Practical Machine Learning - Week3

Saul Lugo

January 22, 2016

Predicting with Trees

The basic algorithm for predicting with trees is the following:

1) Start with all the variables in one group
2) Find the variable/split that best separates the outcomes
3) Divide the data into two groups ("leaves") on that split
4) Within each split, find the variable that best separates the outcome
5) Continue until the groups are too small or sufficiently pure

Example with the Iris dataset

```r
data(iris)
library(ggplot2)
library(caret)
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

```r
table(iris$Species)
```

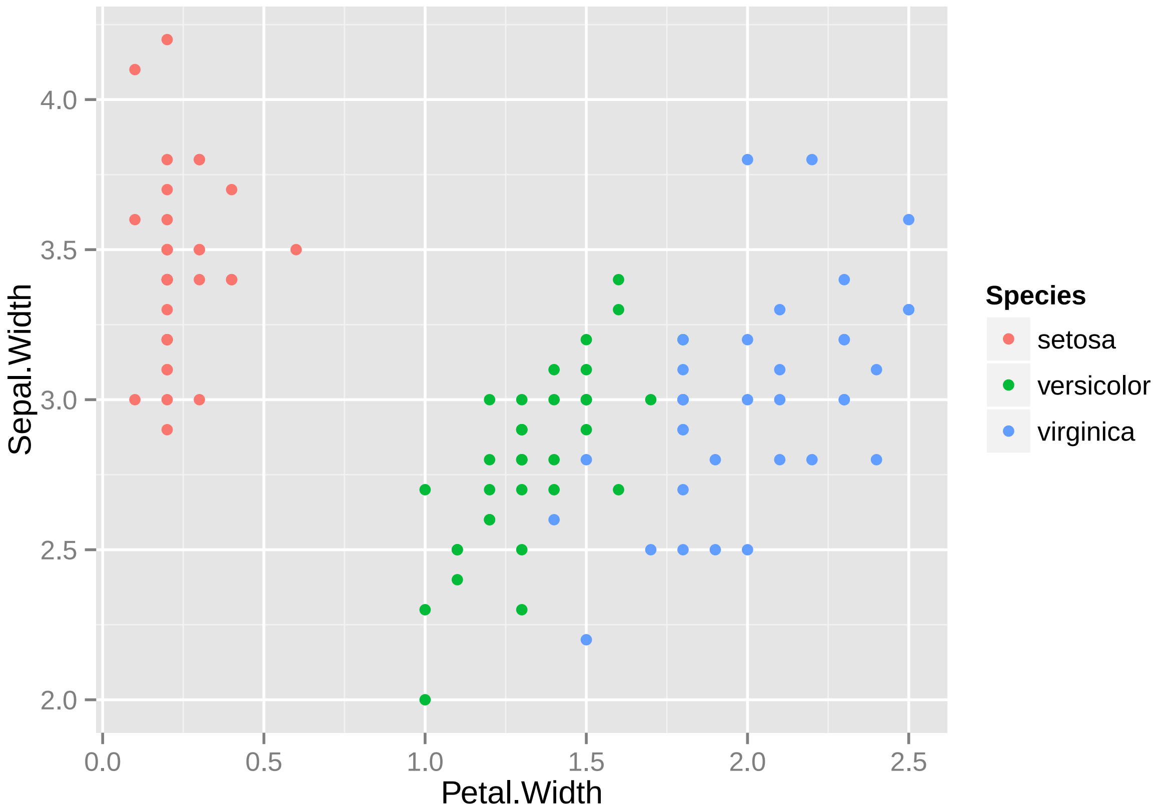
```
##
##     setosa versicolor  virginica
##         50         50         50
```

```r
#Create the data partition
inTrain <- createDataPartition(y=iris$Species,p=0.7,list=FALSE)
training <- iris[inTrain,]
testing <- iris[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 105   5
```

```
## [1] 45  5
```

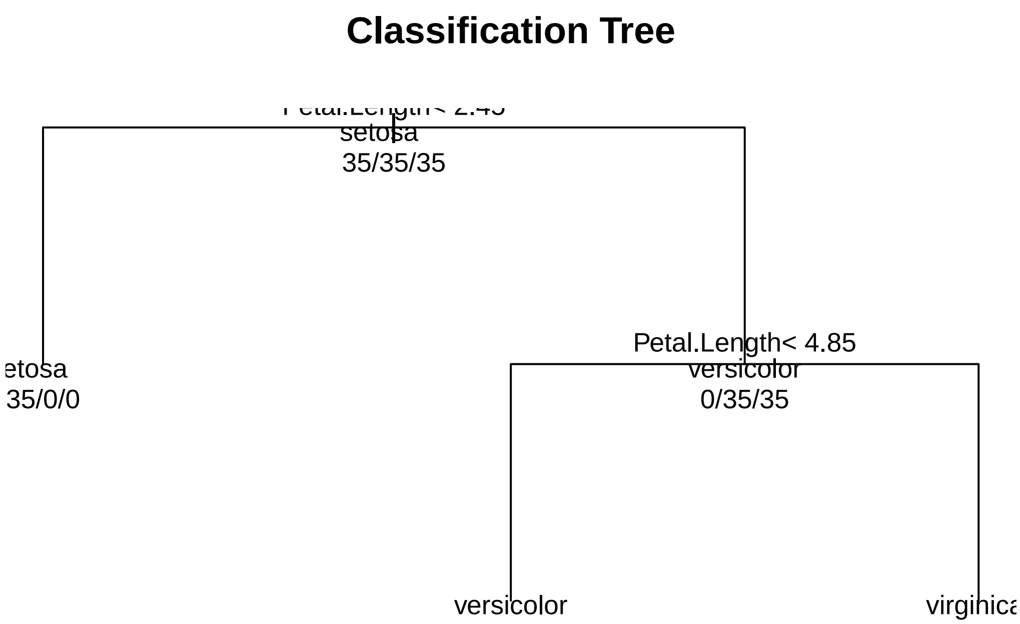
```r
#Exploring the data
qplot(Petal.Width,Sepal.Width,colour=Species,data=training)
```



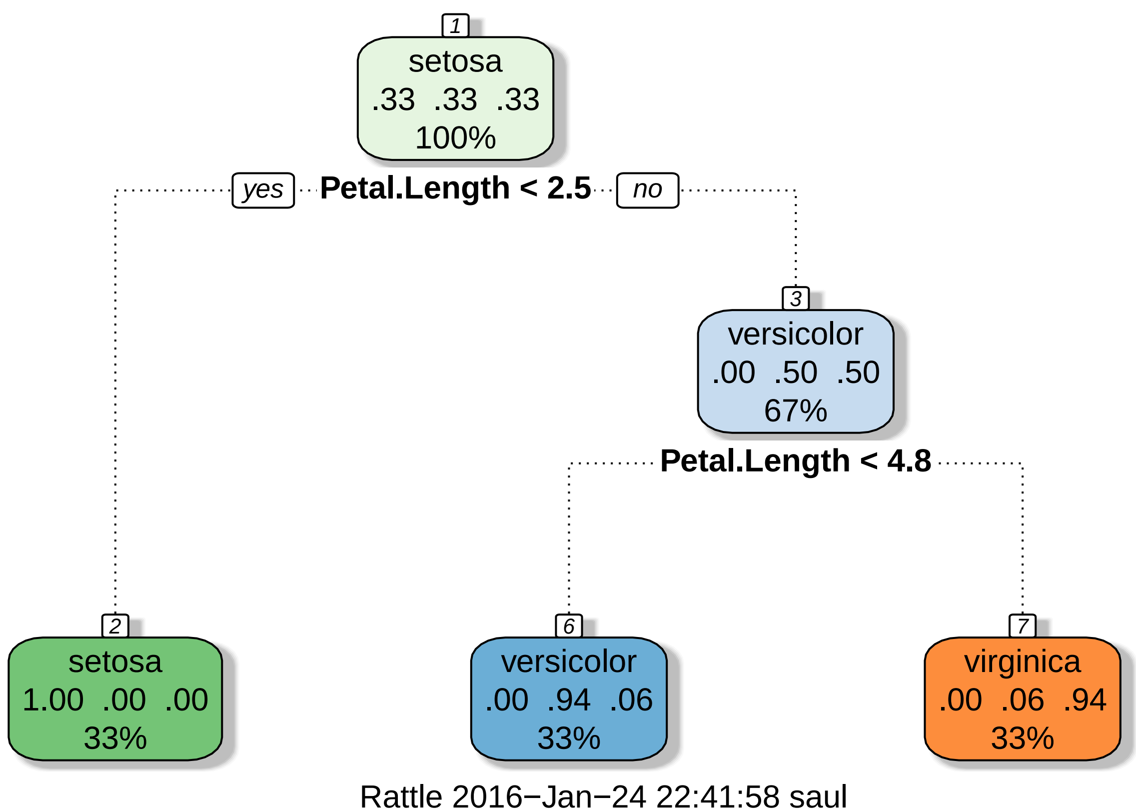

```
#Training the model (a tree model)
modFit <- train(Species ~ ., method="rpart", data = training)
modFit_PredTree <- modFit
print(modFit$finalModel)
```

```
## n= 105
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 105 70 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 35  0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 70 35 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Length< 4.85 35  2 versicolor (0.00000000 0.94285714 0.05714286) *
##     7) Petal.Length>=4.85 35  2 virginica (0.00000000 0.05714286 0.94285714) *
```

```
#Plotting the model
plot(modFit$finalModel, uniform=TRUE,main="Classification Tree")
text(modFit$finalModel, use.n=TRUE, all=TRUE, cex=0.8)
```




```
##A better plot
library(rattle)
fancyRpartPlot(modFit$finalModel)
```




```
#Now predicting the outcome in the testing set
prediction <- predict(modFit,newdata=testing)
prediction
```

```
##  [1] setosa     setosa     setosa     setosa     setosa     setosa    
##  [7] setosa     setosa     setosa     setosa     setosa     setosa    
## [13] setosa     setosa     setosa     versicolor virginica  versicolor
## [19] versicolor versicolor versicolor versicolor versicolor versicolor
## [25] virginica  versicolor versicolor versicolor versicolor versicolor
## [31] virginica  virginica  virginica  virginica  virginica  virginica 
## [37] virginica  virginica  virginica  versicolor virginica  virginica 
## [43] virginica  virginica  virginica 
## Levels: setosa versicolor virginica
```

```r
CM_PredTree <- confusionMatrix(testing$Species,prediction)
CM_PredTree
```

```
## Confusion Matrix and Statistics
## 
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         2
##   virginica       0          1        14
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9333          
##                  95% CI : (0.8173, 0.986)
##     No Information Rate : 0.3556          
##     P-Value [Acc > NIR] : 5.426e-16       
##                                           
##                   Kappa : 0.9             
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9286           0.8750
## Specificity                 1.0000            0.9355           0.9655
## Pos Pred Value              1.0000            0.8667           0.9333
## Neg Pred Value              1.0000            0.9667           0.9333
## Prevalence                  0.3333            0.3111           0.3556
## Detection Rate              0.3333            0.2889           0.3111
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9320           0.9203
```

Notes on predicting with trees

1) Classification trees are non-linear models - They use interactions between variables - Data transformation might be less important - Trees can also be use for regression problems (continious outcome)
2) There are several options for building trees in R: - party, rpart in the caret package - tree, out of the carect package

Bagging with the Caret Package

Bagging stands for **Bootstrap Aggregating**.

The following is an example of Bagging using the caret package:

```
library(ElemStatLearn); data(ozone,package="ElemStatLearn")
library(caret)
head(ozone)
```

```
##   ozone radiation temperature wind
## 1    41       190          67  7.4
## 2    36       118          72  8.0
## 3    12       149          74 12.6
## 4    18       313          62 11.5
## 5    23       299          65  8.6
## 6    19        99          59 13.8
```

```
predictors = data.frame(ozone=ozone$ozone)
temperature = ozone$temperature
treebag <- bag(predictors,temperature,B=10,bagControl = bagControl(fit = ctreeBag$fit, predict = ctreeBa
treebag$fit
```

```
## [[1]]
## [[1]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 46; criterion = 1, statistic = 46.25
##   2) ozone <= 19; criterion = 0.999, statistic = 10.356
##     3)*  weights = 27
##   2) ozone > 19
##     4)*  weights = 47
## 1) ozone > 46
##   5)*  weights = 37
##
## [[1]]$vars
## [1] 1
##
## [[1]]$oob
##         pred obs        key
## 1  76.68085  67 yjqgxypvfd
## 2  70.14815  62 yjqgxypvfd
## 3  70.14815  61 yjqgxypvfd
## 4  70.14815  69 yjqgxypvfd
## 5  70.14815  66 yjqgxypvfd
## 6  70.14815  58 yjqgxypvfd
## 7  76.68085  66 yjqgxypvfd
## 8  76.68085  68 yjqgxypvfd
```

```
## 9  76.68085  81 yjqgxypvfd
## 10 76.68085  76 yjqgxypvfd
## 11 87.48649  90 yjqgxypvfd
## 12 76.68085  87 yjqgxypvfd
## 13 76.68085  82 yjqgxypvfd
## 14 76.68085  77 yjqgxypvfd
## 15 76.68085  72 yjqgxypvfd
## 16 76.68085  65 yjqgxypvfd
## 17 70.14815  73 yjqgxypvfd
## 18 70.14815  76 yjqgxypvfd
## 19 87.48649  85 yjqgxypvfd
## 20 87.48649  83 yjqgxypvfd
## 21 87.48649  89 yjqgxypvfd
## 22 76.68085  81 yjqgxypvfd
## 23 87.48649  81 yjqgxypvfd
## 24 87.48649  86 yjqgxypvfd
## 25 87.48649  81 yjqgxypvfd
## 26 70.14815  81 yjqgxypvfd
## 27 70.14815  82 yjqgxypvfd
## 28 76.68085  86 yjqgxypvfd
## 29 87.48649  80 yjqgxypvfd
## 30 76.68085  77 yjqgxypvfd
## 31 70.14815  72 yjqgxypvfd
## 32 87.48649  96 yjqgxypvfd
## 33 87.48649  91 yjqgxypvfd
## 34 87.48649  92 yjqgxypvfd
## 35 87.48649  93 yjqgxypvfd
## 36 70.14815  67 yjqgxypvfd
## 37 70.14815  82 yjqgxypvfd
## 38 70.14815  64 yjqgxypvfd
## 39 76.68085  71 yjqgxypvfd
## 40 76.68085  70 yjqgxypvfd
## 41 70.14815  76 yjqgxypvfd
## 42 76.68085  68 yjqgxypvfd
##
##
## [[2]]
## [[2]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 65; criterion = 1, statistic = 70.791
##   2) ozone <= 30; criterion = 1, statistic = 25.262
##     3)*  weights = 52
##   2) ozone > 30
##     4)*  weights = 31
## 1) ozone > 65
##   5)*  weights = 28
##
## [[2]]$vars
```

```
## [1] 1
##
## [[2]]$oob
##        pred obs        key
## 1  79.38710  72 dxfxqxfcfi
## 2  71.82692  65 dxfxqxfcfi
## 3  71.82692  59 dxfxqxfcfi
## 4  71.82692  69 dxfxqxfcfi
## 5  71.82692  68 dxfxqxfcfi
## 6  71.82692  58 dxfxqxfcfi
## 7  79.38710  66 dxfxqxfcfi
## 8  71.82692  57 dxfxqxfcfi
## 9  71.82692  62 dxfxqxfcfi
## 10 90.67857  79 dxfxqxfcfi
## 11 79.38710  76 dxfxqxfcfi
## 12 71.82692  82 dxfxqxfcfi
## 13 79.38710  85 dxfxqxfcfi
## 14 79.38710  83 dxfxqxfcfi
## 15 79.38710  83 dxfxqxfcfi
## 16 71.82692  73 dxfxqxfcfi
## 17 71.82692  81 dxfxqxfcfi
## 18 79.38710  81 dxfxqxfcfi
## 19 79.38710  84 dxfxqxfcfi
## 20 79.38710  85 dxfxqxfcfi
## 21 90.67857  86 dxfxqxfcfi
## 22 90.67857  85 dxfxqxfcfi
## 23 79.38710  86 dxfxqxfcfi
## 24 79.38710  81 dxfxqxfcfi
## 25 71.82692  81 dxfxqxfcfi
## 26 71.82692  82 dxfxqxfcfi
## 27 90.67857  89 dxfxqxfcfi
## 28 90.67857  90 dxfxqxfcfi
## 29 71.82692  82 dxfxqxfcfi
## 30 71.82692  77 dxfxqxfcfi
## 31 71.82692  72 dxfxqxfcfi
## 32 90.67857  81 dxfxqxfcfi
## 33 79.38710  87 dxfxqxfcfi
## 34 71.82692  76 dxfxqxfcfi
## 35 71.82692  82 dxfxqxfcfi
## 36 71.82692  71 dxfxqxfcfi
## 37 71.82692  63 dxfxqxfcfi
## 38 71.82692  68 dxfxqxfcfi
##
##
## [[3]]
## [[3]]$fit
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 37; criterion = 1, statistic = 54.42
```

```
##   2) ozone <= 20; criterion = 1, statistic = 15.73
##     3)*  weights = 33
##   2) ozone > 20
##     4)*  weights = 30
## 1) ozone > 37
##   5) ozone <= 65; criterion = 0.957, statistic = 4.077
##     6)*  weights = 23
##   5) ozone > 65
##     7) ozone <= 110; criterion = 0.975, statistic = 5.046
##       8)*  weights = 17
##     7) ozone > 110
##       9)*  weights = 8
##
## [[3]]$vars
## [1] 1
##
## [[3]]$oob
##        pred obs        key
## 1  82.78261  67 shxfortrzy
## 2  68.90909  74 shxfortrzy
## 3  68.90909  62 shxfortrzy
## 4  68.90909  59 shxfortrzy
## 5  68.90909  61 shxfortrzy
## 6  68.90909  69 shxfortrzy
## 7  68.90909  64 shxfortrzy
## 8  76.03333  66 shxfortrzy
## 9  68.90909  57 shxfortrzy
## 10 76.03333  61 shxfortrzy
## 11 68.90909  76 shxfortrzy
## 12 76.03333  81 shxfortrzy
## 13 82.78261  83 shxfortrzy
## 14 82.78261  83 shxfortrzy
## 15 90.17647  92 shxfortrzy
## 16 68.90909  74 shxfortrzy
## 17 90.17647  85 shxfortrzy
## 18 68.90909  82 shxfortrzy
## 19 82.78261  86 shxfortrzy
## 20 82.78261  81 shxfortrzy
## 21 68.90909  82 shxfortrzy
## 22 90.17647  90 shxfortrzy
## 23 68.90909  72 shxfortrzy
## 24 90.17647  94 shxfortrzy
## 25 90.17647  91 shxfortrzy
## 26 90.17647  92 shxfortrzy
## 27 82.78261  81 shxfortrzy
## 28 76.03333  76 shxfortrzy
## 29 68.90909  67 shxfortrzy
## 30 76.03333  68 shxfortrzy
## 31 68.90909  82 shxfortrzy
## 32 68.90909  64 shxfortrzy
## 33 76.03333  71 shxfortrzy
## 34 68.90909  75 shxfortrzy
##
##
```

```
## [[4]]
## [[4]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 46; criterion = 1, statistic = 43.289
##   2) ozone <= 24; criterion = 0.996, statistic = 8.097
##     3)*  weights = 50
##   2) ozone > 24
##     4)*  weights = 27
## 1) ozone > 46
##   5)*  weights = 34
##
## [[4]]$vars
## [1] 1
##
## [[4]]$oob
##        pred obs        key
## 1  71.16000  61 tlxnxzilmf
## 2  71.16000  66 tlxnxzilmf
## 3  71.16000  64 tlxnxzilmf
## 4  71.16000  61 tlxnxzilmf
## 5  77.22222  76 tlxnxzilmf
## 6  77.22222  82 tlxnxzilmf
## 7  71.16000  82 tlxnxzilmf
## 8  71.16000  77 tlxnxzilmf
## 9  77.22222  72 tlxnxzilmf
## 10 86.61765  85 tlxnxzilmf
## 11 86.61765  83 tlxnxzilmf
## 12 86.61765  92 tlxnxzilmf
## 13 86.61765  92 tlxnxzilmf
## 14 77.22222  82 tlxnxzilmf
## 15 86.61765  87 tlxnxzilmf
## 16 86.61765  85 tlxnxzilmf
## 17 71.16000  74 tlxnxzilmf
## 18 86.61765  86 tlxnxzilmf
## 19 71.16000  82 tlxnxzilmf
## 20 86.61765  88 tlxnxzilmf
## 21 86.61765  89 tlxnxzilmf
## 22 86.61765  90 tlxnxzilmf
## 23 77.22222  86 tlxnxzilmf
## 24 71.16000  77 tlxnxzilmf
## 25 71.16000  76 tlxnxzilmf
## 26 71.16000  77 tlxnxzilmf
## 27 71.16000  72 tlxnxzilmf
## 28 86.61765  86 tlxnxzilmf
## 29 86.61765  97 tlxnxzilmf
## 30 86.61765  92 tlxnxzilmf
## 31 86.61765  93 tlxnxzilmf
## 32 86.61765  87 tlxnxzilmf
```

```
## 33 71.16000  80 tlxnxzilmf
## 34 71.16000  78 tlxnxzilmf
## 35 71.16000  75 tlxnxzilmf
## 36 77.22222  81 tlxnxzilmf
## 37 71.16000  76 tlxnxzilmf
## 38 71.16000  71 tlxnxzilmf
## 39 77.22222  81 tlxnxzilmf
##
##
## [[5]]
## [[5]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 37; criterion = 1, statistic = 45.114
##   2) ozone <= 20; criterion = 0.964, statistic = 4.407
##     3)*  weights = 35
##   2) ozone > 20
##     4)*  weights = 28
## 1) ozone > 37
##   5)*  weights = 48
##
## [[5]]$vars
## [1] 1
##
## [[5]]$oob
##        pred obs        key
## 1  86.35417  67 ppoxfdqegj
## 2  74.14286  72 ppoxfdqegj
## 3  68.34286  74 ppoxfdqegj
## 4  68.34286  58 ppoxfdqegj
## 5  68.34286  64 ppoxfdqegj
## 6  68.34286  57 ppoxfdqegj
## 7  68.34286  62 ppoxfdqegj
## 8  68.34286  73 ppoxfdqegj
## 9  86.35417  81 ppoxfdqegj
## 10 74.14286  76 ppoxfdqegj
## 11 86.35417  90 ppoxfdqegj
## 12 68.34286  65 ppoxfdqegj
## 13 68.34286  76 ppoxfdqegj
## 14 86.35417  81 ppoxfdqegj
## 15 74.14286  82 ppoxfdqegj
## 16 68.34286  82 ppoxfdqegj
## 17 86.35417  86 ppoxfdqegj
## 18 86.35417  86 ppoxfdqegj
## 19 86.35417  83 ppoxfdqegj
## 20 68.34286  82 ppoxfdqegj
## 21 86.35417  90 ppoxfdqegj
## 22 86.35417  86 ppoxfdqegj
## 23 86.35417  79 ppoxfdqegj
```

```
## 24 86.35417  78 ppoxfdqegj
## 25 68.34286  72 ppoxfdqegj
## 26 86.35417  86 ppoxfdqegj
## 27 86.35417  94 ppoxfdqegj
## 28 86.35417  96 ppoxfdqegj
## 29 86.35417  91 ppoxfdqegj
## 30 86.35417  92 ppoxfdqegj
## 31 74.14286  84 ppoxfdqegj
## 32 68.34286  80 ppoxfdqegj
## 33 74.14286  78 ppoxfdqegj
## 34 74.14286  73 ppoxfdqegj
## 35 86.35417  81 ppoxfdqegj
## 36 74.14286  76 ppoxfdqegj
## 37 68.34286  67 ppoxfdqegj
## 38 74.14286  68 ppoxfdqegj
## 39 68.34286  82 ppoxfdqegj
## 40 74.14286  71 ppoxfdqegj
## 41 74.14286  70 ppoxfdqegj
## 42 68.34286  76 ppoxfdqegj
##
##
## [[6]]
## [[6]]$fit
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 41; criterion = 1, statistic = 58.444
##   2) ozone <= 7; criterion = 0.989, statistic = 6.472
##     3)*  weights = 8
##   2) ozone > 7
##     4)*  weights = 60
## 1) ozone > 41
##   5) ozone <= 64; criterion = 0.988, statistic = 6.327
##     6)*  weights = 18
##   5) ozone > 64
##     7) ozone <= 89; criterion = 0.973, statistic = 4.909
##       8)*  weights = 18
##     7) ozone > 89
##       9)*  weights = 7
##
## [[6]]$vars
## [1] 1
##
## [[6]]$oob
##        pred obs        key
## 1  72.41667  61 zmikpzroyt
## 2  72.41667  66 zmikpzroyt
## 3  63.50000  57 zmikpzroyt
## 4  72.41667  73 zmikpzroyt
## 5  82.50000  81 zmikpzroyt
```

```
## 6  72.41667  87 zmikpzroyt
## 7  72.41667  77 zmikpzroyt
## 8  72.41667  72 zmikpzroyt
## 9  88.14286  84 zmikpzroyt
## 10 72.41667  81 zmikpzroyt
## 11 90.11111  88 zmikpzroyt
## 12 88.14286  92 zmikpzroyt
## 13 72.41667  73 zmikpzroyt
## 14 63.50000  80 zmikpzroyt
## 15 82.50000  81 zmikpzroyt
## 16 82.50000  84 zmikpzroyt
## 17 90.11111  87 zmikpzroyt
## 18 90.11111  86 zmikpzroyt
## 19 88.14286  85 zmikpzroyt
## 20 72.41667  82 zmikpzroyt
## 21 82.50000  86 zmikpzroyt
## 22 82.50000  83 zmikpzroyt
## 23 72.41667  81 zmikpzroyt
## 24 88.14286  90 zmikpzroyt
## 25 72.41667  82 zmikpzroyt
## 26 90.11111  80 zmikpzroyt
## 27 72.41667  77 zmikpzroyt
## 28 82.50000  79 zmikpzroyt
## 29 72.41667  76 zmikpzroyt
## 30 90.11111  97 zmikpzroyt
## 31 90.11111  92 zmikpzroyt
## 32 88.14286  93 zmikpzroyt
## 33 72.41667  84 zmikpzroyt
## 34 72.41667  78 zmikpzroyt
## 35 72.41667  73 zmikpzroyt
## 36 72.41667  77 zmikpzroyt
## 37 72.41667  71 zmikpzroyt
## 38 82.50000  78 zmikpzroyt
## 39 72.41667  67 zmikpzroyt
## 40 72.41667  68 zmikpzroyt
## 41 72.41667  71 zmikpzroyt
## 42 72.41667  81 zmikpzroyt
## 43 72.41667  63 zmikpzroyt
## 44 72.41667  75 zmikpzroyt
## 45 72.41667  76 zmikpzroyt
##
##
## [[7]]
## [[7]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 41; criterion = 1, statistic = 49.074
##   2) ozone <= 18; criterion = 0.997, statistic = 8.907
##     3)*  weights = 31
```

```
##   2) ozone > 18
##     4)*  weights = 38
## 1) ozone > 41
##   5)*  weights = 42
##
## [[7]]$vars
## [1] 1
##
## [[7]]$oob
##        pred obs        key
## 1  75.28947  72 iogyeetcrf
## 2  75.28947  59 iogyeetcrf
## 3  69.58065  61 iogyeetcrf
## 4  69.58065  69 iogyeetcrf
## 5  69.58065  64 iogyeetcrf
## 6  75.28947  68 iogyeetcrf
## 7  69.58065  62 iogyeetcrf
## 8  69.58065  61 iogyeetcrf
## 9  87.26190  79 iogyeetcrf
## 10 75.28947  82 iogyeetcrf
## 11 75.28947  65 iogyeetcrf
## 12 69.58065  73 iogyeetcrf
## 13 75.28947  81 iogyeetcrf
## 14 87.26190  83 iogyeetcrf
## 15 87.26190  88 iogyeetcrf
## 16 87.26190  92 iogyeetcrf
## 17 69.58065  73 iogyeetcrf
## 18 75.28947  81 iogyeetcrf
## 19 75.28947  82 iogyeetcrf
## 20 87.26190  87 iogyeetcrf
## 21 87.26190  85 iogyeetcrf
## 22 75.28947  82 iogyeetcrf
## 23 87.26190  86 iogyeetcrf
## 24 87.26190  88 iogyeetcrf
## 25 87.26190  89 iogyeetcrf
## 26 87.26190  90 iogyeetcrf
## 27 87.26190  78 iogyeetcrf
## 28 69.58065  72 iogyeetcrf
## 29 87.26190  86 iogyeetcrf
## 30 87.26190  97 iogyeetcrf
## 31 87.26190  92 iogyeetcrf
## 32 87.26190  93 iogyeetcrf
## 33 75.28947  80 iogyeetcrf
## 34 75.28947  73 iogyeetcrf
## 35 87.26190  81 iogyeetcrf
## 36 75.28947  77 iogyeetcrf
## 37 69.58065  71 iogyeetcrf
## 38 87.26190  78 iogyeetcrf
## 39 69.58065  67 iogyeetcrf
## 40 69.58065  64 iogyeetcrf
## 41 75.28947  71 iogyeetcrf
## 42 69.58065  69 iogyeetcrf
## 43 75.28947  70 iogyeetcrf
## 44 75.28947  68 iogyeetcrf
```

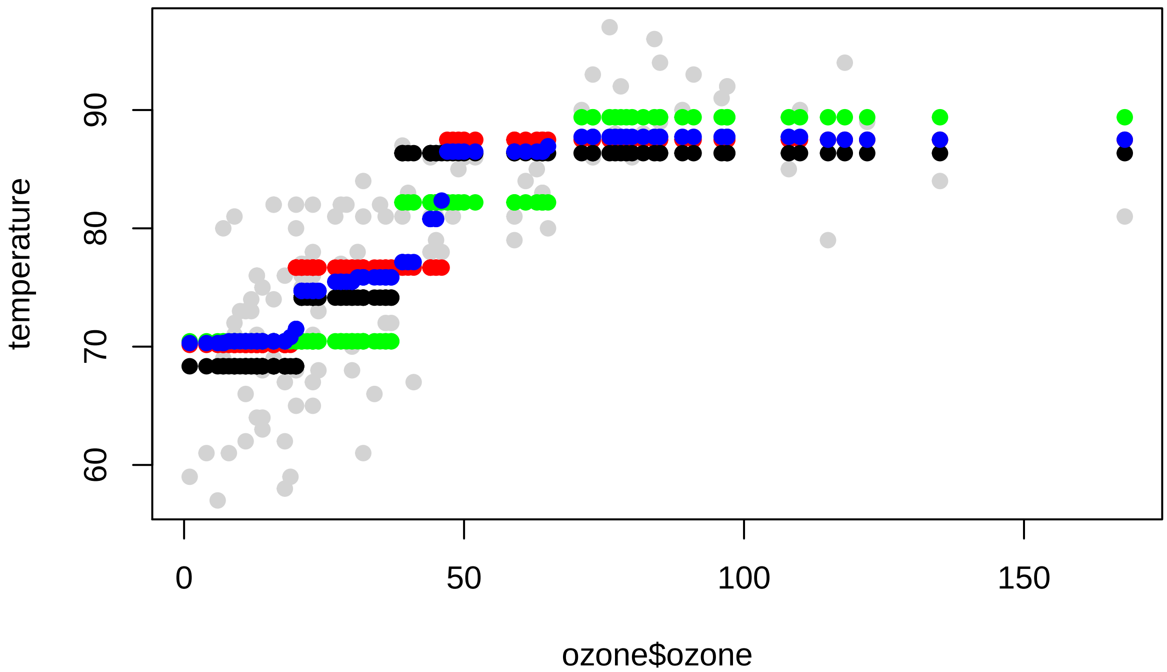
```
##
##
## [[8]]
## [[8]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 46; criterion = 1, statistic = 47.438
##   2) ozone <= 14; criterion = 0.995, statistic = 8.039
##     3)*  weights = 23
##   2) ozone > 14
##     4)*  weights = 62
## 1) ozone > 46
##   5)*  weights = 26
##
## [[8]]$vars
## [1] 1
##
## [[8]]$oob
##        pred obs        key
## 1  70.47826  74 ibegordtbe
## 2  75.67742  62 ibegordtbe
## 3  75.67742  58 ibegordtbe
## 4  70.47826  64 ibegordtbe
## 5  75.67742  68 ibegordtbe
## 6  70.47826  59 ibegordtbe
## 7  70.47826  73 ibegordtbe
## 8  87.50000  79 ibegordtbe
## 9  87.50000  90 ibegordtbe
## 10 75.67742  82 ibegordtbe
## 11 75.67742  77 ibegordtbe
## 12 75.67742  65 ibegordtbe
## 13 75.67742  81 ibegordtbe
## 14 87.50000  88 ibegordtbe
## 15 87.50000  92 ibegordtbe
## 16 87.50000  89 ibegordtbe
## 17 70.47826  73 ibegordtbe
## 18 87.50000  87 ibegordtbe
## 19 75.67742  74 ibegordtbe
## 20 87.50000  86 ibegordtbe
## 21 87.50000  86 ibegordtbe
## 22 87.50000  83 ibegordtbe
## 23 87.50000  81 ibegordtbe
## 24 87.50000  89 ibegordtbe
## 25 75.67742  86 ibegordtbe
## 26 75.67742  82 ibegordtbe
## 27 75.67742  78 ibegordtbe
## 28 70.47826  72 ibegordtbe
## 29 87.50000  81 ibegordtbe
## 30 87.50000  86 ibegordtbe
```

```
## 31 87.50000  94 ibegordtbe
## 32 87.50000  94 ibegordtbe
## 33 87.50000  91 ibegordtbe
## 34 87.50000  93 ibegordtbe
## 35 87.50000  87 ibegordtbe
## 36 75.67742  78 ibegordtbe
## 37 70.47826  76 ibegordtbe
## 38 75.67742  71 ibegordtbe
## 39 75.67742  81 ibegordtbe
## 40 75.67742  70 ibegordtbe
##
##
## [[9]]
## [[9]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 37; criterion = 1, statistic = 61.733
##   2)*  weights = 62
## 1) ozone > 37
##   3) ozone <= 65; criterion = 1, statistic = 13.63
##     4)*  weights = 26
##   3) ozone > 65
##     5)*  weights = 23
##
## [[9]]$vars
## [1] 1
##
## [[9]]$oob
##        pred obs        key
## 1  82.19231  67 stpcxtuojz
## 2  70.45161  72 stpcxtuojz
## 3  70.45161  74 stpcxtuojz
## 4  70.45161  61 stpcxtuojz
## 5  70.45161  57 stpcxtuojz
## 6  70.45161  73 stpcxtuojz
## 7  70.45161  61 stpcxtuojz
## 8  89.39130  79 stpcxtuojz
## 9  70.45161  76 stpcxtuojz
## 10 89.39130  90 stpcxtuojz
## 11 82.19231  87 stpcxtuojz
## 12 70.45161  77 stpcxtuojz
## 13 70.45161  65 stpcxtuojz
## 14 70.45161  73 stpcxtuojz
## 15 70.45161  76 stpcxtuojz
## 16 82.19231  85 stpcxtuojz
## 17 70.45161  81 stpcxtuojz
## 18 89.39130  88 stpcxtuojz
## 19 89.39130  92 stpcxtuojz
## 20 70.45161  73 stpcxtuojz
```

```
## 21 70.45161  81 stpcxtuojz
## 22 70.45161  80 stpcxtuojz
## 23 70.45161  82 stpcxtuojz
## 24 70.45161  82 stpcxtuojz
## 25 82.19231  86 stpcxtuojz
## 26 82.19231  81 stpcxtuojz
## 27 70.45161  82 stpcxtuojz
## 28 89.39130  89 stpcxtuojz
## 29 70.45161  82 stpcxtuojz
## 30 70.45161  72 stpcxtuojz
## 31 89.39130  81 stpcxtuojz
## 32 89.39130  96 stpcxtuojz
## 33 89.39130  94 stpcxtuojz
## 34 89.39130  91 stpcxtuojz
## 35 70.45161  76 stpcxtuojz
## 36 70.45161  77 stpcxtuojz
## 37 70.45161  71 stpcxtuojz
## 38 70.45161  82 stpcxtuojz
## 39 70.45161  71 stpcxtuojz
## 40 70.45161  81 stpcxtuojz
## 41 70.45161  68 stpcxtuojz
##
##
## [[10]]
## [[10]]$fit
##
##   Conditional inference tree with 3 terminal nodes
##
## Response:  y
## Input:  ozone
## Number of observations:  111
##
## 1) ozone <= 45; criterion = 1, statistic = 48.188
##   2) ozone <= 20; criterion = 0.993, statistic = 7.307
##     3)*  weights = 35
##   2) ozone > 20
##     4)*  weights = 43
## 1) ozone > 45
##   5)*  weights = 33
##
## [[10]]$vars
## [1] 1
##
## [[10]]$oob
##        pred obs        key
## 1  77.06977  72 grucogeveh
## 2  70.45714  74 grucogeveh
## 3  77.06977  65 grucogeveh
## 4  70.45714  61 grucogeveh
## 5  70.45714  68 grucogeveh
## 6  70.45714  57 grucogeveh
## 7  70.45714  62 grucogeveh
## 8  70.45714  59 grucogeveh
## 9  77.06977  67 grucogeveh
```

```
## 10 77.06977  81 grucogeveh
## 11 87.96970  90 grucogeveh
## 12 77.06977  87 grucogeveh
## 13 87.96970  83 grucogeveh
## 14 77.06977  83 grucogeveh
## 15 87.96970  89 grucogeveh
## 16 87.96970  81 grucogeveh
## 17 77.06977  82 grucogeveh
## 18 87.96970  84 grucogeveh
## 19 87.96970  87 grucogeveh
## 20 87.96970  86 grucogeveh
## 21 70.45714  82 grucogeveh
## 22 87.96970  86 grucogeveh
## 23 87.96970  88 grucogeveh
## 24 70.45714  82 grucogeveh
## 25 87.96970  89 grucogeveh
## 26 87.96970  90 grucogeveh
## 27 77.06977  86 grucogeveh
## 28 77.06977  76 grucogeveh
## 29 77.06977  78 grucogeveh
## 30 70.45714  72 grucogeveh
## 31 87.96970  86 grucogeveh
## 32 87.96970  94 grucogeveh
## 33 77.06977  78 grucogeveh
## 34 77.06977  81 grucogeveh
## 35 87.96970  78 grucogeveh
## 36 77.06977  71 grucogeveh
## 37 77.06977  81 grucogeveh
## 38 70.45714  69 grucogeveh
## 39 70.45714  63 grucogeveh
## 40 77.06977  70 grucogeveh
## 41 70.45714  75 grucogeveh
## 42 70.45714  76 grucogeveh
```

```r
plot(ozone$ozone, temperature, col="lightgrey", pch=19)
#Plotting the prediction 1 of the treebag
points(ozone$ozone, predict(treebag$fits[[1]]$fit,predictors),pch=19,col="red")
#Plotting the prediction 9 of the treebag
points(ozone$ozone, predict(treebag$fits[[9]]$fit,predictors),pch=19,col="green")
#Plotting the prediction 5 of the treebag
points(ozone$ozone, predict(treebag$fits[[5]]$fit,predictors),pch=19,col="black")
#Plotting the aggregated predictions of the treebag
points(ozone$ozone, predict(treebag,predictors),pch=19, col="blue")
```



We can see that the aggregated prediction (the blue dots in the plot) is the closest to the real values of temperature.

Also we can verify that the aggregated prediction has the lowest **RSME** (Root Mean Squared Error):

```
#The following is the RSME of the aggregated prediction:
sqrt(sum((ozone$temperature-predict(treebag,predictors))^2)/111)
```

```
## [1] 6.004662
```

```
#The next 10 are the RSME of each of the fitted predictions. One can see that the lowest one is the RSM
sqrt(sum((ozone$temperature-predict(treebag$fits[[1]]$fit,predictors))^2)/111)
```

```
## [1] 6.213156
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[2]]$fit,predictors))^2)/111)
```

```
## [1] 6.380369
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[3]]$fit,predictors))^2)/111)
```

```
## [1] 5.882444
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[4]]$fit,predictors))^2)/111)
```

```
## [1] 6.283436
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[5]]$fit,predictors))^2)/111)
```

```
## [1] 6.456446
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[6]]$fit,predictors))^2)/111)
```

```
## [1] 6.219213
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[7]]$fit,predictors))^2)/111)
```

```
## [1] 6.395468
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[8]]$fit,predictors))^2)/111)
```

```
## [1] 6.43415
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[9]]$fit,predictors))^2)/111)
```

```
## [1] 6.351157
```

```
sqrt(sum((ozone$temperature-predict(treebag$fits[[10]]$fit,predictors))^2)/111)
```

```
## [1] 6.325346
```

Random Forrest

The **Random Forrest** algorithm is similar to bagging. It bootstraps samples of the data and built a tree. At each split of the tree it bootstrap a set of the varibles (of the predictors). Then, the algorithm also creates many trees following the same logic.

For the prediction, each set of predictors is passed through each tree and then the final answer is the average of all the answers from each tree.

RF is a hightly accurate algorithm, however it can lead to overfitting. Therefore, cross-validation must be used in order to detect the overfitting.

The following is an example of RF algorithm in R:

```
data(iris); library(ggplot2)
library(caret)
inTrain <- createDataPartition(y=iris$Species,p=0.7, list=FALSE)
training <- iris[inTrain,]
testing <- iris[-inTrain,]
modFit_RF <- train(Species ~ .,data=training,method="rf",prox=TRUE)
modFit_RF
```

```
## Random Forest
##
## 105 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##   2     0.9490312  0.9221163  0.03738606   0.05756782
##   3     0.9521264  0.9269105  0.03618188   0.05549883
##   4     0.9522851  0.9271339  0.03604642   0.05530885
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 4.
```

```
#Comparing the RF model with the Prediction Tree Model
modFit_PredTree
```

```
## CART
##
## 105 samples
##   4 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...
## Resampling results across tuning parameters:
##
##   cp         Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.0000000  0.9370643  0.9043921  0.02805579   0.04254972
##   0.4428571  0.7500694  0.6348239  0.17135543   0.24405380
##   0.5000000  0.5339367  0.3305523  0.14747235   0.19752127
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.
```

```
#I can check one of the specific trees:
getTree(modFit_RF$finalModel,k=2)
```

```
##   left daughter right daughter split var split point status prediction
## 1             2              3         3        2.45      1          0
## 2             0              0         0        0.00     -1          1
## 3             4              5         4        1.65      1          0
## 4             0              0         0        0.00     -1          2
## 5             6              7         3        5.05      1          0
## 6             8              9         2        2.90      1          0
## 7             0              0         0        0.00     -1          3
## 8             0              0         0        0.00     -1          3
## 9             0              0         0        0.00     -1          2
```

```r
#Predicting over the testing set:
pred <- predict(modFit_RF,testing)
confusionMatrix(testing$Species,pred)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         14         1
##   virginica       0          0        15
##
## Overall Statistics
##
##                Accuracy : 0.9778
##                  95% CI : (0.8823, 0.9994)
##     No Information Rate : 0.3556
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9667
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9375
## Specificity                 1.0000            0.9677           1.0000
## Pos Pred Value              1.0000            0.9333           1.0000
## Neg Pred Value              1.0000            1.0000           0.9667
## Prevalence                  0.3333            0.3111           0.3556
## Detection Rate              0.3333            0.3111           0.3333
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9839           0.9688
```

```r
#Comparing the confusion Matrices of RF model and the pred tree model:

CM_PredTree
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         2
##   virginica       0          1        14
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.8173, 0.986)
##     No Information Rate : 0.3556
##     P-Value [Acc > NIR] : 5.426e-16
##
```

```
##                  Kappa : 0.9
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9286           0.8750
## Specificity                 1.0000            0.9355           0.9655
## Pos Pred Value              1.0000            0.8667           0.9333
## Neg Pred Value              1.0000            0.9667           0.9333
## Prevalence                  0.3333            0.3111           0.3556
## Detection Rate              0.3333            0.2889           0.3111
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9320           0.9203
```

Comparing the confusion matrix of the random forest algorithm over the testing set with that of the prediction tree, we can see that the RF is more accurate.

Boosting

Boosting algorithm takes several classifiers weigths and averages them in order to obtain a better one. It can use trees, glms, RF trees, etc. In R, in the **caret** packages, there are several options for boosting:

- **gbm** - boosting with trees
- **mboost** - model based boosting
- **ada** - statistical boosting based on additive logistic regression
- **gamBoost** - for boosting generalized additive models

In the next example we will use boosting prediction in the same problem of predicting the flower Species:

```r
data(iris); library(ggplot2)
library(caret)
inTrain <- createDataPartition(y=iris$Species,p=0.7, list=FALSE)
training <- iris[inTrain,]
testing <- iris[-inTrain,]
MF_Boosting <- train(Species ~ .,method="gbm",data=training,verbose=FALSE)
pred <- predict(MF_Boosting,testing)
confusionMatrix(testing$Species,pred)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         14         1
##   virginica       0          1        14
##
## Overall Statistics
##
##                Accuracy : 0.9556
##                  95% CI : (0.8485, 0.9946)
```

```
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9333
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9333           0.9333
## Specificity                 1.0000            0.9667           0.9667
## Pos Pred Value              1.0000            0.9333           0.9333
## Neg Pred Value              1.0000            0.9667           0.9667
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3111           0.3111
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9500           0.9500
```

Model Based Prediction

The basic ideas of model based prediction are:

- Assume that data follow a probabilistic model
- Use Bayes's theorem to identify optimal classifiers

There are several algorithms for Model Based Prediction:

- **Linear Discriminant Analysis - lda:** the discrimination function is a multivariate Gaussian with the same covariances
- **Quadratic Discriminant Analysis - qda:** the discrimination function is multivariate Gaussian with different covariances
- **Model Based Prediction**: assumes more complicated versions of the covariance matrix
- **Naive Bayes - nb:** assumes independence between features model building

In the following example we compare the Linear Discriminant Analysis algorithm with the Naive Bayes:

```
modlda <- train(Species ~ .,data=training,method="lda")
modnb <- train(Species ~ .,data=training,method="nb")
plda <- predict(modlda,testing)
pnb <- predict(modnb,testing)
confusionMatrix(testing$Species,plda)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         15         0
##   virginica       0          0        15
##
```

```
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9213, 1)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
## Neg Pred Value              1.0000            1.0000           1.0000
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.3333
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            1.0000           1.0000
```

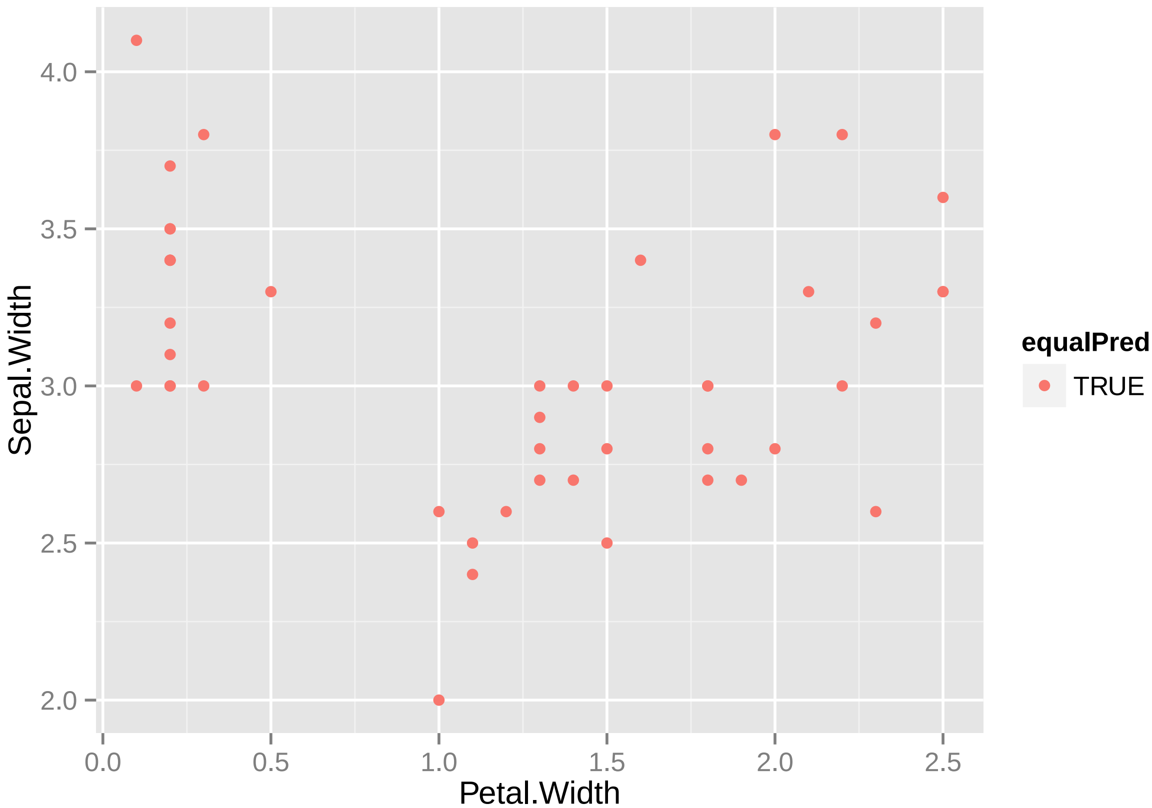
```
confusionMatrix(testing$Species,pnb)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         15         0
##   virginica       0          0        15
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9213, 1)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
## Neg Pred Value              1.0000            1.0000           1.0000
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.3333
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            1.0000           1.0000
```

```r
table(plda,pnb)
```

```
##             pnb
## plda         setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         15         0
##   virginica       0          0        15
```

```r
equalPred <- plda==pnb
qplot(Petal.Width,Sepal.Width,colour=equalPred,data=testing)
```



In the plot we can see that the values in which the predictions desagree are in the boundary between two classes.

Quiz-3

Question-1:

Load the cell segmentation data from the AppliedPredictiveModeling package using the commands:

library(AppliedPredictiveModeling) data(segmentationOriginal) library(caret) 1. Subset the data to a training set and testing set based on the Case variable in the data set.

2. Set the seed to 125 and fit a CART model with the rpart method using all predictor variables and default caret settings.

3. In the final model what would be the final model prediction for cases with the following variable values:

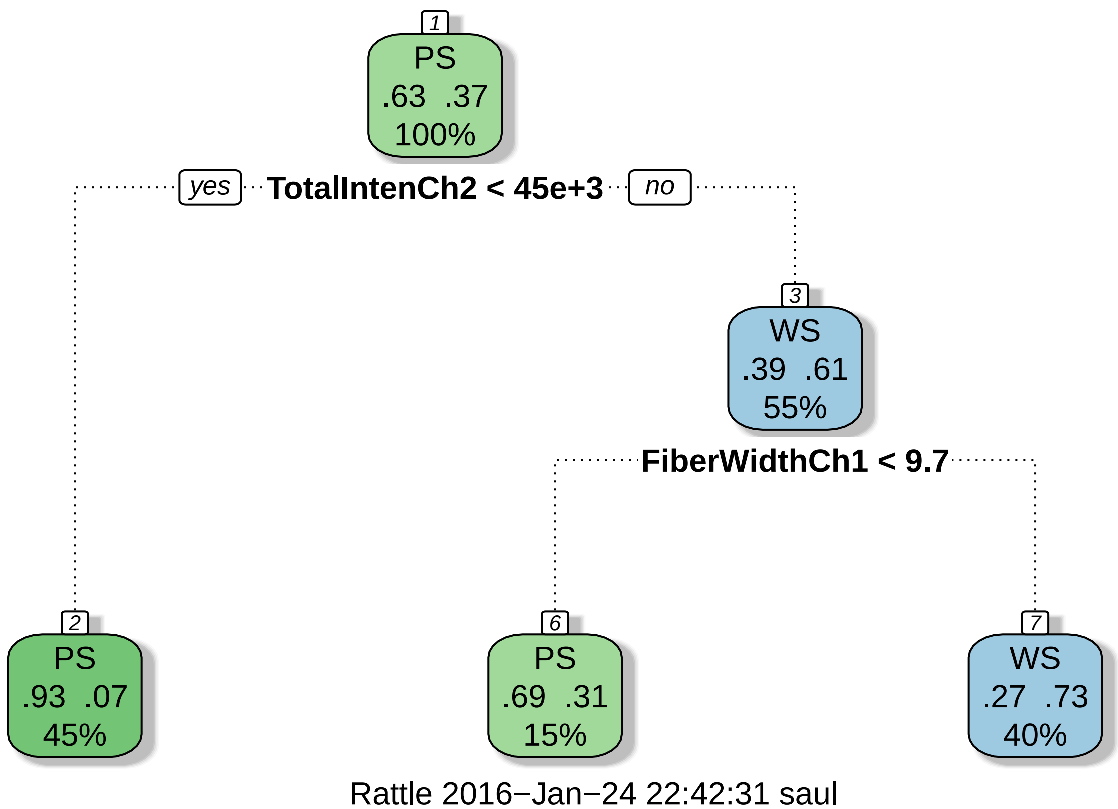
a. TotalIntench2 = 23,000; FiberWidthCh1 = 10; PerimStatusCh1=2

b. TotalIntench2 = 50,000; FiberWidthCh1 = 10;VarIntenCh4 = 100

c. TotalIntench2 = 57,000; FiberWidthCh1 = 8;VarIntenCh4 = 100

d. FiberWidthCh1 = 8;VarIntenCh4 = 100; PerimStatusCh1=2

```
library(AppliedPredictiveModeling)
data(segmentationOriginal)
library(caret)
training <- subset(segmentationOriginal,Case=="Train")
testing <- subset(segmentationOriginal,Case=="Test")
set.seed(125)
model_CART <- train(Class ~.,data=training,method="rpart")
library(rattle)
fancyRpartPlot(model_CART$finalModel)
```



Analyzing the Plot of the tree growth by the CART model we can see that the answer must be:

a.- PS b.- WS c.- PS d.- Not possible to predict

Question-2

If K is small in a K-fold cross validation is the bias in the estimate of out-of-sample (test set) accuracy smaller or bigger? If K is small is the variance in the estimate of out-of-sample (test set) accuracy smaller or bigger. Is K large or small in leave one out cross validation?

The bias is smaller and the variance is smaller. Under leave one out cross validation K is equal to one.

The bias is larger and the variance is smaller. Under leave one out cross validation K is equal to the sample size.

The bias is larger and the variance is smaller. Under leave one out cross validation K is equal to two.

The bias is smaller and the variance is bigger. Under leave one out cross validation K is equal to one.

As a reference for this question I read the following link [Cross-validation in Wikipedia] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

When k=n (the sample size) k-fold cross-validation is the same as leave-one-out cross-validation.

Also, the k is small you get more bias and less variance, so the answer is:

The bias is larger and the variance is smaller. Under leave one out cross validation K is equal to the sample size.

Question-3

Load the olive oil data using the commands:

library(pgmm) data(olive) olive = olive[,-1] (NOTE: If you have trouble installing the pgmm package, you can download the -code-olive-/code- dataset here: olive_data.zip. After unzipping the archive, you can load the file using the -code-load()-/code- function in R.)

These data contain information on 572 different Italian olive oils from multiple regions in Italy. Fit a classification tree where Area is the outcome variable. Then predict the value of area for the following data frame using the tree command with all defaults

newdata = as.data.frame(t(colMeans(olive))) What is the resulting prediction? Is the resulting prediction strange? Why or why not?

2.783. There is no reason why this result is strange.

4.59965. There is no reason why the result is strange.

0.005291005 0 0.994709 0 0 0 0 0 0. The result is strange because Area is a numeric variable and we should get the average within each leaf.

2.783. It is strange because Area should be a qualitative variable - but tree is reporting the average value of Area as a numeric variable in the leaf predicted for newdata

```
library(pgmm)
data(olive)
olive = olive[,-1]
head(olive)
```

```
##   Area Palmitic Palmitoleic Stearic Oleic Linoleic Linolenic Arachidic
## 1    1     1075          75     226  7823      672        36        60
## 2    1     1088          73     224  7709      781        31        61
## 3    1      911          54     246  8113      549        31        63
## 4    1      966          57     240  7952      619        50        78
## 5    1     1051          67     259  7771      672        50        80
## 6    1      911          49     268  7924      678        51        70
##   Eicosenoic
## 1         29
## 2         29
```

```
## 3          29
## 4          35
## 5          46
## 6          44
```

```r
system.time(modFit <- train(Area ~., data=olive, method="rpart"))
```

```
##    user  system elapsed
##   1.427   0.006   1.447
```

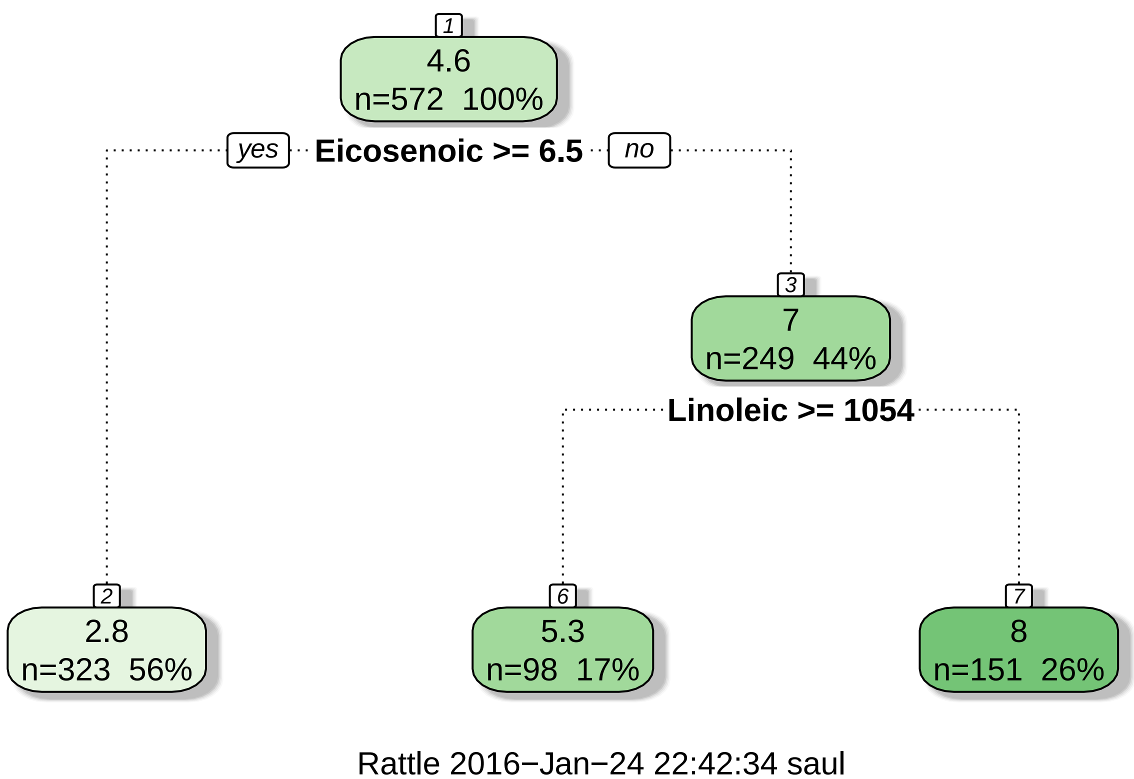
```r
newdata = as.data.frame(t(colMeans(olive)))
newdata
```

```
##      Area Palmitic Palmitoleic  Stearic    Oleic Linoleic Linolenic
## 1 4.59965 1231.741    126.0944 228.8654 7311.748  980.528  31.88811
##   Arachidic Eicosenoic
## 1   58.0979   16.28147
```

```r
predict(modFit,newdata)
```

```
##        1
## 2.783282
```

```r
fancyRpartPlot(modFit$finalModel)
```



```r
summary(olive$Area)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     3.0     3.0     4.6     7.0     9.0
```

Examining the tree model plot we can see that the prediction is correct (2.783). In my opinion the result is not extrange because in the dataset "Area" is not a qualitative variable but a quantitative one.

Question-4

Load the South Africa Heart Disease Data and create training and test sets with the following code:

library(ElemStatLearn) data(SAheart) set.seed(8484) train = sample(1:dim(SAheart)[1],size=dim(SAheart)[1]/2,replace=F) trainSA = SAheart[train,] testSA = SAheart[-train,] Then set the seed to 13234 and fit a logistic regression model (method="glm", be sure to specify family="binomial") with Coronary Heart Disease (chd) as the outcome and age at onset, current alcohol consumption, obesity levels, cumulative tabacco, type-A behavior, and low density lipoprotein cholesterol as predictors. Calculate the misclassification rate for your model using this function and a prediction on the "response" scale:

missClass = function(values,prediction){sum(((prediction > 0.5)*1) != values)/length(values)} What is the misclassification rate on the training set? What is the misclassification rate on the test set?

```r
library(ElemStatLearn)
data(SAheart)
set.seed(8484)
train = sample(1:dim(SAheart)[1],size=dim(SAheart)[1]/2,replace=F)
trainSA = SAheart[train,]
testSA = SAheart[-train,]
set.seed(13234)
head(trainSA)
```

```
##     sbp tobacco  ldl adiposity famhist typea obesity alcohol age chd
## 238 176    5.76 4.89     26.10 Present    46   27.30   19.44  57   0
## 114 174    0.00 8.46     35.10 Present    35   25.27    0.00  61   1
## 312 174    3.50 5.26     21.97 Present    36   22.04    8.33  59   1
## 301 166    4.10 4.00     34.30 Present    32   29.51    8.23  53   0
## 311 130    0.05 2.44     28.25 Present    67   30.86   40.32  34   0
## 179 128    0.04 8.22     28.17  Absent    65   26.24   11.73  24   0
```

```r
modFit <- train(chd ~ age + alcohol + obesity + tobacco + typea + ldl,data=trainSA,method="glm",family=

missClass = function(values,prediction){sum(((prediction > 0.5)*1) != values)/length(values)}

missClass(trainSA$chd,predict(modFit,trainSA))
```

```
## [1] 0.2727273
```

```r
missClass(testSA$chd,predict(modFit,testSA))
```

```
## [1] 0.3116883
```

Therefore, the misclassification error on the training set is 0.27 and on the test set is 0.31

Question-5

Load the vowel.train and vowel.test data sets:

library(ElemStatLearn) data(vowel.train) data(vowel.test) Set the variable y to be a factor variable in both the training and test set. Then set the seed to 33833. Fit a random forest predictor relating the factor variable y to the remaining variables. Read about variable importance in random forests here: http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr The caret package uses by default the Gini importance.

Calculate the variable importance using the varImp function in the caret package. What is the order of variable importance?

The order of the variables is:

x.10, x.7, x.9, x.5, x.8, x.4, x.6, x.3, x.1,x.2

The order of the variables is:

x.1, x.2, x.3, x.8, x.6, x.4, x.5, x.9, x.7,x.10

The order of the variables is:

x.10, x.7, x.5, x.6, x.8, x.4, x.9, x.3, x.1,x.2

The order of the variables is:

x.2, x.1, x.5, x.6, x.8, x.4, x.9, x.3, x.7,x.10

```r
library(ElemStatLearn)
data(vowel.train)
data(vowel.test)
summary(vowel.train)
```

```
##        y           x.1              x.2               x.3
##  Min.   : 1   Min.   :-5.211   Min.   :-1.2740   Min.   :-2.48700
##  1st Qu.: 3   1st Qu.:-3.923   1st Qu.: 0.9167   1st Qu.:-0.94550
##  Median : 6   Median :-3.097   Median : 1.7330   Median :-0.50250
##  Mean   : 6   Mean   :-3.167   Mean   : 1.7353   Mean   :-0.44800
##  3rd Qu.: 9   3rd Qu.:-2.512   3rd Qu.: 2.4038   3rd Qu.: 0.04925
##  Max.   :11   Max.   :-0.941   Max.   : 5.0740   Max.   : 1.41300
##       x.4               x.5               x.6               x.7
##  Min.   :-1.4090   Min.   :-2.1270   Min.   :-0.8360   Min.   :-1.53700
##  1st Qu.:-0.0835   1st Qu.:-0.9307   1st Qu.: 0.1085   1st Qu.:-0.29700
##  Median : 0.4565   Median :-0.4170   Median : 0.5275   Median : 0.04000
##  Mean   : 0.5250   Mean   :-0.3893   Mean   : 0.5850   Mean   : 0.01748
##  3rd Qu.: 1.1640   3rd Qu.: 0.1155   3rd Qu.: 1.0097   3rd Qu.: 0.34800
##  Max.   : 2.1910   Max.   : 1.8310   Max.   : 2.3270   Max.   : 1.40300
##       x.8                x.9               x.10
##  Min.   :-1.29300   Min.   :-1.6130   Min.   :-1.68000
##  1st Qu.:-0.01825   1st Qu.:-0.6737   1st Qu.:-0.50700
##  Median : 0.47700   Median :-0.2550   Median :-0.08250
##  Mean   : 0.41739   Mean   :-0.2681   Mean   :-0.08457
##  3rd Qu.: 0.86125   3rd Qu.: 0.1375   3rd Qu.: 0.30100
##  Max.   : 1.67300   Max.   : 1.3090   Max.   : 1.39600
```

```r
vowel.train$y <- as.factor(vowel.train$y)
vowel.test$y <- as.factor(vowel.test$y)
set.seed(33833)
library(caret)
modFit <- train(y ~., data=vowel.train,method="rf")
modFit
```

```
## Random Forest
##
## 528 samples
##  10 predictor
##  11 classes: '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 528, 528, 528, 528, 528, 528, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9348086  0.9281044  0.01989488   0.02191478
##    6    0.9084462  0.8990394  0.02196780   0.02423451
##   10    0.8769423  0.8643308  0.02686962   0.02962649
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```r
varImp(modFit)
```

```
## rf variable importance
##
##      Overall
## x.1  100.000
## x.2   93.548
## x.5   41.699
## x.6   27.438
## x.8   18.965
## x.4    9.033
## x.3    6.735
## x.9    4.706
## x.7    1.903
## x.10   0.000
```