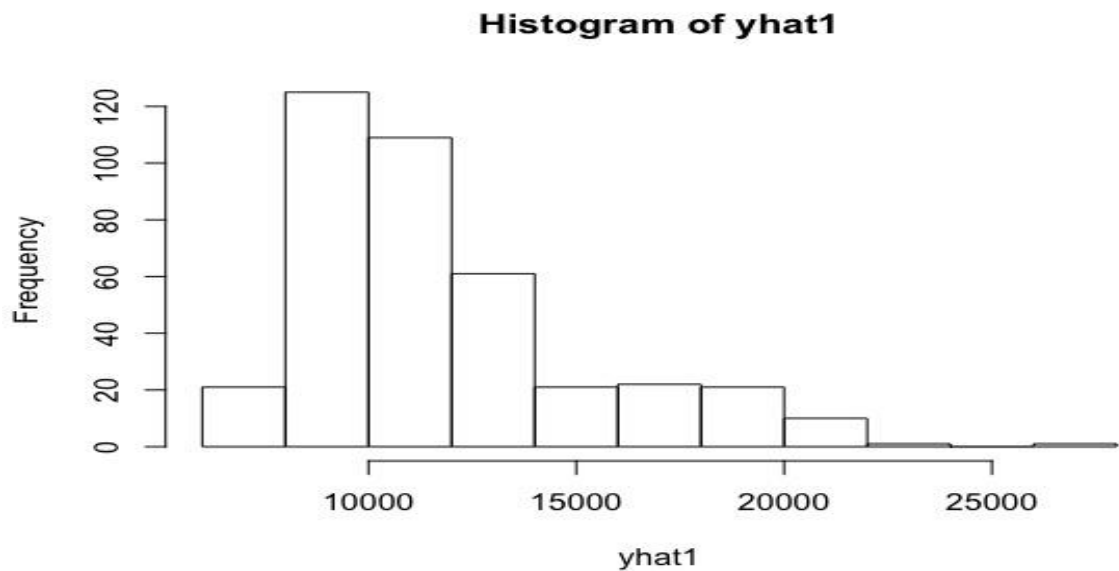
Description: These unary and binary operators perform arithmetic on numeric or complex vectors (or objects which can be coerced to them).

Usage:

```
+ x
- x
x + y
x - y
x * y
x / y
x ^ y
x %*% y
x %/% y
```

Arguments:

- x, y: numeric or complex vectors or objects which can be coerced to such, or other objects for which methods have been written.



b) Partition the data into training (60%) and validation (40%).

```

1 rm(list=ls()); gc()
2 setwd("~/Users/architjajoo/Desktop/d/CSUF classes/Semester 2/ISDS 574/HW2")
3
4 dat = read.csv('toyota_clean2.csv', stringsAsFactors=T, head=T)
5 View(dat)
6
7 # b). - Partition the data into training (60%) and validation (40%).
8 set.seed(1)
9 id.train = sample(1:nrow(dat), nrow(dat)*.6)
10 id.test = setdiff(1:nrow(dat), id.train)
11
12 obj = lm(Price ~ ., data = dat[id.train, ])
13 summary(obj)
14

```

Console Output:

```

Mistlamps      220.1727  170.4118   1.292 0.196892
Sport_Model    171.4837  136.6224   1.255 0.209946
Backseat_Divider -179.2190  205.4264  -0.872 0.383354
Metallic_Rim    226.4355  145.7760   1.553 0.120918
Tow_Bar        111.1975  126.8950   0.876 0.381248
other.color    -68.8499  203.0017  -0.339 0.734618
red            -395.9147  172.2523  -2.298 0.021906 *
green          -252.1111  182.0006  -1.385 0.166540
blue           -288.8612  172.4742  -1.675 0.094534 .
black           -1.5517  186.0160  -0.008 0.993347
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

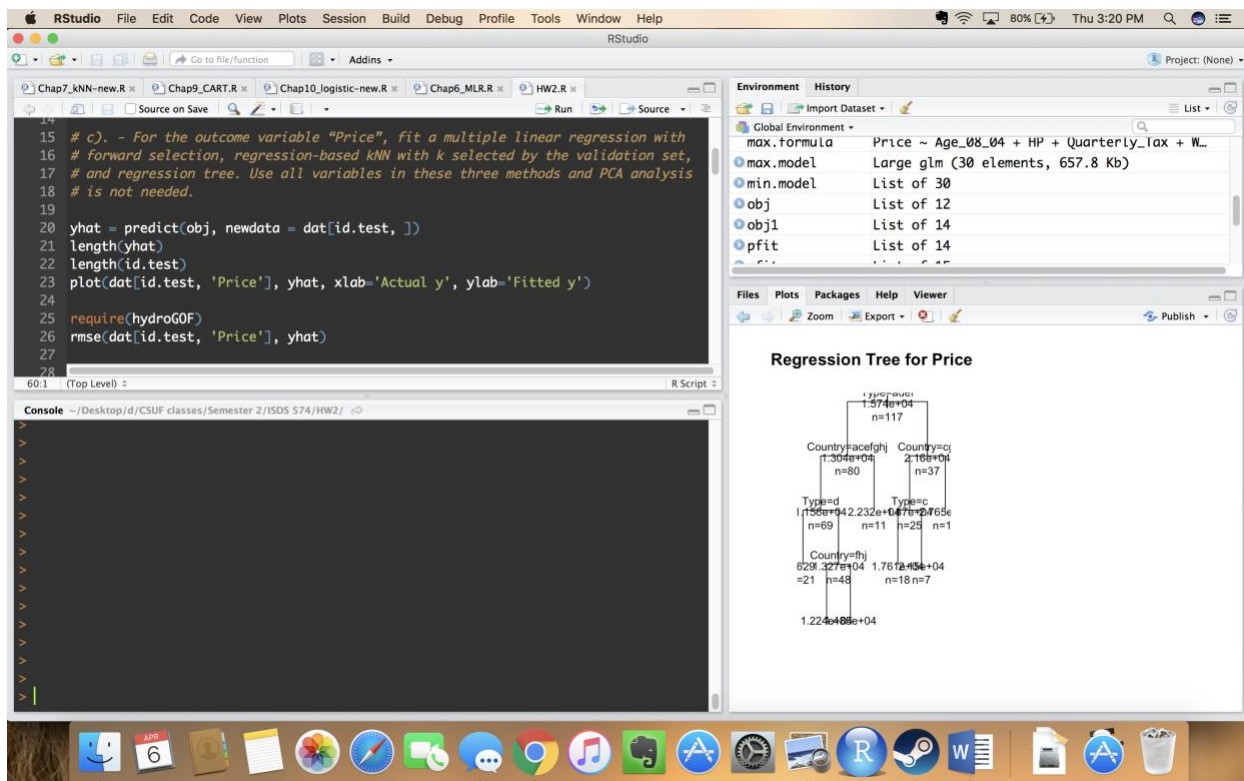
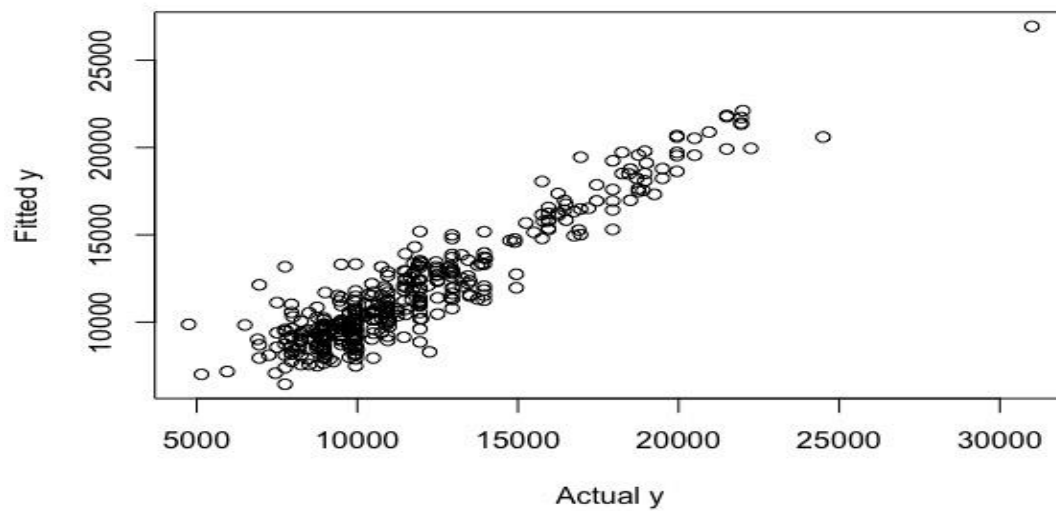
Residual standard error: 1245 on 556 degrees of freedom
Multiple R-squared:  0.8828,    Adjusted R-squared:  0.8767
F-statistic: 144.4 on 29 and 556 DF,  p-value: < 2.2e-16

```

Environment:

Variable	Value
dat	978 obs. of 30 variables
id.test	int [1:392] 2 3 6 8 9 10 11 15 16 17 ...
id.train	int [1:586] 260 364 560 886 197 875 919 642...
obj	List of 12

c) For the outcome variable “Price”, fit a multiple linear regression with forward selection, regression-based kNN with k selected by the validation set, and regression tree. Use all variables in these three methods and PCA analysis is not needed.



Forward selection:

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for fitting a multiple linear regression model using forward selection, regression-based k-NN, and regression tree. The code includes comments and functions like `predict()`, `length()`, `plot()`, `require(hydroGOF)`, `rmse()`, `library(MASS)`, and `stepAIC()`.
- Console:** Displays the output of the code execution, including a table of model coefficients, significance codes, and summary statistics like Residual standard error, Multiple R-squared, Adjusted R-squared, and F-statistic.
- Environment:** Shows the global environment with variables `dat` (978 obs. of 30 variables), `id.test` (int [1:392]), `id.train` (int [1:586]), `obj` (List of 12), and `obj1` (List of 14).
- Help Pane:** Displays the documentation for Arithmetic Operators, including a description and usage examples.

Console Output:

```

Metallic.Kim      226.4355    145.7760     1.553 0.120918
Tow_Bar           111.1975    126.8950     0.876 0.381248
other.color       -68.8499    203.0017    -0.339 0.734618
red              -395.9147    172.2523    -2.298 0.021906 *
green            -252.1111    182.0006    -1.385 0.166540
blue             -288.8612    172.4742    -1.675 0.094534 .
black            -1.5517     186.0160    -0.008 0.993347
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1245 on 556 degrees of freedom
Multiple R-squared:  0.8828,    Adjusted R-squared:  0.8767 
F-statistic: 144.4 on 29 and 556 DF,  p-value: < 2.2e-16

```

kNN:

The screenshot shows the RStudio interface with a script editor, console, and environment pane. The script defines a `knn.reg` function that takes training and testing data, a number of neighbors `k`, and a probability `prob`. It calculates the predicted class for each test point and returns the optimal `k` value. The console shows the execution of the function with `k=18` and `pe.min=158`.

```
# knn
install.packages("FNN")
library("FNN")

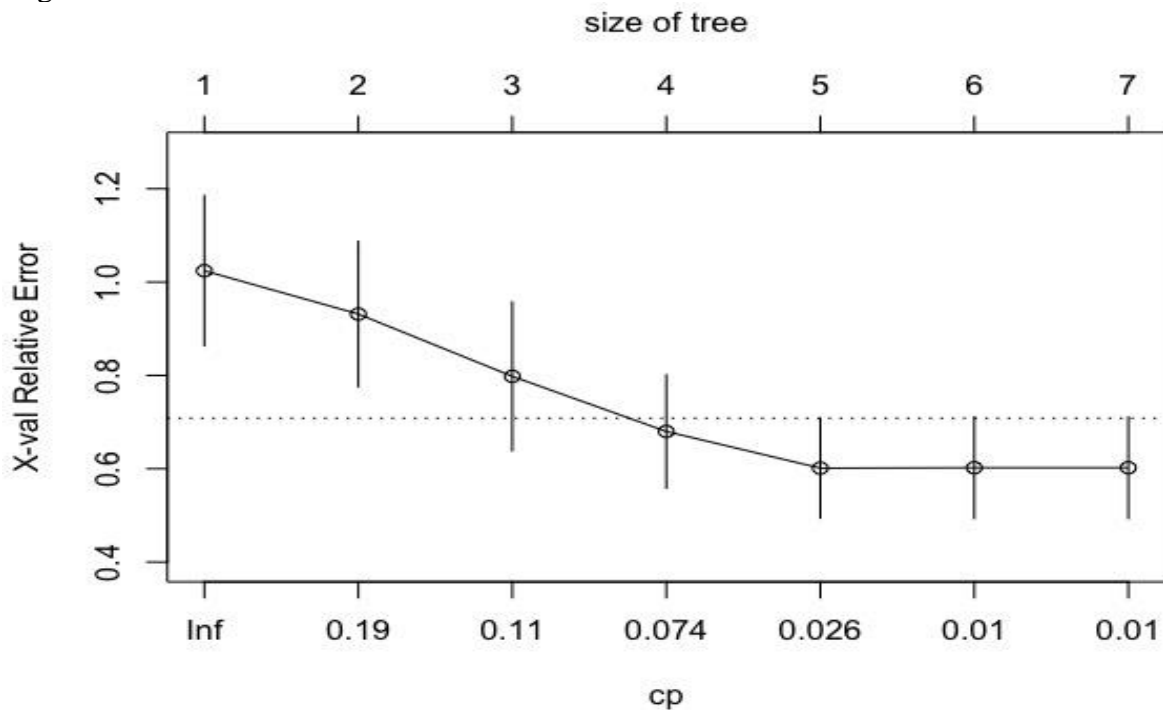
knn.reg = function(train, test, y.train, y.test, k.max = 20) {
  pe = rep(NA, k.max)
  for (ii in 1:k.max) {
    y.hat = knn(train, test, y.train, k = ii, prob=F)
    pe[ii] = sum(y.hat != y.test)
  }
  list(k.optimal = which.min(pe), pe.min = min(pe))
}

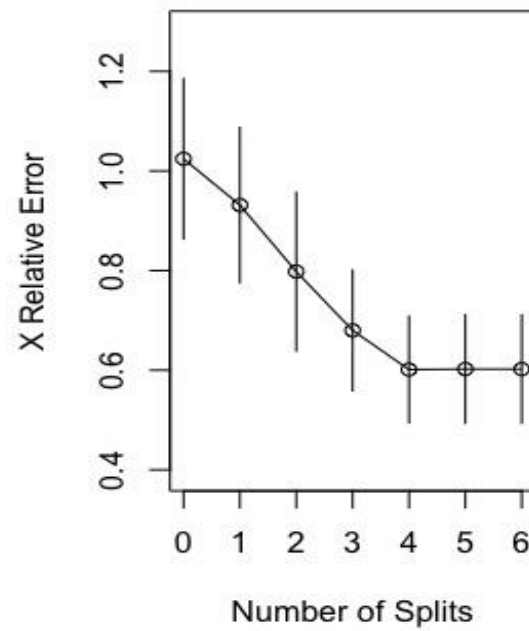
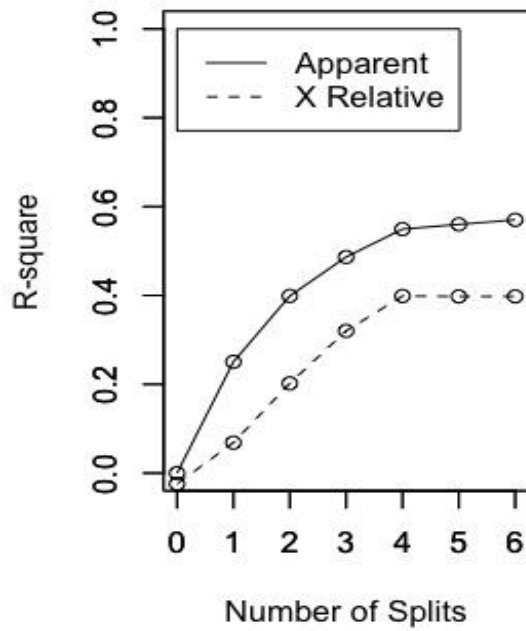
knn.reg(dat[id.train,1:2], dat[id.test,1:2], dat[id.train,3], dat[id.test,3])
```

```
> knn.reg(train, test, y.train, y.test, k.max = 20) {
+   pe = rep(NA, k.max)
+   for (ii in 1:k.max) {
+     y.hat = knn(train, test, y.train, k = ii, prob=F)
+     pe[ii] = sum(y.hat != y.test)
+   }
+   list(k.optimal = which.min(pe), pe.min = min(pe))
+ }
> knn.reg(dat[id.train,1:2], dat[id.test,1:2], dat[id.train,3], dat[id.test,3])
$k.optimal
[1] 18

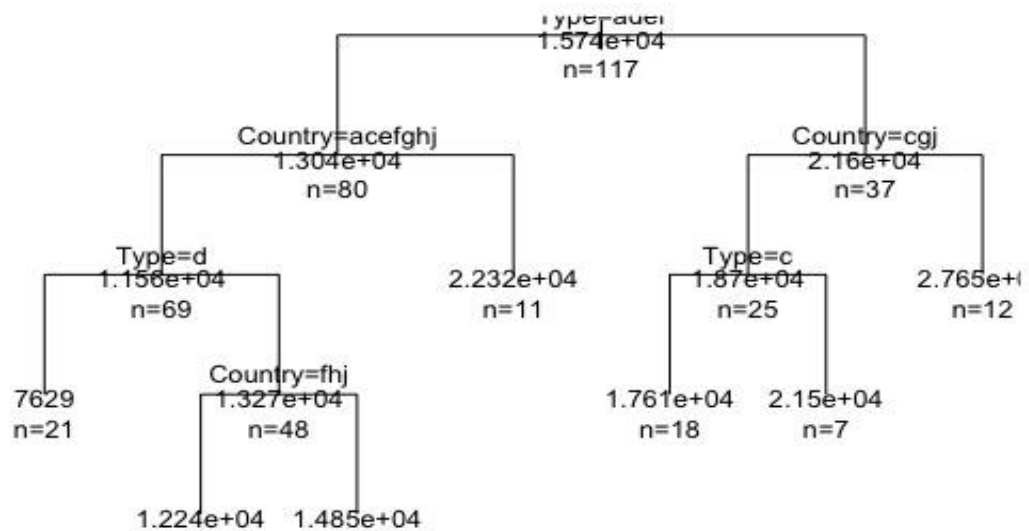
$pe.min
[1] 158
```

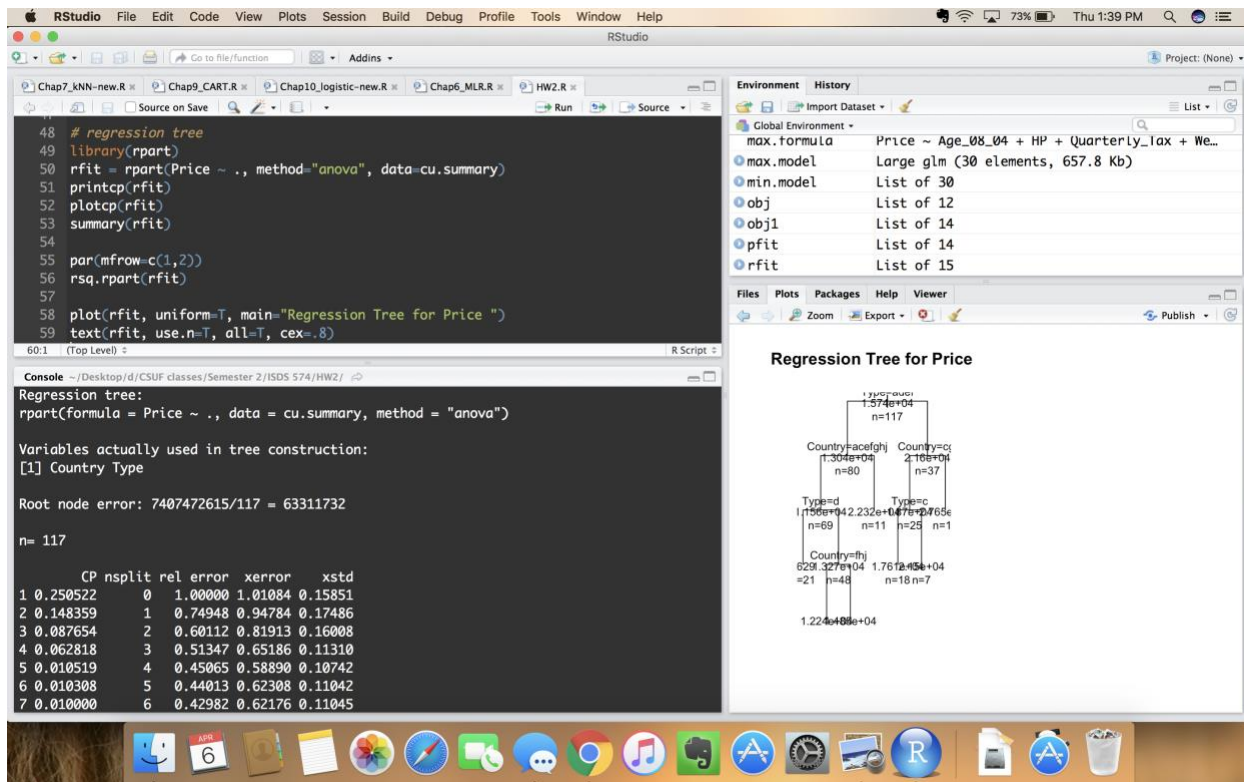
Regression tree:





Regression Tree for Price





d) Compare the above three models obtained and pick up the best one. Explain why you think it is the best one. Note that I already took out the “grey” as the reference group for color. You don’t have to do any data cleaning in this homework.

Multiple Linear Regression through forward selection is the best model because it gives lowest RMSE.

e) For the outcome variable “HighPrice”, fit a logistic regression with forward regression, classification-based kNN with k selected by the validation set, and classification tree.

Forward:

```

81 #forward
82 min.model = glm(HighPrice ~ 1, data = dat.train, family = 'gaussian')
83 max.model = glm(HighPrice ~ ., data = dat.train, family = 'gaussian')
84 max.formula = formula(max.model)
85 HPobj = step(min.model, direction='forward', scope=max.formula)
86 summary(HPobj)
87

```

Console Output:

Weight	10.8333	1.6395	6.608	8.99e-11	***
Powered_Windows	374.1826	134.8980	2.774	0.005722	**
CC	-2.5354	0.4569	-5.549	4.40e-08	***
Quarterly_Tax	11.0614	2.2608	4.893	1.30e-06	***
Boardcomputer	-613.7306	151.1003	-4.062	5.55e-05	***
BOVAG_Guarantee	717.7202	190.5900	3.766	0.000183	***
Guarantee_Period	73.2172	17.6746	4.142	3.96e-05	***
Mistlamps	252.2017	153.9675	1.638	0.101970	
Mfr_Guarantee	235.5086	111.2600	2.117	0.034715	*
Automatic	480.2772	235.0281	2.043	0.041464	*
red	-264.5917	131.7811	-2.008	0.045135	*
Metallic_Rim	216.5301	142.4647	1.520	0.129095	
blue	-216.3206	143.5129	-1.507	0.132282	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1540903)

Null deviance: 7356639915 on 585 degrees of freedom
Residual deviance: 876773938 on 569 degrees of freedom
AIC: 10031

Number of Fisher Scoring iterations: 2

kNN:

```

88 #knn
89 require(class)
90 knNobj = knn(dat[id.train,1:2], dat[id.test,1:2], dat[id.train,3], k = 3, prob=TRUE)
91 class(knNobj)
92
93
94 knn.bestKHP = function(train, test, y.train, y.test, k.max = 20) {
95   pe = rep(NA, k.max)
96   for (ii in 1:k.max) {
97     y.hat = knn(train, test, y.train, k = ii, prob=F)
98     pe[ii] = sum(y.hat != y.test)
99   }
100   list(k.optimal = which.min(pe), pe.min = min(pe))
101 }
102 knn.bestKHP(dat[id.train,1:2], dat[id.test,1:2], dat[id.train,3], dat[id.test,3])
103

```

Console Output:

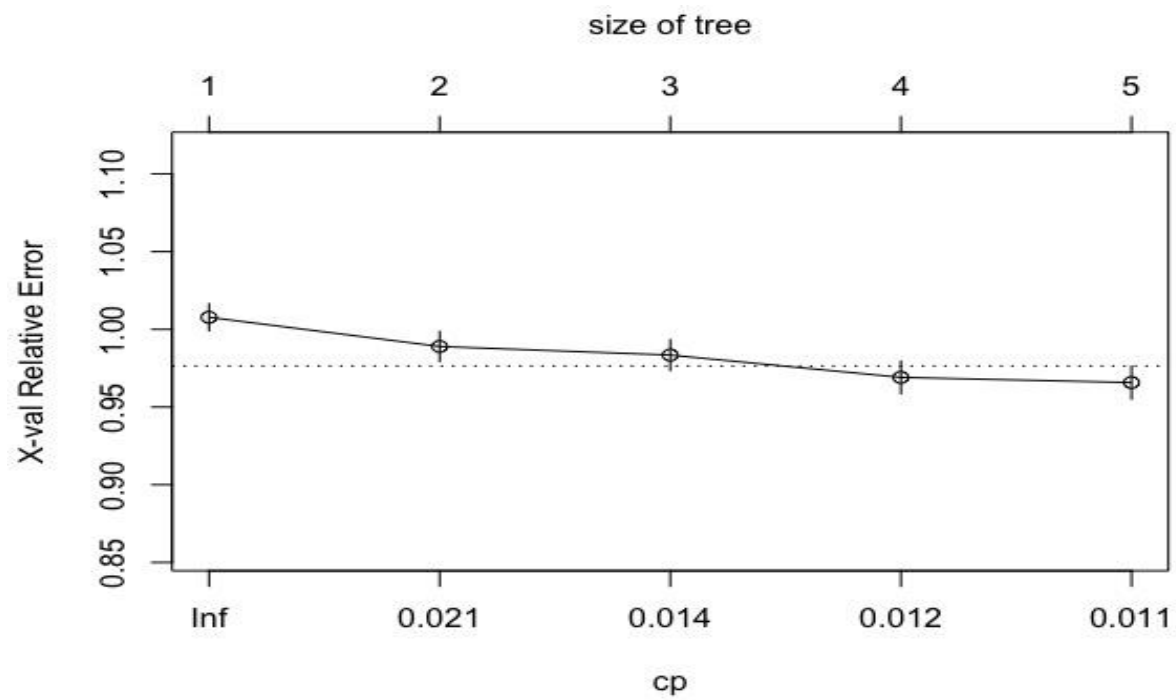
```

+ for (ii in 1:k.max) {
+   y.hat = knn(train, test, y.train, k = ii, prob=F)
+   pe[ii] = sum(y.hat != y.test)
+ }
+ list(k.optimal = which.min(pe), pe.min = min(pe))
+ }
> knn.bestKHP(dat[id.train,1:2], dat[id.test,1:2], dat[id.train,3], dat[id.test,3])
$k.optimal
[1] 17

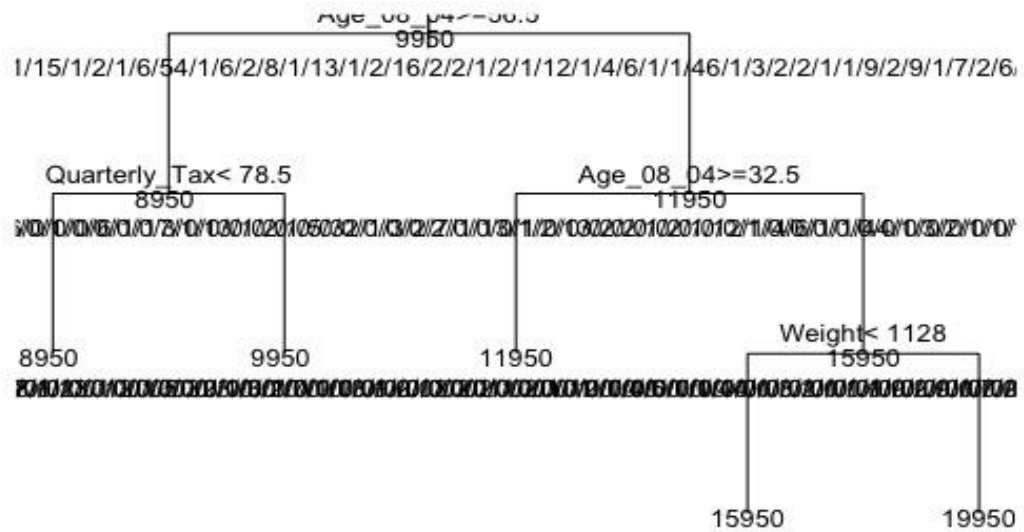
$pe.min
[1] 156
>

```

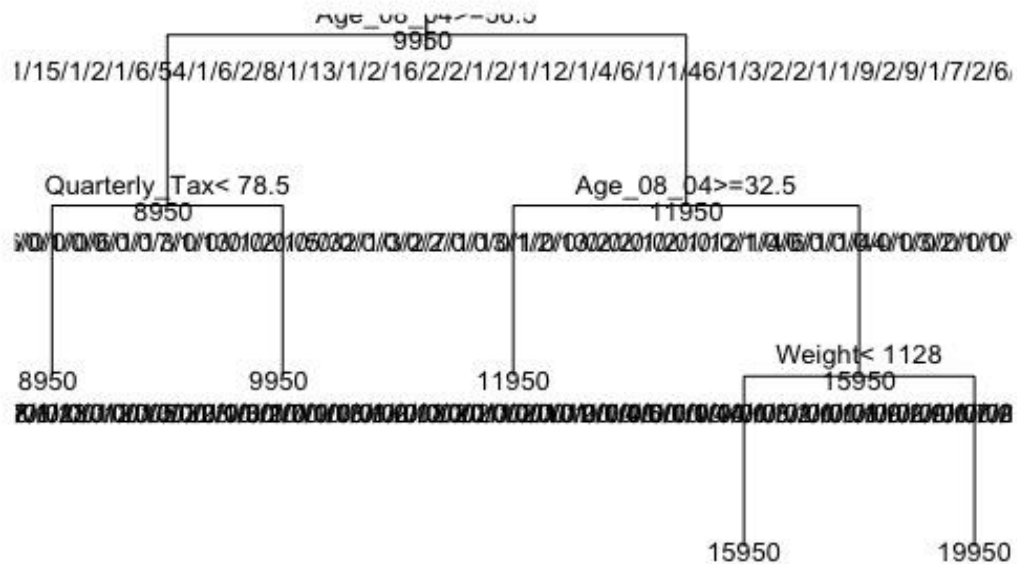

Classification tree:

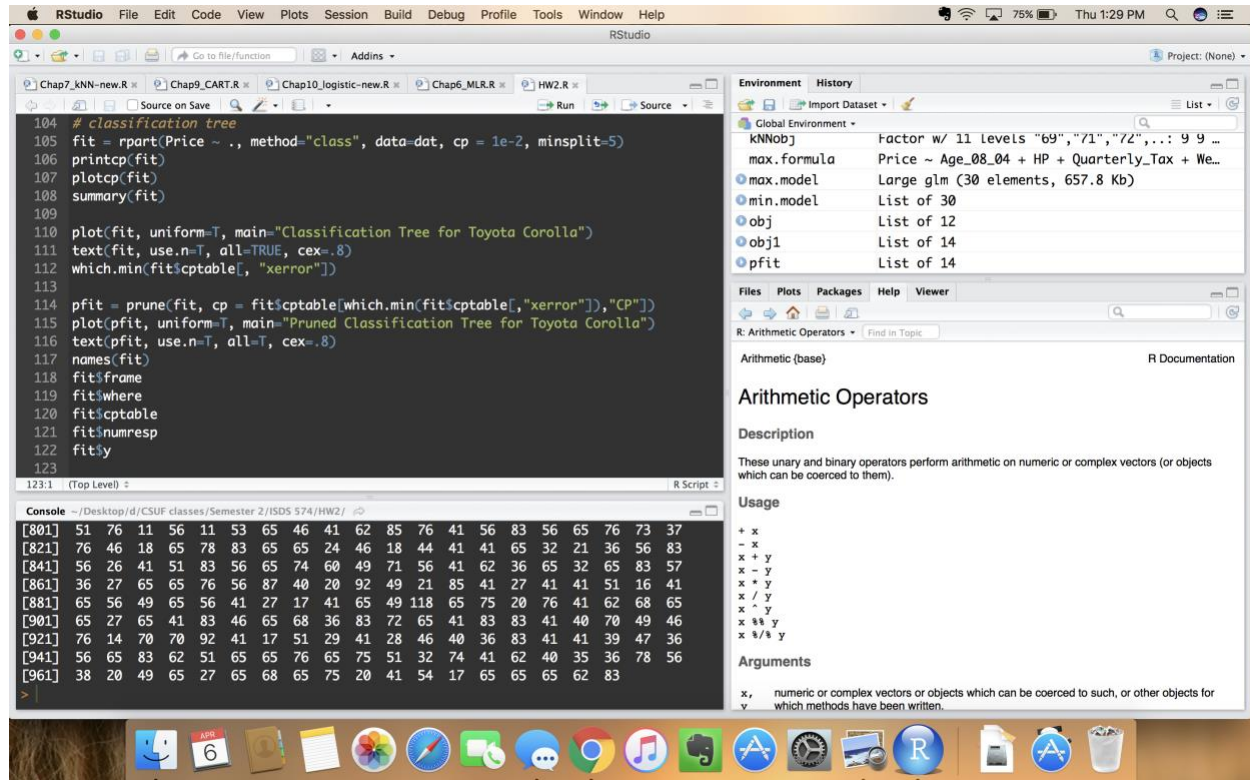


Classification Tree for Toyota Corolla



Pruned Classification Tree for Toyota Corolla





- f) Compare the models obtained and pick up the best one. Explain why you think it is the best one.

Classification tree is the best model because it gives lowest misclassification rate.