Practical_ML_week4

Alankrit Gupta

February 12, 2017

AIM

The aim is to predict whether the manner in which they did the exercise was correct or incorrect. This is the "classe" variable in the training set. You may use any of the other variables to predict with. Create a report describing how the modelwas built, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Executive summary

This report will describe how the data captured are used to identify the parameters involved in predicting the movement involved based on the above classification, and then to predict the movement for 20 test cases.

The training data were divided into two groups, a training data and a validation data (to be used to validate the data), to derived the prediction model by using the training data, to validate the model where an expected out-of-sample error rate of less than 0.5%, or 99.5% accuracy, would be acceptable before it is used to perform the prediction on the 20 test cases - that must have 100% accuracy (to obtain 20 points awarded).

The modle trained using decision tree was able to achieve almost 87% accuracy on the validation set whereas the training model developed using Random Forest was able to achieve over 99.89% accuracy, and was able to predict the 20 test cases with 100% accuracy.

Laoding Libraries

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(knitr)
library(randomForest)

## 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
```

```r
library(readr)
library(rattle)
```

```
## 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.
```

Laoding data

```r
setwd("C:/Users/ag14721/Downloads/Self Study/prac_ML")
training <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
testing <- read.csv("pml-testing.csv")
```

Creating test-train split

```r
set.seed(12345)
split <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- (training[split, ])
myTesting <- (training[-split, ])
```

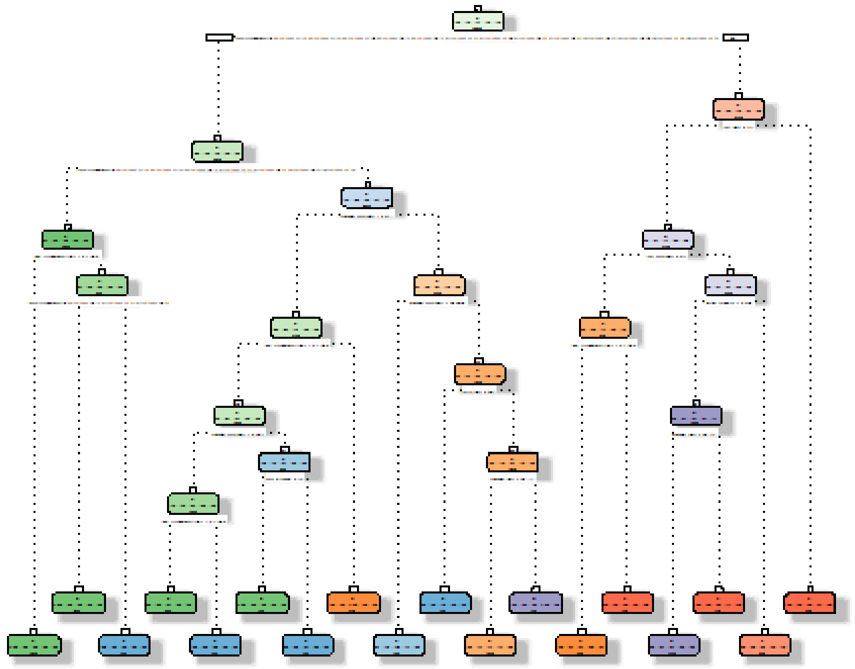
Making some changes in the datasets

Few variables are not very relevant to the model because they do not have contributing insights

```r
#removing variables nearly zero variance



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

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



#removing variables that have more than a 70% of NA's.
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
rm(trainingV3)

#Doing the same cleaning to myTesting and testing data sets

clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -59])
myTesting <- myTesting[clean1]
testing <- testing[clean2]
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, -59] , testing)
testing <- testing[-1,]

## removing user ID varaibles so that they do not interfere with the model
myTraining$X<- NULL
myTesting$X<-NULL

testing$X<-NULL
```

Modeling using decision trees

```
set.seed(1000)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```


Rattle 2017-Feb-14 12:10:05 ag14721

```
# prediction
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
```

Modeling using Random forest

```
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
```

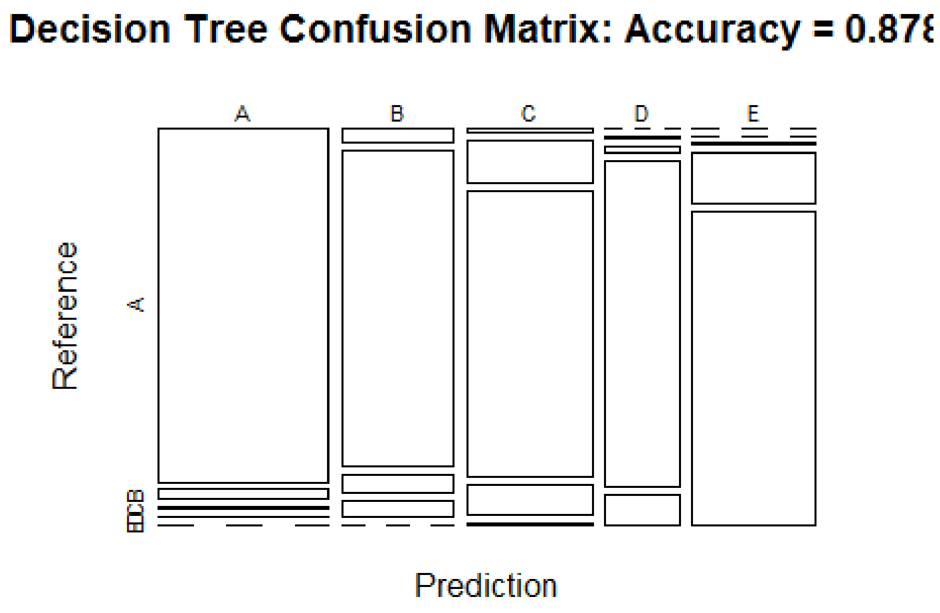
Prediction on test files

```
predictionsB2 <- predict(modFitB1, testing, type = "class")

# Write the results to a text file for submission
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(predictionsB2)
```

The accuracy of random forest is better than the decision tree model, and therefore i'll be using thsi for predictions on the test dataset.

Including Plots

Decision tree plot

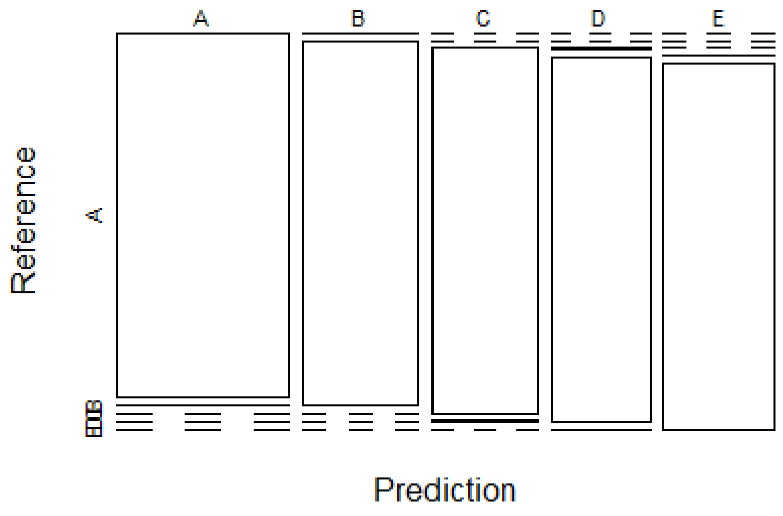


### Random

Forest plot

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

Random Forest Confusion Matrix: Accuracy = 0.99


A
B
C
D
E
Reference
A
B
C
D
E
Prediction