classification_Decision_Tree

Nivedita

2024-04-09

Objective

To fit classification decision tree for Carseats data set in ISLR2 package.

Analysis

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.3
```

```
attach(Carseats)
```

Carseats is a simulated data set containing sales of child car seats at 400 different stores. It is a data frame with 400 observations on the following 11 variables. The variables are as follows:

Sales-Unit sales (in thousands) at each location

CompPrice-Price charged by competitor at each location

Income-Community income level (in thousands of dollars)

Advertising-Local advertising budget for company at each location (in thousands of dollars)

Population-Population size in region (in thousands)

Price-Price company charges for car seats at each site

ShelveLoc-A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site

Age-Average age of the local population

Education-Education level at each location

Urban-A factor with levels No and Yes to indicate whether the store is in an urban or rural location

US-A factor with levels No and Yes to indicate whether the store is in the US or not

We want to fit classification tree for this data set. For this first we create a variable, called High, which takes on a value of Yes if the Sales variable exceeds 8, and takes on a value of No otherwise.

```
High <- factor(ifelse(Sales <= 8, "No", "Yes"))
#to merge High with the rest of the Carseats data.
Carseats <- data.frame(Carseats, High)
```

We now use the tree() function to fit a classification tree in order to predict High using all variables except sales.

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.3
```

```
tree.carseats <- tree(High ~ .- Sales, Carseats)
summary(tree.carseats)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

From summary we can see that “ShelveLoc”, “Price”, “Income”, “Population”, “Advertising”, “Age”,“CompPrice” and “US” are the variables which are used in tree construction. There are 27 terminal nodes and residual mean deviance is 0.4575. We see that the training error rate is 9%.

Now, for visual representation we use the plot() function to display the tree structure and the text() function to display the node labels.

```
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```

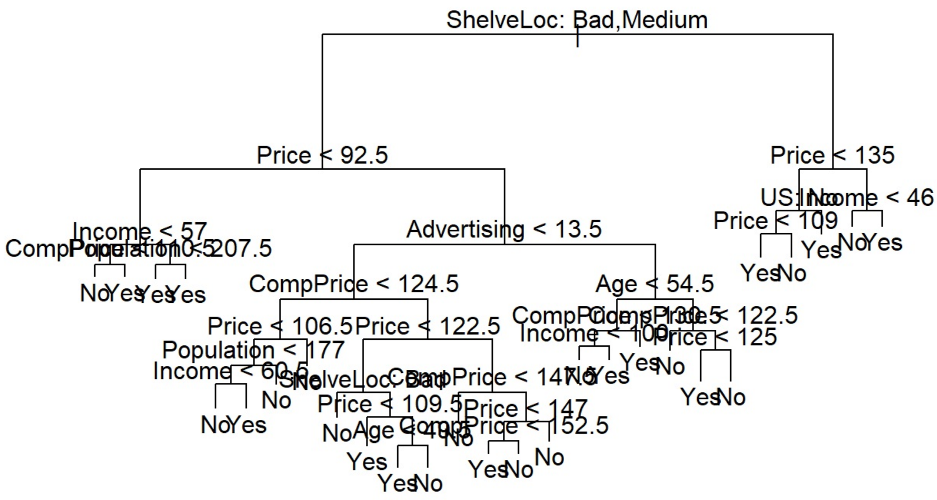


Fig.1 Plot of classification tree structure

In order to properly evaluate the performance of a classification tree on these data, we must estimate the test error rather than simply computing the training error. We split the observations into a training set and a test set, build the tree using the training set, and evaluate its performance on the test data.

```
set.seed(2)
train <- sample(1:nrow(Carseats), 200)
Carseats.test <- Carseats[-train, ]
High.test <- High[-train]
tree.carseats <- tree(High ~ .- Sales, Carseats, subset = train)
tree.pred <- predict(tree.carseats, Carseats.test, type = "class")
table(tree.pred, High.test)
```

```
##          High.test
## tree.pred  No Yes
##       No  104  33
##       Yes  13  50
```

```
(104 + 50) / 200
```

```
## [1] 0.77
```

We can see that this approach leads to correct predictions for around 77% of the locations in the test data set. To observe the number of terminal nodes of each tree considered (size) as well as the corresponding error rate and the value of the cost-complexity parameter used (k) we performed-

```
set.seed(7)
cv.carseats <- cv.tree(tree.carseats, FUN = prune.misclass)
names(cv.carseats)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```
cv.carseats
```

```
## $size
## [1] 21 19 14  9  8  5  3  2  1
##
## $dev
## [1] 75 75 75 74 82 83 83 85 82
##
## $k
## [1] -Inf  0.0  1.0  1.4  2.0  3.0  4.0  9.0 18.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

We can see that the tree with 9 terminal nodes results in only 74 cross-validation errors. Now we plot the error rate as a function of both size and k.

```
par(mfrow = c(1, 2))
plot(cv.carseats$size, cv.carseats$dev, type = "b",main='Plot of error rate function of size' )
plot(cv.carseats$k, cv.carseats$dev, type = "b",main='Plot of error rate function of k')
```

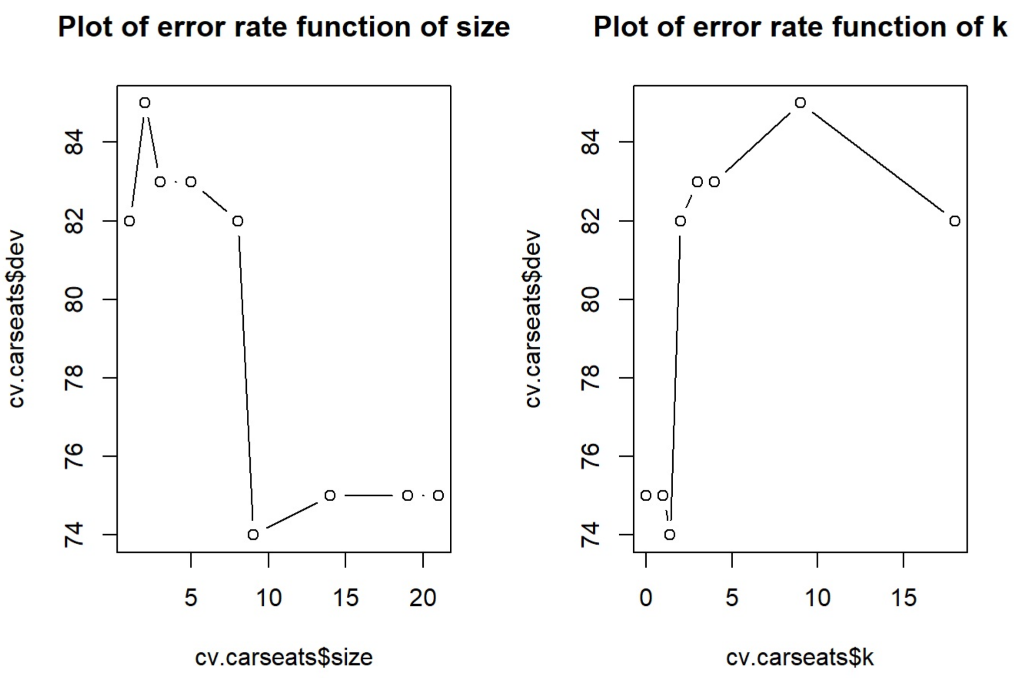


Fig. 2 Plot of error rate function of size and k

From Fig. 2 we can see that 9 terminal nodes results minnimum cross validation errors. We now want to check whether pruning the tree might lead to improved results so we apply the prune.misclass() function in order to prune the tree to obtain the nine-node tree.

```
prune.carseats <- prune.misclass(tree.carseats, best = 9)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```

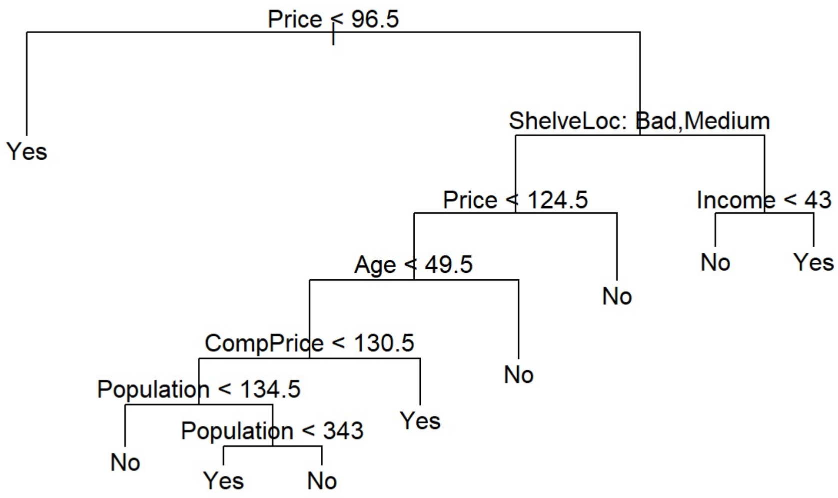


Fig. 3 Plot of prune tree

Now we test how does this prune tree perform on test data.

```
tree.pred <- predict(prune.carseats, Carseats.test, type = "class")
table(tree.pred, High.test)
```

```
##          High.test
## tree.pred No Yes
##       No  97  25
##       Yes 20  58
```

```
High.test
```

```
##   [1] Yes No  No  Yes No  No  Yes Yes Yes Yes No  Yes No  Yes Yes No  Yes Yes
##  [19] No  No  No  No  No  Yes No  No  No  No  Yes No  No  No  Yes Yes No  No
##  [37] Yes Yes No  Yes Yes No  No  No  No  No  Yes Yes No  No  No  Yes No  No
##  [55] Yes Yes No  No  No  No  Yes No  Yes Yes No  No  No  Yes No  Yes Yes No
##  [73] Yes Yes Yes No  No  Yes No  No  Yes Yes No  Yes No  Yes No  Yes Yes No
##  [91] No  No  No  No  No  No  No  No  Yes No  No  Yes Yes No  No  Yes Yes Yes
## [109] Yes No  Yes No  Yes Yes No  No  Yes No  No  Yes No  No  No  No  No  No
## [127] No  No  No  No  No  No  Yes No  No  No  No  No  No  Yes Yes Yes No  No
## [145] Yes No  Yes No  Yes Yes No  No  Yes No  Yes No  Yes No  No  Yes Yes No
## [163] No  No  Yes No  No  Yes Yes No  Yes No  Yes No  Yes Yes Yes Yes No  No
## [181] Yes No  Yes No  Yes No  Yes Yes No  No  Yes No  No  No  Yes No  Yes No
## [199] No  Yes
## Levels: No Yes
```

```
(97 + 58) / 200
```

```
## [1] 0.775
```

Now 77.5% of the test observations are correctly classified, so not only has the pruning process produced a more interpretable tree, but it has also slightly improved the classification accuracy.

Bagging

Here we apply bagging to the Carseats data, using the randomForest package in R. The argument mtry = 10 indicates that all 10 predictors should be considered for each split of the tree—in other words, that bagging should be done.

```
set.seed(1)
bag.Carseats <- randomForest(High ~ .- Sales, data = Carseats,subset = train, mtry = 10, importance = TRUE)
bag.Carseats
```

```
##
## Call:
##  randomForest(formula = High ~ . - Sales, data = Carseats, mtry = 10,      importance = TRUE, subset = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 27.5%
## Confusion matrix:
##     No Yes class.error
## No  98  21   0.1764706
## Yes 34  47   0.4197531
```

Now to check how well does this bagged model perform on the test set.

```
yhat.bag <- predict(bag.Carseats, Carseats.test, type = "class")
b.table<-table(yhat.bag,High.test)
b.table
```

```
##         High.test
## yhat.bag  No Yes
##      No  104  22
##      Yes  13  61
```

```
(104+61)/(104+22+13+61)
```

```
## [1] 0.825
```

Now from bagging 82.5% of the test observations are correctly classified which is more than that we have obtained using an optimally-pruned single tree.

Random Forest

Here we apply random forest to the Carseats data, using same randomForest package in R. Now we use $\sqrt{p}$ variables when building a random forest of classification trees. Here we use mtry=3.

```
set.seed(1)
rf.Carseats <-randomForest(High ~ .- Sales, data = Carseats, subset = train, mtry = 3, importance = TRUE)
yhat.rf <-predict(rf.Carseats, Carseats.test, type = "class")
r.table<-table(yhat.rf,High.test)
r.table
```

```
##        High.test
## yhat.rf  No Yes
##     No  110  24
##     Yes   7  59
```

```
(110+59)/(110+59+7+24)
```

```
## [1] 0.845
```

We can see that from random forest 84.5% of the test observations are correctly classified which is more than bagging.

We can use the importance() function, to view the importance of each variable. From this function two measures of variable importance are reported. The first is based upon the mean decrease of accuracy in predictions on the out of bag samples when a given variable is permuted. The second is a measure of the total decrease in node impurity that results from splits over that variable, averaged over all trees.

```
# to check importance
importance(rf.Carseats)
```

```
##                     No         Yes MeanDecreaseAccuracy MeanDecreaseGini
## CompPrice    2.1304913  2.21583500            3.0233624        11.332341
## Income      -1.2187382  0.08332858           -0.9127061        11.247612
## Advertising  2.7107740  9.28221142            8.0200032        11.539777
## Population  -2.5279538 -1.96748887           -3.4496266         9.515984
## Price       17.6102844 16.18029714           22.2205678        21.853751
## ShelveLoc   11.8799961 11.50873822           15.0797611        10.106813
## Age          3.7038805  6.83255965            6.9440472        11.738663
## Education   -2.9218854 -0.16331173           -2.5039109         6.031744
## Urban       -0.9337001  0.16405241           -0.4965739         1.114652
## US          -0.4330561  1.48555372            0.7195542         1.127466
```

```
#plot
varImpPlot(rf.Carseats,main = 'Plots of importance measures')
```

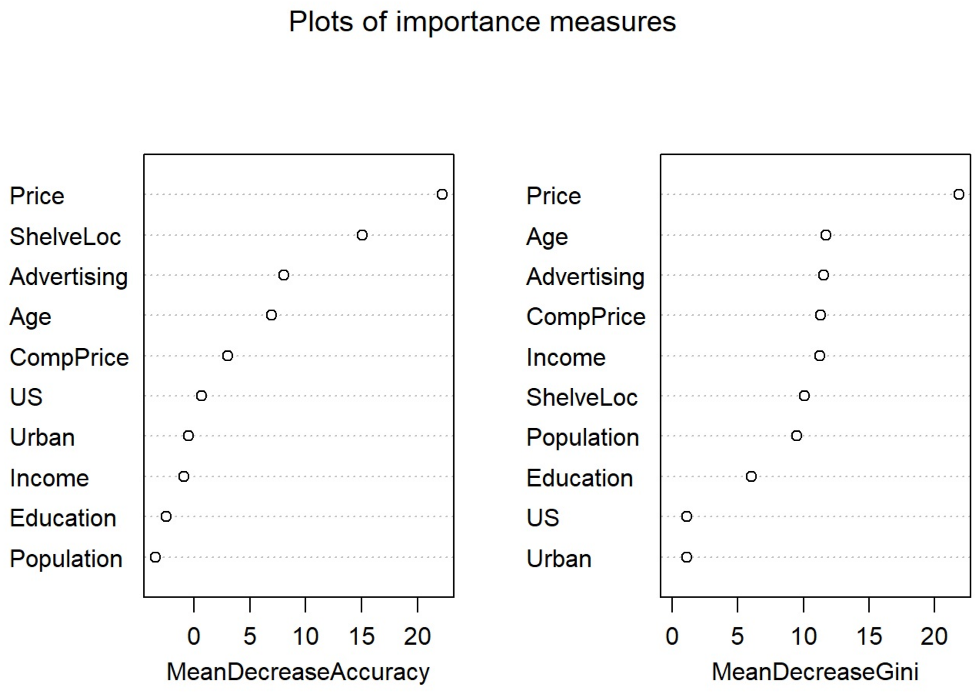


Fig. 4 Plots of importance measures

From Fig. 4 we can see that across all of the trees considered in the random forest, the Price company charges for car seats at each site (Price) is most important variables.

Boosting

Here we use the gbm package, and within it the gbm() function, to fit boosted gbm() regression trees to the Boston data set.

```
library(gbm)
```

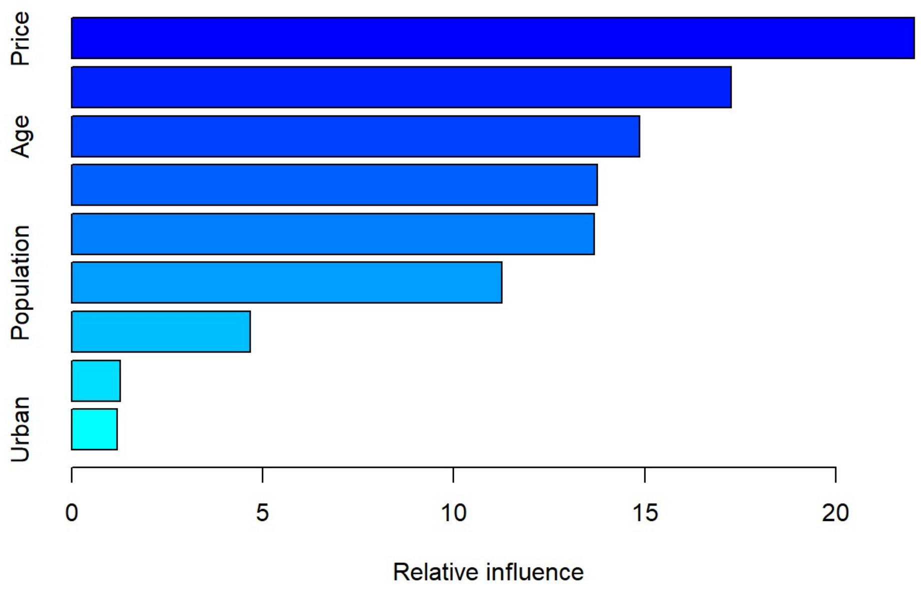
```
## Warning: package 'gbm' was built under R version 4.3.3
```

```
## Loaded gbm 2.1.9
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-dev
elopers/gbm3
```

```
set.seed(1)
#to convert into dummy variable
d1<-data.frame(na.omit(Carseats[train,c('US', "High",'Urban') ]),stringsAsFactors=FALSE)
dyn<-ifelse(d1 == "Yes",1,0)
c.train<-Carseats[train, ]
boost.d1<-c.train[,-c(10:12)]
boost.d2<-boost.d1[,-7]
carseat.d<-data.frame(boost.d2,dyn)

boost.Carseats <- gbm(High ~ .- Sales, data = carseat.d, distribution = "bernoulli", n.trees = 5000, interaction.
depth = 4)
summary(boost.Carseats)
```




```
##                     var   rel.inf
## Price             Price 22.052626
## CompPrice     CompPrice 17.257112
## Age                 Age 14.870763
## Income           Income 13.756026
## Advertising Advertising 13.674207
## Population   Population 11.262025
## Education     Education  4.670754
## US                   US  1.268163
## Urban             Urban  1.188323
```

From summary we can see that Price is by far the most important variables. We can also produce partial dependence plots for this variable

```
plot(boost.Carseats, i = "Price", main='Partial dependence plot of Price')
```

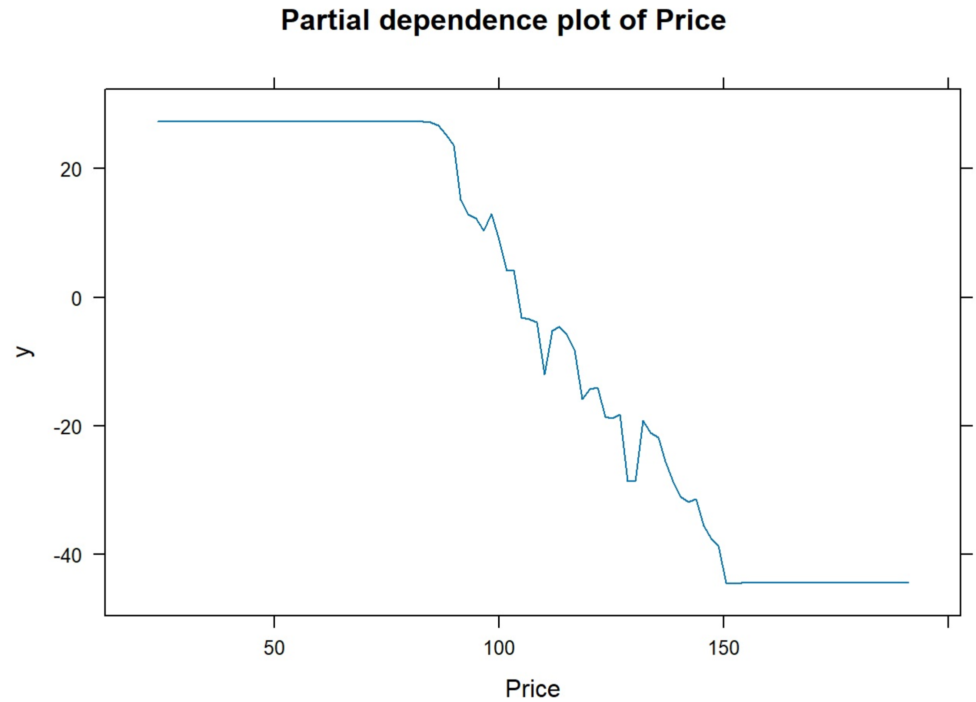


Fig. 5 Partial dependence plot of Price

Conclusion

We performed fitting of classification tree, bagging, boosting and random forest for Carseats data set in ISLR2 package. We observed that classification accuracy of random forest is maximum and Price is the most important variable.