R Notebook

Code

Hide

```
#Loading the data file for Regression
attach(parkinsons_updrs)
```

Hide

```
#Loading and cleaning the data file for Classification
attach(breast.cancer.wisconsin)

#Missing Value check and Solution
breast.cancer.wisconsin$V7 <- replace(breast.cancer.wisconsin$V7, breast.cancer.wisconsin$V7
== "?", NA)
breast.cancer.wisconsin$V7 <- as.integer(breast.cancer.wisconsin$V7)

breast_cancer_wisconsin<-na.omit(breast.cancer.wisconsin)
breast_cancer_wisconsin
```

	V1 <int>	V2 <int>	V3 <int>	V4 <int>	V5 <int>	V6 <int>	V7 <int>	V8 <int>	V9 <int>
1	1000025	5	1	1	1	2	1	3	1
2	1002945	5	4	4	5	7	10	3	2
3	1015425	3	1	1	1	2	2	3	1
4	1016277	6	8	8	1	3	4	3	7
5	1017023	4	1	1	3	2	1	3	1
6	1017122	8	10	10	8	7	10	9	7
7	1018099	1	1	1	1	2	10	3	1
8	1018561	2	1	2	1	2	1	3	1
9	1033078	2	1	1	1	2	1	1	1
10	1033078	4	2	1	1	2	1	2	1

1-10 of 683 rows | 1-10 of 11 columns Previous **1** 2 3 4 5 6 … 69 Next

Hide

```
summary(parkinsons_updrs)
```

```
   subject.          age             sex           test_time      
 Min.   : 1.00   Min.   :36.0   Min.   :0.0000   Min.   : -4.263  
 1st Qu.:10.00   1st Qu.:58.0   1st Qu.:0.0000   1st Qu.: 46.847  
 Median :22.00   Median :65.0   Median :0.0000   Median : 91.523  
 Mean   :21.49   Mean   :64.8   Mean   :0.3178   Mean   : 92.864  
 3rd Qu.:33.00   3rd Qu.:72.0   3rd Qu.:1.0000   3rd Qu.:138.445  
 Max.   :42.00   Max.   :85.0   Max.   :1.0000   Max.   :215.490  
  motor_UPDRS      total_UPDRS      Jitter...          Jitter.Abs.      
 Min.   : 5.038   Min.   : 7.00   Min.   :0.000830   Min.   :2.250e-06  
 1st Qu.:15.000   1st Qu.:21.37   1st Qu.:0.003580   1st Qu.:2.244e-05  
 Median :20.871   Median :27.58   Median :0.004900   Median :3.453e-05  
 Mean   :21.296   Mean   :29.02   Mean   :0.006154   Mean   :4.403e-05  
 3rd Qu.:27.596   3rd Qu.:36.40   3rd Qu.:0.006800   3rd Qu.:5.333e-05  
 Max.   :39.511   Max.   :54.99   Max.   :0.099990   Max.   :4.456e-04  
   Jitter.RAP        Jitter.PPQ5         Jitter.DDP       
 Min.   :0.000330   Min.   :0.000430   Min.   :0.000980  
 1st Qu.:0.001580   1st Qu.:0.001820   1st Qu.:0.004730  
 Median :0.002250   Median :0.002490   Median :0.006750  
 Mean   :0.002987   Mean   :0.003277   Mean   :0.008962  
 3rd Qu.:0.003290   3rd Qu.:0.003460   3rd Qu.:0.009870  
 Max.   :0.057540   Max.   :0.069560   Max.   :0.172630  
    Shimmer          Shimmer.dB.      Shimmer.APQ3       Shimmer.APQ5      
 Min.   :0.00306   Min.   :0.026   Min.   :0.00161   Min.   :0.00194  
 1st Qu.:0.01912   1st Qu.:0.175   1st Qu.:0.00928   1st Qu.:0.01079  
 Median :0.02751   Median :0.253   Median :0.01370   Median :0.01594  
 Mean   :0.03404   Mean   :0.311   Mean   :0.01716   Mean   :0.02014  
 3rd Qu.:0.03975   3rd Qu.:0.365   3rd Qu.:0.02057   3rd Qu.:0.02375  
 Max.   :0.26863   Max.   :2.107   Max.   :0.16267   Max.   :0.16702  
 Shimmer.APQ11       Shimmer.DDA            NHR                HNR         
 Min.   :0.00249   Min.   :0.00484   Min.   :0.000286   Min.   : 1.659  
 1st Qu.:0.01566   1st Qu.:0.02783   1st Qu.:0.010955   1st Qu.:19.406  
 Median :0.02271   Median :0.04111   Median :0.018448   Median :21.920  
 Mean   :0.02748   Mean   :0.05147   Mean   :0.032120   Mean   :21.680  
 3rd Qu.:0.03272   3rd Qu.:0.06173   3rd Qu.:0.031463   3rd Qu.:24.444  
 Max.   :0.27546   Max.   :0.48802   Max.   :0.748260   Max.   :37.875  
      RPDE              DFA               PPE         
 Min.   :0.1510   Min.   :0.5140   Min.   :0.02198  
 1st Qu.:0.4698   1st Qu.:0.5962   1st Qu.:0.15634  
 Median :0.5423   Median :0.6436   Median :0.20550  
 Mean   :0.5415   Mean   :0.6532   Mean   :0.21959  
 3rd Qu.:0.6140   3rd Qu.:0.7113   3rd Qu.:0.26449  
 Max.   :0.9661   Max.   :0.8656   Max.   :0.73173  
```

Hide

```
#Descriptive Dataset's Info
#Class Label- 2 - Benign and 4- Malignant
names(breast_cancer_wisconsin)<-c('ID', 'C_Thickness','UCSize', 'UCShape', 'M_Adhesion', 'Sin
gle_ECSize', 'Bare_Nuclei', 'Bland_Chromatin', 'Normal_Nucleoli', 'Mitoses', 'Class_Label')

#Encondings of Class Label
breast_cancer_wisconsin$Class_Label<-ifelse(breast_cancer_wisconsin$Class_Label== 2, 0, 1)
breast_cancer_wisconsin$Class_Label
```

```
  [1] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0
 [36] 1 0 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 0 0 1
 [71] 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1
[106] 1 0 1 0 1 1 1 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1
[141] 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 1 0 1 0
[176] 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 1
[211] 0 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 1 0
[246] 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 1
[281] 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 1 0 1
[316] 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0
[351] 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
[386] 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0
[421] 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0
[456] 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
[491] 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
[526] 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1
[561] 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0
[596] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
[631] 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1
[666] 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1
```

Hide

```
summary(breast_cancer_wisconsin)
```

```
       ID            C_Thickness         UCSize          UCShape      
 Min.   :   63375   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
 1st Qu.:  877617   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000  
 Median : 1171795   Median : 4.000   Median : 1.000   Median : 1.000  
 Mean   : 1076720   Mean   : 4.442   Mean   : 3.151   Mean   : 3.215  
 3rd Qu.: 1238705   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000  
 Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000  
  M_Adhesion    Single_ECSize     Bare_Nuclei     Bland_Chromatin 
 Min.   : 1.00   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
 1st Qu.: 1.00   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 2.000  
 Median : 1.00   Median : 2.000   Median : 1.000   Median : 3.000  
 Mean   : 2.83   Mean   : 3.234   Mean   : 3.545   Mean   : 3.445  
 3rd Qu.: 4.00   3rd Qu.: 4.000   3rd Qu.: 6.000   3rd Qu.: 5.000  
 Max.   :10.00   Max.   :10.000   Max.   :10.000   Max.   :10.000  
 Normal_Nucleoli    Mitoses        Class_Label    
 Min.   : 1.00   Min.   : 1.000   Min.   :0.0000  
 1st Qu.: 1.00   1st Qu.: 1.000   1st Qu.:0.0000  
 Median : 1.00   Median : 1.000   Median :0.0000  
 Mean   : 2.87   Mean   : 1.603   Mean   :0.3499  
 3rd Qu.: 4.00   3rd Qu.: 1.000   3rd Qu.:1.0000  
 Max.   :10.00   Max.   :10.000   Max.   :1.0000  
```

Hide

```
#Divide the data into training and testing

library(caret)
```

```
Loading required package: ggplot2
Loading required package: lattice
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
```

Hide

```
set.seed(1) # for reproducibility

# Create a vector of row indices
rows <- 1:nrow(parkinsons_updrs)


# Randomly sample 80% of the row indices for the training set
training_rows <- sample(rows, floor(0.8 * length(rows)))

# The remaining rows are for the testing set
testing_rows <- setdiff(rows, training_rows)

# Write the training and testing sets to separate files
write.table(parkinsons_updrs[training_rows, ], file = "Park_training_data.txt", row.names = F
ALSE, col.names = FALSE)
write.table(parkinsons_updrs[testing_rows, ], file = "Park_testing_data.txt", row.names = FAL
SE, col.names = FALSE)

training_data_old <- parkinsons_updrs[training_rows, ]
testing_data_old <- parkinsons_updrs[-training_rows, ]

# Remove the variable 'motor_UPDRS' (Training and Testing)
training_data <- subset(training_data_old, select = -motor_UPDRS)
testing_data <- subset(testing_data_old, select = -motor_UPDRS)


# Create X.train and X.test data frames that exclude the total_UPDRS
X.train <- training_data[, -which(names(training_data) == "total_UPDRS")]
X.test <- testing_data[, -which(names(testing_data) == "total_UPDRS")]
Y.train <- training_data$total_UPDRS
Y.test <- testing_data$total_UPDRS

#b) Division Verification in number of Examples
cat("Number of examples in training data:", nrow(training_data), "\n")
```

```
Number of examples in training data: 4700
```

Hide

```
cat("Number of examples in testing data:", nrow(testing_data), "\n")
```

```
Number of examples in testing data: 1175
```

Hide

```
#Divide the data into training(80%) and testing(20%)

breast_cancer_wisconsin$Class_Label <- factor(breast_cancer_wisconsin$Class_Label)

#a) Divide the data into training and testing sets
library(caret)
set.seed(1) # for reproducibility

# Create a vector of row indices
rows1 <- 1:nrow(breast_cancer_wisconsin)

# Randomly sample 80% of the row indices for the training set
training_rows1 <- sample(rows1, floor(0.8 * length(rows1)))

# The remaining rows are for the testing set
testing_rows1 <- setdiff(rows1, training_rows1)

training_data1 <- breast_cancer_wisconsin[training_rows1, ]
testing_data1 <- breast_cancer_wisconsin[-training_rows1, ]

# Create X.train and X.test data frames that exclude the class label
X.train1 <- training_data1[, -which(names(training_data1) == "Class_Label")]
X.test1 <- testing_data1[, -which(names(testing_data1) == "Class_Label")]
Y.train1 <- training_data1$Class_Label
Y.test1 <- testing_data1$Class_Label

#b) Division Verification in number of Examples
cat("Number of examples in training data:", nrow(training_data1), "\n")
```

```
Number of examples in training data: 546
```

Hide

```
cat("Number of examples in testing data:", nrow(testing_data1), "\n")
```

```
Number of examples in testing data: 137
```

Hide

```
#1.a) Bagging - Regression
library(randomForest)
```

```
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.

Attaching package: ‘randomForest’

The following object is masked from ‘package:ggplot2’:

    margin
```

Hide

```
# Bagging is just and RF with m=p
set.seed(1)
# mtry: number of predictors (m) to be taken into account in the model
bag.park=randomForest(total_UPDRS~.,training_data,mtry=20,importance=T)
bag.park
```

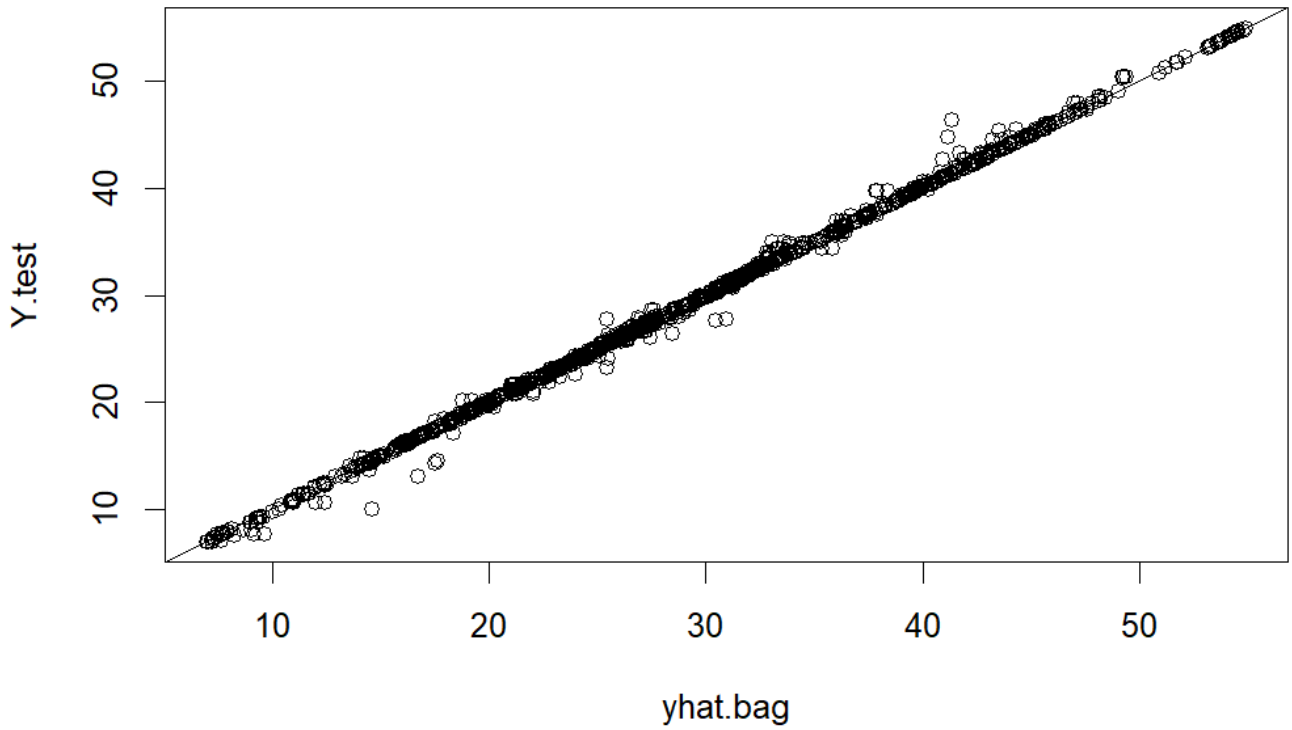
```

Call:
 randomForest(formula = total_UPDRS ~ ., data = training_data,      mtry = 20, importance =
T)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 20

          Mean of squared residuals: 0.2014956
                    % Var explained: 99.82
```

Hide

```
# Performance on test set
yhat.bag=predict(bag.park,testing_data)
plot(yhat.bag, Y.test)
abline(0,1)
```



Hide

```
#i)Performance Test MSE
mean((yhat.bag-Y.test)^2)
```

```
[1] 0.2111907
```

Hide

```
###################
# b) Random Forests - Regression

library(randomForest)
# By default randomForest() uses m=p/3 for regression and m=sqrt(p) for classification
set.seed(1)
rf.park=randomForest(total_UPDRS~.,training_data,mtry=20/3,importance =T)
yhat.rf = predict(rf.park ,testing_data)

#i) Test MSE
mean((yhat.rf-Y.test)^2)
```

```
[1] 2.827568
```

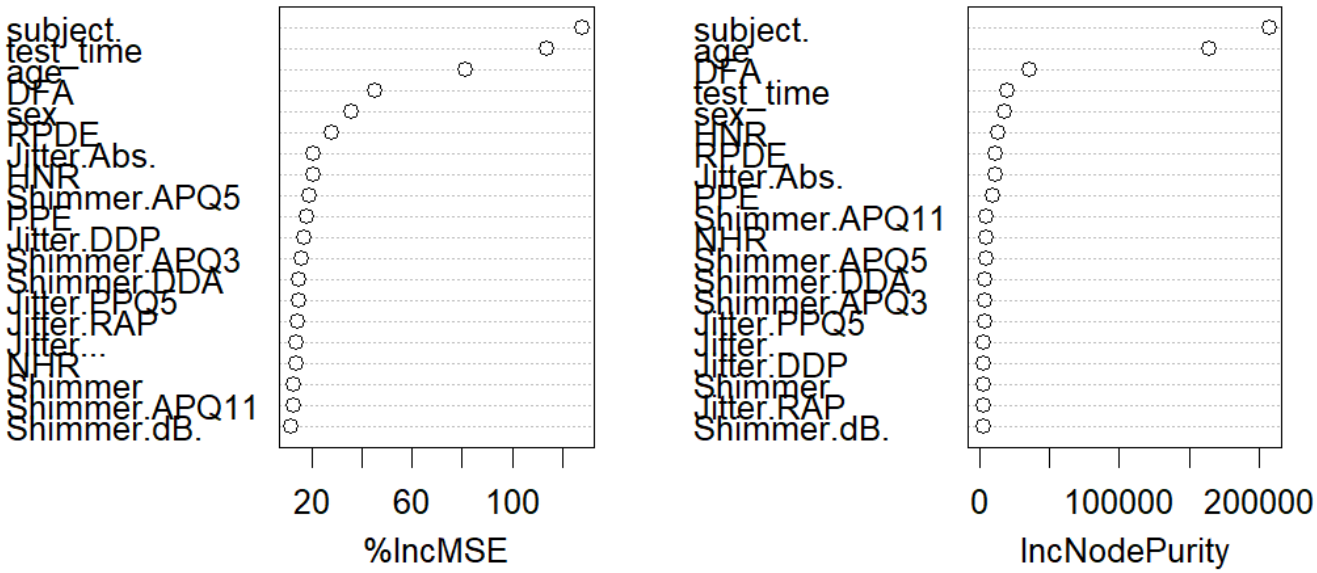
Hide

```
# importance(): view the importance of each variable
# %IncMSE: mean decrease of accuracy in predictions on the OOB samples when a
# given variable is excluded from the model
# IncNodeImpurity: total decrease in node impurity that results from splits over
# that variable, averaged over all trees (RSS in regr. vs. deviance in class)
importance(rf.park)
```

```
               %IncMSE IncNodePurity
subject.      128.04524    207920.402
age            81.36791    164116.633
sex            35.83029     18302.275
test_time     114.15915     20017.791
Jitter...      13.80827      3562.756
Jitter.Abs.    20.67854     11222.622
Jitter.RAP     14.19765      3221.542
Jitter.PPQ5    14.66408      3938.261
Jitter.DDP     16.56443      3501.982
Shimmer        12.70978      3276.419
Shimmer.dB.    11.61624      3045.770
Shimmer.APQ3   15.51536      4003.207
Shimmer.APQ5   18.87825      4567.601
Shimmer.APQ11  12.56983      5045.740
Shimmer.DDA    14.82533      4172.176
NHR            13.67961      4998.148
HNR            20.19554     13443.800
RPDE           27.98000     11823.178
DFA            45.28773     36006.964
PPE            17.58112      9210.416
```

Hide

```
# varImpPlot(): Variance importance plot
varImpPlot(rf.park)
```



Hide

```
#c) Boosting - Regression

# gbm: library for boosting
library(gbm)
```

```
Warning: package ‘gbm’ was built under R version 4.2.3Loaded gbm 2.1.8.1
```

Hide

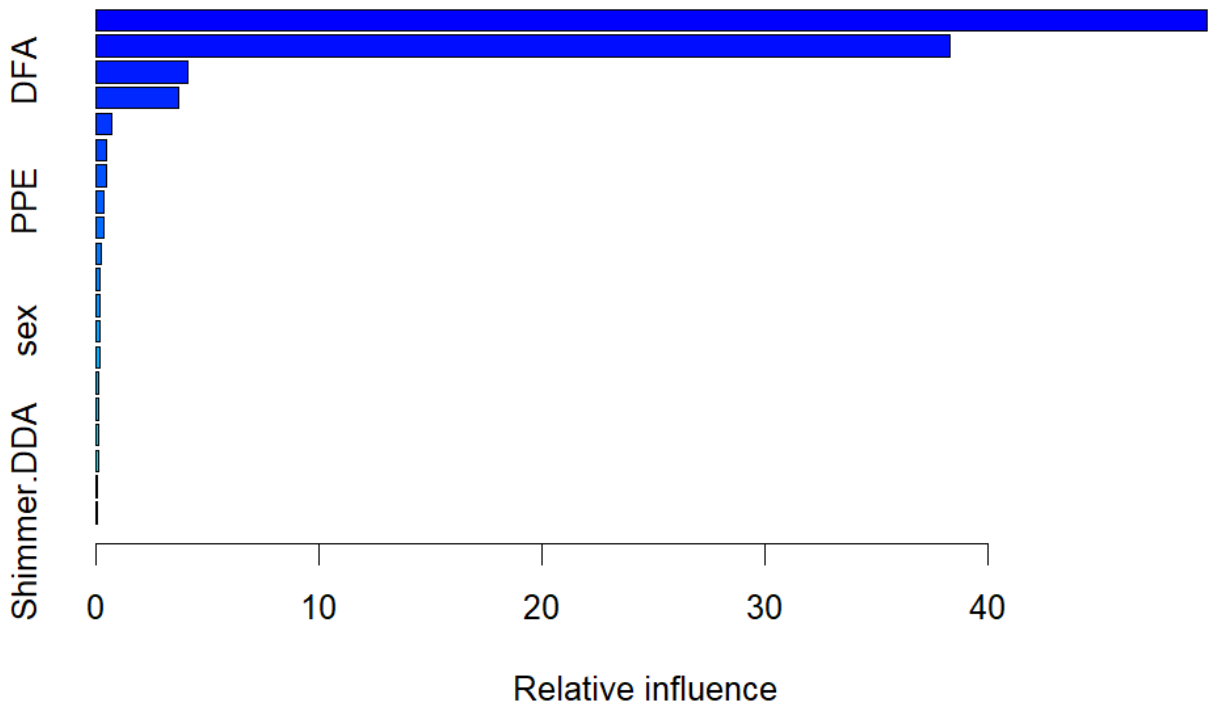
```
set.seed(1)
# Since this is a regression problem, we set the distribution to "gaussian"
# For binary classification, we would use "bernoulli"
# n.trees: number of trees we want
# interaction.depth: limits the depth of each tree
boost.park=gbm(total_UPDRS~.,data=training_data,distribution="gaussian",n.trees=5000, interac
tion.depth=4)

# In this case, summary() produces the relative influence plot and outputs
# the relative influence statistics
summary(boost.park)
```

	var <chr>	rel.inf <dbl>
subject.	subject.	49.81842382
age	age	38.28416168
DFA	DFA	4.14339575

	var <chr>	rel.inf <dbl>
test_time	test_time	3.69974374
HNR	HNR	0.73524494
Shimmer.APQ5	Shimmer.APQ5	0.49513030
RPDE	RPDE	0.45997826
PPE	PPE	0.36989422
Jitter.Abs.	Jitter.Abs.	0.33782891
Shimmer.APQ11	Shimmer.APQ11	0.26981946

1-10 of 20 rows Previous **1** 2 Next



Hide

```
#MSE Test
# Performance on the test set
yhat.boost=predict(boost.park,testing_data,n.trees=5000)
mean((yhat.boost -Y.test)^2)
```

```
[1] 0.8775685
```

Hide

```
# We can set the shrinkage parameter (i.e., lambda). Default value: 0.001
boost.park1=gbm(total_UPDRS~.,training_data,distribution= "gaussian",
                 n.trees=5000,interaction.depth=4,shrinkage=0.2,verbose=F)
yhat.boost1=predict(boost.park1,testing_data,n.trees=5000)

#MSE Test
# Performance on the test set
mean((yhat.boost1-Y.test)^2)
```

```
[1] 1.094136
```

Hide

```
#d) Compare and comment on the error obtained with each approach. Which model seems to perfor
m the best?

# Overall bagging performed better compared to boosting and random forests as it resulted in
a average value for MSE of 0.21 vs random forests 2.92 and boosting 0.88 which performed more
closely to bagging.
```

Hide

```
# 1. a) Decision Tree Classification
library(tree)

# Performance on the test set
# tree(): Fit a classification tree
# Similar syntax to lm()

tree.breast=tree(Class_Label~.,training_data1)

# summary() lists the variables that are used as internal nodes in the tree,
# the number of terminal nodes, and the (training) error rate
# A small deviance indicates a tree that provides a good fit to the (training) data
summary(tree.breast)
```

```
Classification tree:
tree(formula = Class_Label ~ ., data = training_data1)
Variables actually used in tree construction:
[1] "UCSize"          "Bare_Nuclei"     "Bland_Chromatin" "C_Thickness"
[5] "Normal_Nucleoli" "ID"              "M_Adhesion"
Number of terminal nodes:  11
Residual mean deviance:  0.1006 = 53.8 / 535
Misclassification error rate: 0.01832 = 10 / 546
```

Hide

```
# The object name prints split criterion, the number of observations
# in that branch, the deviance, the overall prediction for the branch, and the
# fraction of observations in that branch that take on values of Yes and No.
# Branches that lead to terminal nodes are indicated using asterisks
tree.breast
```

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 546 716.300 0 ( 0.635531 0.364469 )
   2) UCSize < 2.5 328  96.320 0 ( 0.966463 0.033537 )
     4) Bare_Nuclei < 3.5 308  13.460 0 ( 0.996753 0.003247 )
       8) Bland_Chromatin < 3.5 303   0.000 0 ( 1.000000 0.000000 ) *
       9) Bland_Chromatin > 3.5 5   5.004 0 ( 0.800000 0.200000 ) *
     5) Bare_Nuclei > 3.5 20  27.730 0 ( 0.500000 0.500000 )
      10) C_Thickness < 3.5 9   0.000 0 ( 1.000000 0.000000 ) *
      11) C_Thickness > 3.5 11   6.702 1 ( 0.090909 0.909091 ) *
   3) UCSize > 2.5 218 174.700 1 ( 0.137615 0.862385 )
     6) Bare_Nuclei < 3.5 54  74.560 1 ( 0.462963 0.537037 )
      12) UCSize < 4.5 33  36.550 0 ( 0.757576 0.242424 )
        24) Normal_Nucleoli < 1.5 15   0.000 0 ( 1.000000 0.000000 ) *
        25) Normal_Nucleoli > 1.5 18  24.730 0 ( 0.555556 0.444444 )
          50) ID < 1.17042e+06 9   9.535 1 ( 0.222222 0.777778 ) *
          51) ID > 1.17042e+06 9   6.279 0 ( 0.888889 0.111111 ) *
      13) UCSize > 4.5 21   0.000 1 ( 0.000000 1.000000 ) *
     7) Bare_Nuclei > 3.5 164  44.750 1 ( 0.030488 0.969512 )
      14) C_Thickness < 6.5 65  35.250 1 ( 0.076923 0.923077 )
        28) M_Adhesion < 5.5 28  26.280 1 ( 0.178571 0.821429 ) *
        29) M_Adhesion > 5.5 37   0.000 1 ( 0.000000 1.000000 ) *
      15) C_Thickness > 6.5 99   0.000 1 ( 0.000000 1.000000 ) *
```

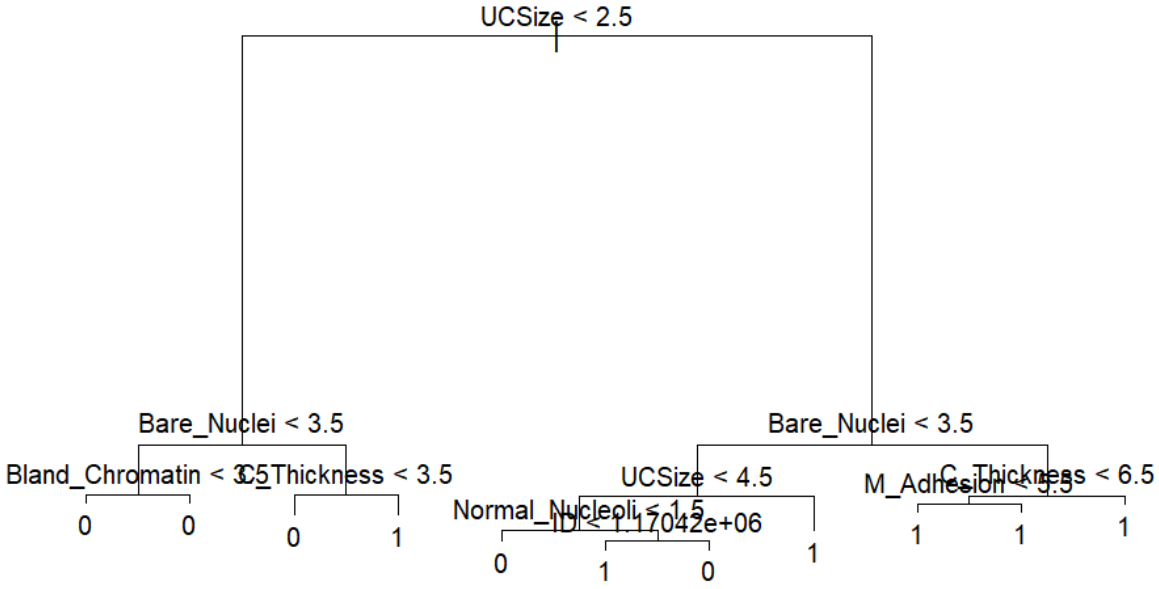
Hide

```
# Test on test set using predict()
# type="class" to return the class prediction
tree.pred=predict(tree.breast,testing_data1,type="class")
```

Hide

```
# plot() to display the tree structure
plot(tree.breast)

# text() to display the node labels
# pretty=0: include the category names for any qualitative predictors,
# rather than simply displaying a letter for each category
# cex for font size
text(tree.breast, pretty = 0, cex=0.75)
```



Hide

```
# Confusion matrix
conf.matrix <- table(tree.pred,Y.test1)
conf.matrix
```

```
         Y.test1
tree.pred  0  1
        0 91  1
        1  6 39
```

Hide

```
#i) Accuracy on test set
(conf.matrix[1,1] + conf.matrix[2,2])/200
```

```
[1] 0.65
```

Hide

```
##b) Tree pruning

# cv.tree(): CV to determine the optimal level of tree complexity
# FUN=prune.misclass: classification error rate to guide the CV and pruning process
# default: deviance
set.seed(3)

cv.breast=cv.tree(tree.breast,FUN=prune.misclass)

# Number of terminal nodes of each tree considered (size)
cv.breast$size
```

```
[1] 11  8  6  4  2  1
```

Hide

```
# Corresponding CV error rate
cv.breast$dev
```

```
[1]  19  19  28  29  35 199
```

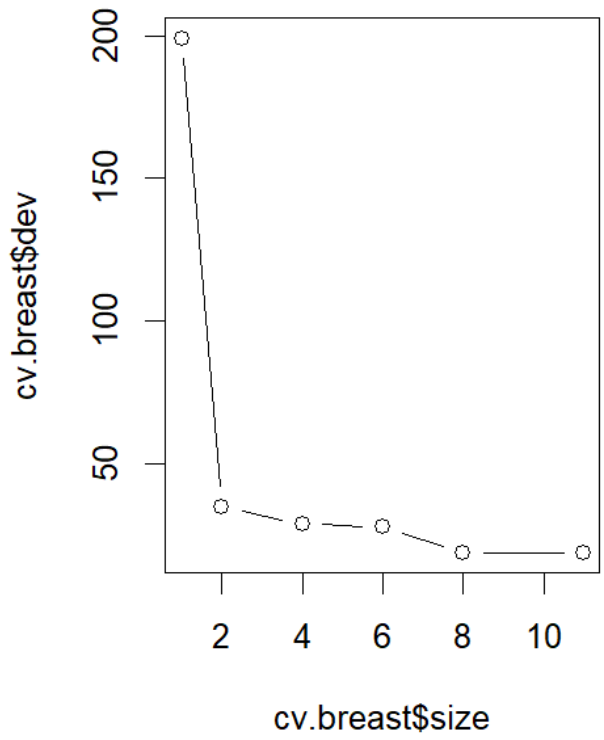
Hide

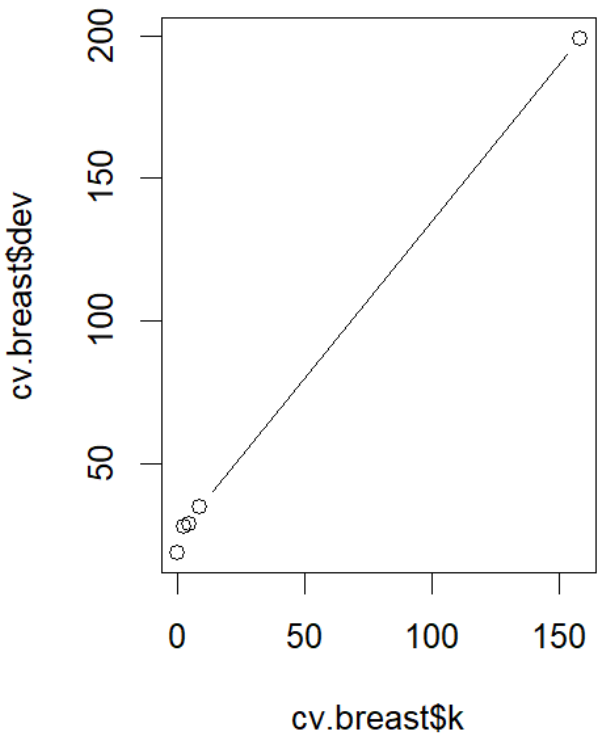
```
# Value of the cost-complexity parameter used (k here, alpha in formula - see 8.4)
cv.breast$k
```

```
[1]  -Inf   0.0   2.5   4.5   8.5 158.0
```

Hide

```
# Plot the error rate as a function of both size and k
par(mfrow=c(1,2))
plot(cv.breast$size, cv.breast$dev, type="b")
plot(cv.breast$k, cv.breast$dev, type="b")
```



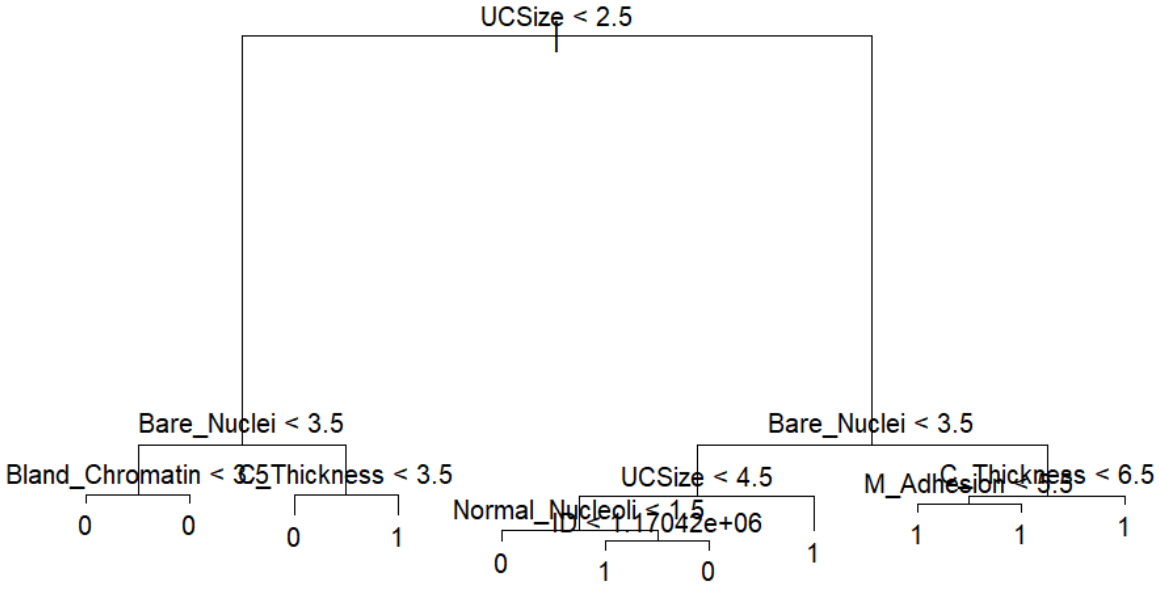


Hide

```
# prune.misclass(): prune the tree based on the CV error
prune.breast=prune.misclass(tree.breast,best=11)
```

Hide

```
# Plot resulting decision tree
plot(prune.breast)
text(prune.breast, pretty=0, cex=0.75)
```



Hide

```
# Let's check how well the pruned tree performs on the test set
tree.pred=predict(prune.breast,testing_data1,type="class")
conf.matrix1 <- table(tree.pred,Y.test1)
conf.matrix1
```

```
         Y.test1
tree.pred  0  1
        0 91  1
        1  6 39
```

Hide

```
#ii)Prune Accuracy
(conf.matrix1[1,1] + conf.matrix1[2,2])/200
```

```
[1] 0.65
```