Homework.2

Eleonora Giuliani 247161

2024-05-05

Introduction

Prostate cancer is one of the most prevalent cancers among men. In this study, it is investigated the association between the level of prostate-specific antigen (PSA) and various clinical measures in a cohort of 97 men who were about to receive a radical prostatectomy. In particular, the explanatory variables are: `lcavol`: log(cancer volume in cm3), `lweight`: log(prostate weight in g), `age` in years,lbph: `log(amount of benign prostatic hyperplasia in cm2)`,svi: `seminal vesicle invasion (1 = yes, 0 = no)`,lcp: `log(capsular penetration in cm)`,gleason: `Gleason score for prostate cancer (6,7,8,9)`,pgg45‘: percentage of Gleason scores 4 or 5, recorded over their visit history before their final current Gleason score.

As previously mentioned, the variables included in the dataset are 9 and they are all numerical variables, specifically: lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45 and lpsa. Before properly starting the analysis, we shall briefly examine the present dataset, beginning verifying the correctness of the opening.

```
head(df0)
```

```
      lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
```

It is also important to verify the presence of NA values. In this case they are not present.

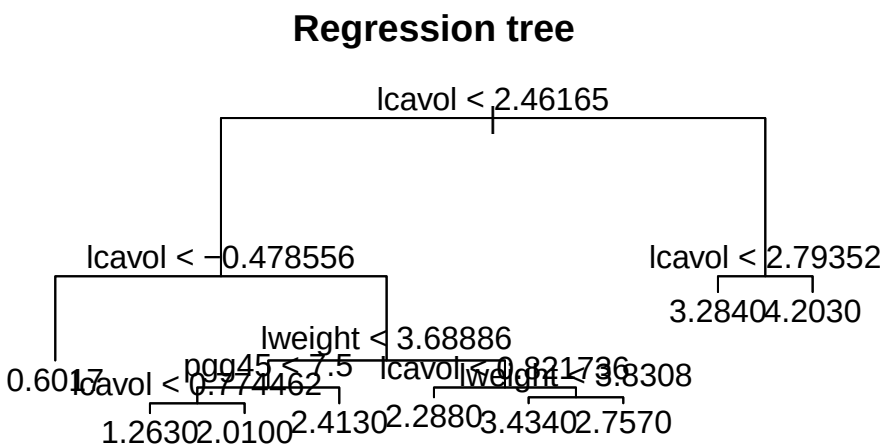
The next step is to fit the data into a decision tree model using the “tree” package in R, with ‘lpsa’ as the response variable and other variables as predictors. The analysis reveals that

the variables were used to construct the tree are only three: 'lcavol', 'lweight' and 'pgg45'. Additionally, the tree comprises 9 terminal nodes, and that the residual mean deviance of 0.4119 indicates a reasonable fit of the model to the data.

```
library(tree)
set.seed(1)
tree.data <- tree(lpsa ~ ., data = df0)
summary(tree.data)
```

```
Regression tree:
tree(formula = lpsa ~ ., data = df0)
Variables actually used in tree construction:
[1] "lcavol"  "lweight" "pgg45"
Number of terminal nodes:  9
Residual mean deviance:  0.4119 = 36.24 / 88
Distribution of residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-1.499000 -0.488000  0.003621  0.000000  0.481200  1.380000
```

Following there is the plot of the model.



To select the optimal tree complexity, it is necessary to perform cross-validation. Indeed, by analyzing cross-validated error rates for different tree sizes, it is possible to decide whether to prune the tree.

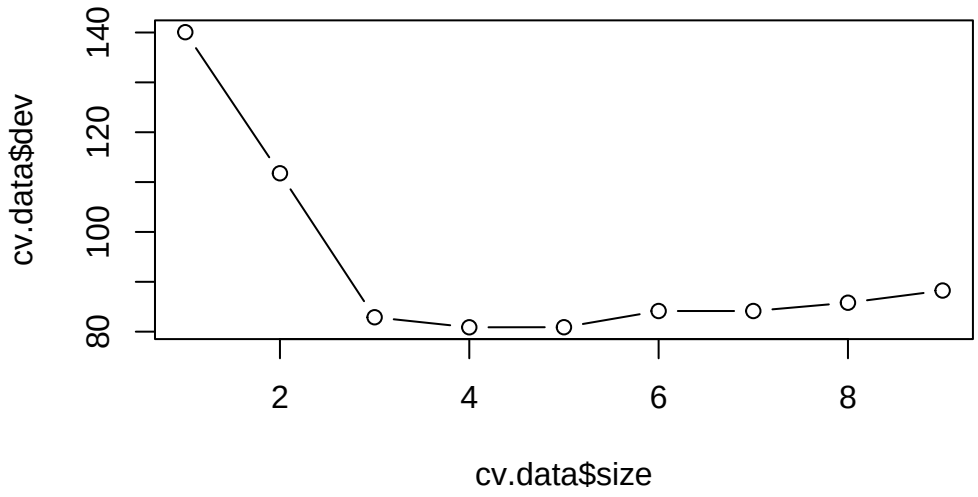
```
set.seed(1)
cv.data <- cv.tree(object = tree.data)
names(cv.data)
```

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

```
cv.data
```

```
$size
[1] 9 8 7 6 5 4 3 2 1

$dev
[1]  88.24714  85.80665  84.14467  84.14467  80.89206  80.87758  82.89215
[8] 111.76913 140.06860

$k
[1]      -Inf  1.687786  2.746385  2.758375  4.427103  4.445951  7.587544
[8] 23.619667 44.401279

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

```r
plot(cv.data$size, cv.data$dev, type = "b")
```



To determine the optimal tree complexity, it is typically necessary to look for the tree size corresponding to the lowest cross-validated error. In this specific case, the lowest cross-validated error(80.87758) is associated with a tree having 4 terminal nodes(size 4). However, as we can observe, the deviance is not strictly decreasing with the tree size. To prevent overfitting, it is prudent to choose the point where the deviance stops decreasing or starts to increase. A good choice could be to select a tree of 4 or 5 for pruning.

```r
set.seed(1)
prune_tree <- prune.tree(tree.data, best = 4)
summary(prune_tree)
```

```
Regression tree:
snip.tree(tree = tree.data, nodes = c(11L, 3L, 10L))
Variables actually used in tree construction:
[1] "lcavol"  "lweight"
Number of terminal nodes:  4
Residual mean deviance:  0.5625 = 52.31 / 93
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.66200 -0.54020 -0.01154  0.00000  0.44560  1.81700
```

Generally, a lower residual mean deviance indicates a better model fit to the data. In this specific case, the residual mean deviance of the unpruned tree is 0.4119, which is lower than the residual mean deviance of the pruned tree(0.5625). So,the unpruned tree has a lower residual mean deviance and is more complex with more terminal nodes. Therefore,based on these factors, it would be considered better.

Now, using the 'randomForest' package, the dataset is fitted to a random forest model. By adjusting the 'mtry' parameter, it is possible to change the number of predictors considered at each split. To find the best fit, the model is evaluated using different values of 'mtry'.

```
library(randomForest)
set.seed(1)
n_pred <- ncol(df0) - 1
bag.prostate <- randomForest(lpsa ~ ., df0, mtry = n_pred, importance = TRUE)
bag.prostate
```

```
Call:
 randomForest(formula = lpsa ~ ., data = df0, mtry = n_pred, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 8

          Mean of squared residuals: 0.6015186
                    % Var explained: 54.39
```

```
# random forest
library(randomForest)
set.seed(1)
bag.prostate <- randomForest(lpsa ~ ., df0, mtry = 5, importance = TRUE)
bag.prostate
```

```
Call:
 randomForest(formula = lpsa ~ ., data = df0, mtry = 5, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 5

          Mean of squared residuals: 0.5846996
                    % Var explained: 55.66
```

Based on these results, the second random forest model with 'mtry = 5' appears to be the preferred option.

We can check out how important each predictor is by using the `importance()` function. The table shows the importance of each variable in the random forest model, as measured by two metrics: '%IncMSE' and 'IncNodePurity'.

```
knitr::kable(importance(bag.prostate))
```

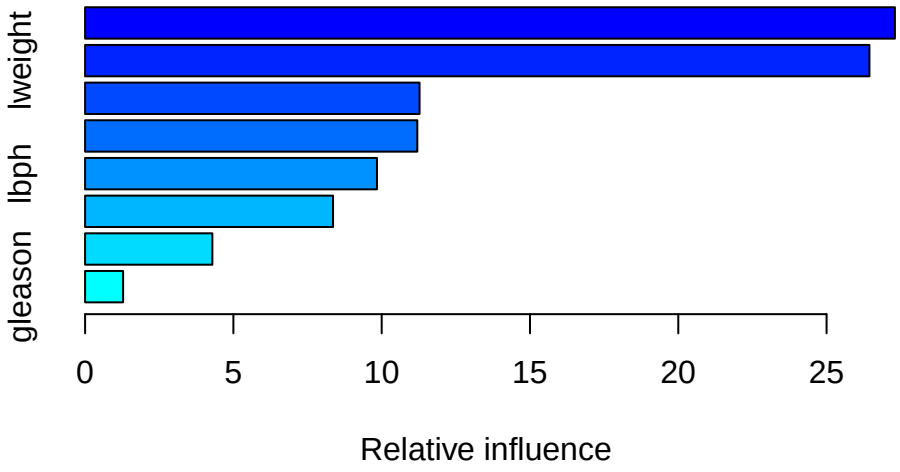
	%IncMSE	IncNodePurity
lcavol	27.557636	59.693300
lweight	13.122026	18.878593
age	-0.039568	6.290316
lbph	4.224494	6.409341
svi	11.457079	12.511114
lcp	2.887746	10.097115
gleason	3.900577	1.921502
pgg45	6.242141	7.810063

To neatly plot these importance measures we use the `varImpPlot()` function:

```
# arImpPlot(bag.prostate)
```

The table demonstartes that 'lcavol' and 'lweight' exhibit higher values in both the metrics, suggesting that they are more influential predictors in the model. To fit boosted regression to the prostate dataset will use the 'gbm' package and its 'gbm( )' function. The regression task requires that we use the option distribution ='gaussian'.

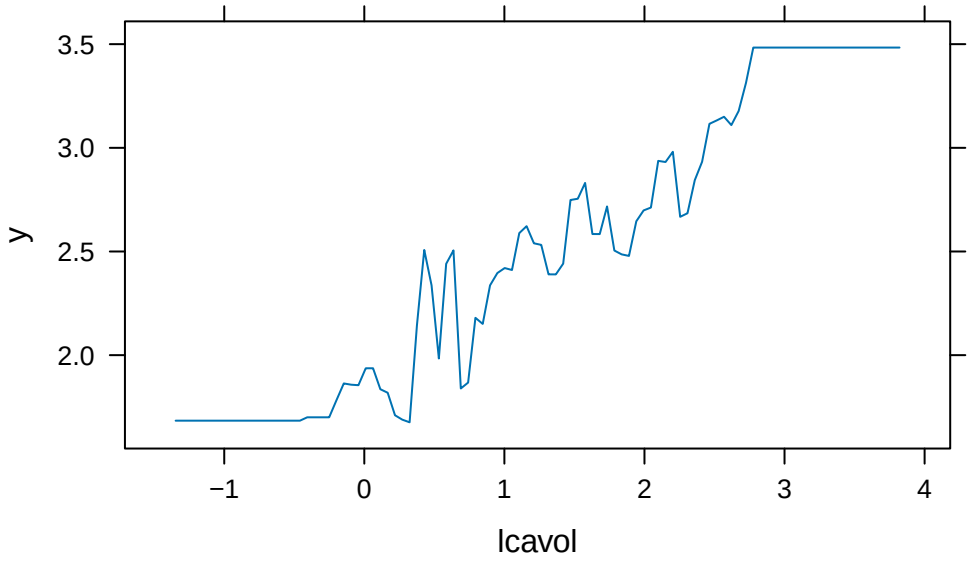
```
library(gbm)
set.seed(1)
boosted <- gbm(lpsa ~ ., data = df0, distribution = "gaussian", n.tree = 5000, interaction.d

knitr::kable(summary(boosted))
```



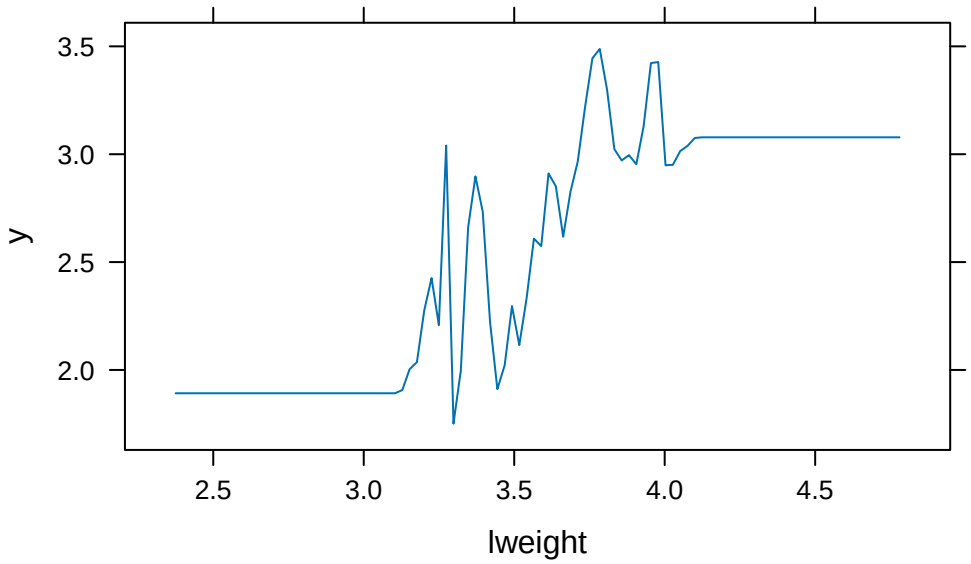
	var	rel.inf
lcavol	lcavol	27.304432
lweight	lweight	26.444565
lcp	lcp	11.276987
age	age	11.201128
lbph	lbph	9.840938
pgg45	pgg45	8.360313
svi	svi	4.290535
gleason	gleason	1.281101

It is possible observed that the top two variables are 'lcavol' and 'lweight' whose have the higher rel.inf values. We can evaluate the marginal effect of these two variables by producing a partial dependence plot.

```
par(mfrow = c(1, 2))
plot(boosted, i = "lcavol")
```




```
plot(boosted, i = "lweight")
```



The two plots illustrate that the lpsa value generally tends to increase as both the 'lcavol' and 'lweight' parameters increase, but not consistently. Instead, there are numerous oscillations.

At this point, we perform cross-validation tests of all three models, seeing. The results of the cross-validation or the regression tree are not reported again. For performing the cross-validation, it is used the library 'caret'.

```
library(caret)
library(randomForest)

train_data <- df0

rf_model <- train(lpsa ~ ., data = train_data, method = "rf", trControl = trainControl(metho
    number = 10), tuneLength = 5)
print(rf_model)
```

```
Random Forest

97 samples
 8 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 87, 88, 88, 86, 89, 88, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  2     0.7856826  0.5801801  0.6346886
  3     0.7786756  0.5974102  0.6350845
  5     0.7772182  0.6121588  0.6412602
  6     0.7746205  0.6187002  0.6412306
  8     0.7835340  0.6182379  0.6435797

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 6.
```

Same code for the boosted model with the only variation being the method set to 'gbm'.

```
Stochastic Gradient Boosting

97 samples
 8 predictor

No pre-processing
```

```
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 88, 87, 88, 88, 88, 87, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  RMSE       Rsquared   MAE
  1                   50      0.7718889  0.6070567  0.5971419
  1                  100      0.7957266  0.5916886  0.6085398
  1                  150      0.8158522  0.5715350  0.6195050
  2                   50      0.7772521  0.6046208  0.6028928
  2                  100      0.7870049  0.5963740  0.6056273
  2                  150      0.8203012  0.5631915  0.6314654
  3                   50      0.8013201  0.5800910  0.6126296
  3                  100      0.8288817  0.5549440  0.6258110
  3                  150      0.8410993  0.5517946  0.6365612

Tuning parameter 'shrinkage' was held constant at a value of 0.1

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 50, interaction.depth =
 1, shrinkage = 0.1 and n.minobsinnode = 10.
```

For the Random Forest model, the optimal RMSE was achieved with 'mtry=5' yielding an RMSE value of approximately 0.744. For the Boosting model, the optimal RMSE was achieved with 'n.tree=50' and 'interaction.depth=1', resulting in an RMSE value of approximately 0.779. So, comparing the results, it appears that the Random Forest model performed better that the Boosting model.

Considering the Decision Tree model, we can see that the mean of squared residual is 0.4119, indicating a low RMSE compared to the other models.

In summary, comparing all the results, it seems that the Decision Tree model performed better that both the Random Forest and Boosting models.