MLPJ

R Markdown

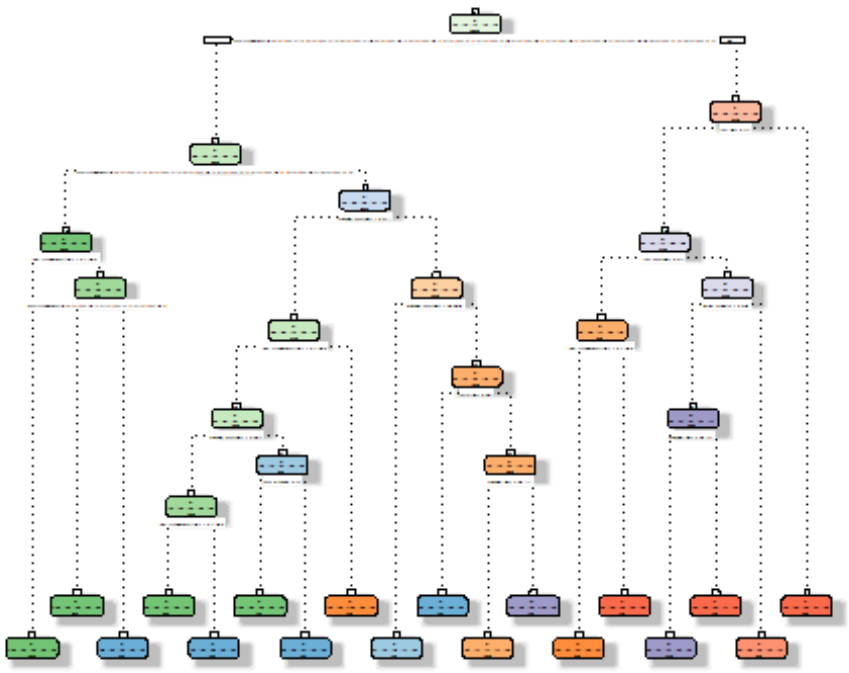
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

library(rpart)

## Warning: package 'rpart' was built under R version 3.3.2

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.3.2

library(RColorBrewer)
library(rattle)

## Warning: package 'rattle' was built under R version 3.3.2

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.3.2

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

library(knitr)

set.seed(12345)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"

training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))

##Partioning the training set into two

inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)

## [1] 11776   160

## [1] 7846  160

nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]

nzv<- nearZeroVar(myTesting,saveMetrics=TRUE)
myTesting <- myTesting[,nzv$nzv==FALSE]

##Remove the first column of the myTraining data set

myTraining <- myTraining[c(-1)]

trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
  if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
    for(j in 1:length(trainingV3)) {
      if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) == 1)  {
        trainingV3 <- trainingV3[ , -j]
      }
    }
  }
}

# Set back to the original variable name
myTraining <- trainingV3
```

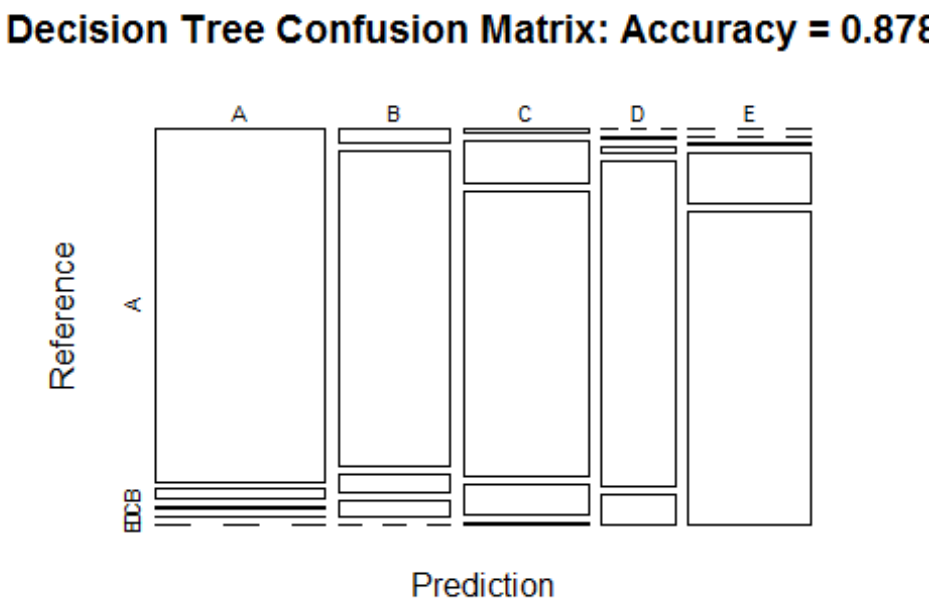
```r
rm(trainingV3)

clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])  # remove the classe column
myTesting <- myTesting[clean1]         # allow only variables in myTesting
that are also in myTraining
testing <- testing[clean2]             # allow only variables in testing that
are also in myTraining

for (i in 1:length(testing) ) {
  for(j in 1:length(myTraining)) {
    if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
      class(testing[j]) <- class(myTraining[i])
    }
  }
}

# To get the same class between testing and myTraining
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]

set.seed(12345)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2150   60    7    1    0
##          B   61 1260   69   64    0
##          C   21  188 1269  143    4
##          D    0   10   14  857   78
##          E    0    0    9  221 1360
##
## Overall Statistics
##
##                Accuracy : 0.8789
##                  95% CI : (0.8715, 0.8861)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8468
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9633   0.8300   0.9276   0.6664   0.9431
## Specificity            0.9879   0.9693   0.9450   0.9845   0.9641
## Pos Pred Value         0.9693   0.8666   0.7809   0.8936   0.8553
## Neg Pred Value         0.9854   0.9596   0.9841   0.9377   0.9869
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2740   0.1606   0.1617   0.1092   0.1733
## Detection Prevalence   0.2827   0.1853   0.2071   0.1222   0.2027
## Balanced Accuracy      0.9756   0.8997   0.9363   0.8254   0.9536
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree
Confusion Matrix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```

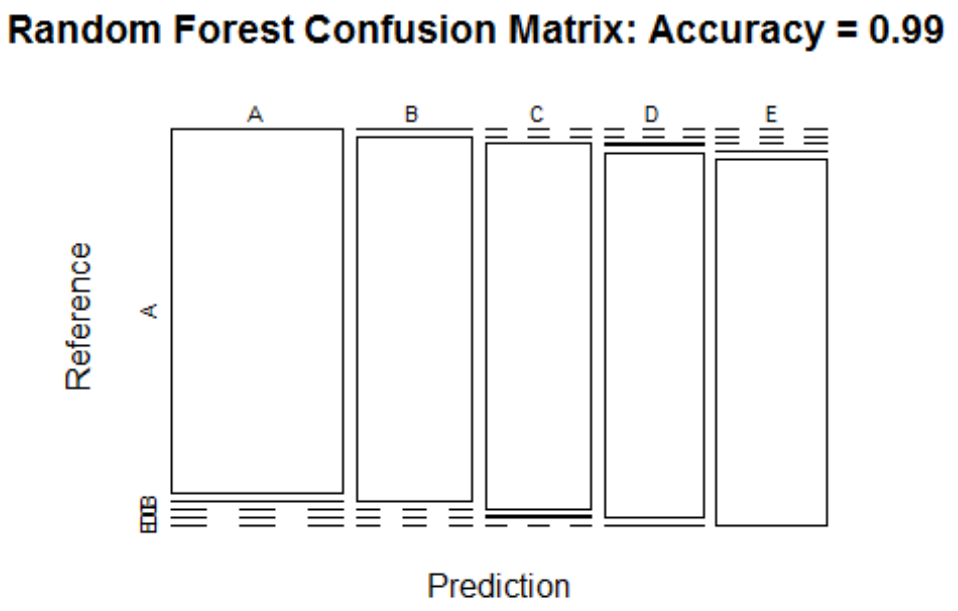



```
set.seed(12345)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    2    0    0    0
##          B    1 1516    0    0    0
##          C    0    0 1367    3    0
##          D    0    0    1 1282    1
##          E    0    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9993   0.9969   0.9993
## Specificity            0.9996   0.9998   0.9995   0.9997   0.9998
## Pos Pred Value         0.9991   0.9993   0.9978   0.9984   0.9993
## Neg Pred Value         0.9998   0.9997   0.9998   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1932   0.1742   0.1634   0.1837
## Detection Prevalence   0.2846   0.1933   0.1746   0.1637   0.1838
## Balanced Accuracy      0.9996   0.9993   0.9994   0.9983   0.9996
```

```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion
Matrix: Accuracy =", round(cmrf$overall['Accuracy'], 4)))
```

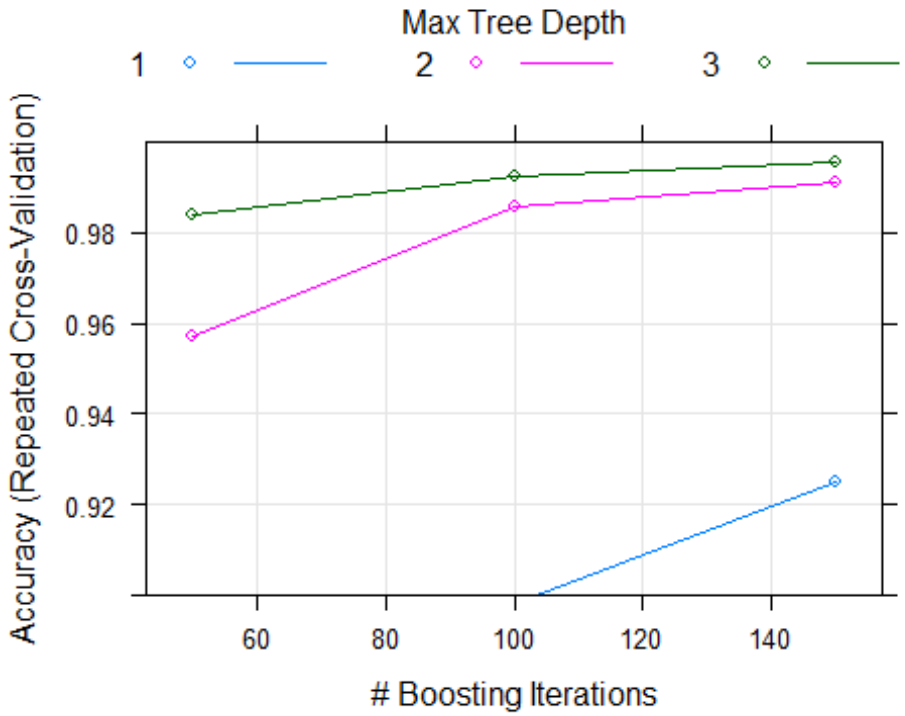



```
set.seed(12345)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

gbmFit1 <- train(classe ~ ., data=myTraining, method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)

## Loading required package: gbm

## Warning: package 'gbm' was built under R version 3.3.2

## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 3.3.2

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.1

## Loading required package: plyr

## Warning: package 'plyr' was built under R version 3.3.2

gbmFinMod1 <- gbmFit1$finalModel

gbmPredTest <- predict(gbmFit1, newdata=myTesting)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, myTesting$classe)
gbmAccuracyTest

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    4    0    0    0
##          B    1 1512    1    0    0
##          C    0    2 1361    3    0
##          D    0    0    6 1274    1
##          E    0    0    0    9 1441
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.995, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9960   0.9949   0.9907   0.9993
## Specificity            0.9993   0.9997   0.9992   0.9989   0.9986
## Pos Pred Value         0.9982   0.9987   0.9963   0.9945   0.9938
## Neg Pred Value         0.9998   0.9991   0.9989   0.9982   0.9998
```

```
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1927   0.1735   0.1624   0.1837
## Detection Prevalence   0.2849   0.1930   0.1741   0.1633   0.1848
## Balanced Accuracy      0.9994   0.9979   0.9971   0.9948   0.9990
```

```
plot(gbmFit1, ylim=c(0.9, 1))
```




```
predictionB2 <- predict(modFitB1, testing, type = "class")
predictionB2
```

```
##  1  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.