Practical Machine Learning Project

DMK

7/9/2020

This exercise uses data made available from [Groupware@LES](mailto:Groupware@LES) (<http://groupware.les.inf.puc-rio.br/har>). This exercise asks that the “how well” rather than the “which” or “how much”, an activity is performed by the wearer of measurement device. Using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants where they were asked to perform barbell lifts correctly and incorrectly in 5 different ways, the model will attempt to predict the manner in which the participant performed the exercise. Data were collected from <http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har>.

Source Paper citation:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: <http://groupware.les.inf.puc-rio.br/har#ixzz6RjNQ1Hsj>

library(caret); library(dplyr); library(rattle); library(randomForest)

## Loading required package: lattice

## Loading required package: ggplot2

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':  
##   
## importance

## The following object is masked from 'package:dplyr':  
##   
## combine

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

# Import data sets  
TrainSet <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))  
TestSet <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))

Partition data into “Training” and “Test” subsets using 80% / 20% rule

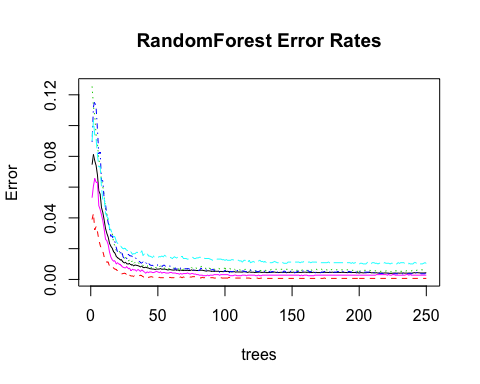
Use <- createDataPartition(TrainSet$classe, p = .8, list = FALSE)  
UseTrain <- TrainSet[Use, ]  
UseTest <- TrainSet[-Use, ]  
  
# The next step cleanses data to remove columns that are mostly NA values  
TrainUse\_Use <- UseTrain[,c(8:11, 37:49, 60:68, 84:86, 102, 113:124, 140, 151:160)]

Developing the model

set.seed(1234)  
forest <- randomForest(classe ~., data = TrainUse\_Use, ntree = 250, importance = TRUE)  
forest

##   
## Call:  
## randomForest(formula = classe ~ ., data = TrainUse\_Use, ntree = 250, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 250  
## No. of variables tried at each split: 7  
##   
## OOB estimate of error rate: 0.43%  
## Confusion matrix:  
## A B C D E class.error  
## A 4461 3 0 0 0 0.000672043  
## B 13 3019 6 0 0 0.006254115  
## C 0 8 2727 3 0 0.004017531  
## D 0 0 25 2546 2 0.010493587  
## E 0 0 1 7 2878 0.002772003

plot(forest, main = "RandomForest Error Rates")



# Check to see which are most influencial variables  
  
Influencers <- importance(forest)  
Sorted <- as.data.frame(Influencers)  
Gini <- Sorted[order(-Sorted$MeanDecreaseGini),]  
Accy <- Sorted[order(-Sorted$MeanDecreaseAccuracy),]  
Gini[1:10,]

## A B C D E  
## roll\_belt 27.18714 34.19279 30.84162 32.84198 28.94027  
## yaw\_belt 34.87575 31.62182 30.30523 32.40665 24.33327  
## pitch\_forearm 25.55180 25.59500 29.64816 25.72528 22.92090  
## magnet\_dumbbell\_z 33.60443 30.46979 34.13395 27.77888 27.38973  
## pitch\_belt 19.42700 33.70051 25.51709 23.19490 21.05078  
## magnet\_dumbbell\_y 24.08777 25.68977 33.35920 24.44546 21.64993  
## roll\_forearm 21.46159 19.81915 24.97609 17.81889 19.17610  
## magnet\_dumbbell\_x 17.69260 18.97622 20.50253 19.37178 16.29543  
## accel\_dumbbell\_y 18.45418 20.59992 23.26267 20.39579 20.44798  
## roll\_dumbbell 17.08171 19.61875 20.05042 18.89985 18.69797  
## MeanDecreaseAccuracy MeanDecreaseGini  
## roll\_belt 37.47919 988.7864  
## yaw\_belt 45.53688 702.0780  
## pitch\_forearm 31.90402 649.5957  
## magnet\_dumbbell\_z 38.54984 611.4066  
## pitch\_belt 30.73966 550.9180  
## magnet\_dumbbell\_y 29.83012 518.0337  
## roll\_forearm 22.53848 506.3469  
## magnet\_dumbbell\_x 20.58528 396.0522  
## accel\_dumbbell\_y 25.53489 338.1597  
## roll\_dumbbell 21.24291 337.2925

Accy[1:10,]

## A B C D E  
## yaw\_belt 34.87575 31.62182 30.30523 32.40665 24.33327  
## magnet\_dumbbell\_z 33.60443 30.46979 34.13395 27.77888 27.38973  
## roll\_belt 27.18714 34.19279 30.84162 32.84198 28.94027  
## pitch\_forearm 25.55180 25.59500 29.64816 25.72528 22.92090  
## pitch\_belt 19.42700 33.70051 25.51709 23.19490 21.05078  
## magnet\_dumbbell\_y 24.08777 25.68977 33.35920 24.44546 21.64993  
## gyros\_arm\_y 14.94091 19.89207 17.32532 18.69564 15.45260  
## roll\_arm 16.13957 22.15490 22.14432 21.16644 17.04248  
## gyros\_dumbbell\_x 14.21842 19.16199 16.29872 13.81586 14.69348  
## accel\_dumbbell\_y 18.45418 20.59992 23.26267 20.39579 20.44798  
## MeanDecreaseAccuracy MeanDecreaseGini  
## yaw\_belt 45.53688 702.07796  
## magnet\_dumbbell\_z 38.54984 611.40655  
## roll\_belt 37.47919 988.78638  
## pitch\_forearm 31.90402 649.59566  
## pitch\_belt 30.73966 550.91801  
## magnet\_dumbbell\_y 29.83012 518.03369  
## gyros\_arm\_y 27.23873 111.34342  
## roll\_arm 27.04291 267.46004  
## gyros\_dumbbell\_x 26.13494 98.22861  
## accel\_dumbbell\_y 25.53489 338.15967

Applying model to predeictions

Predict = predict(forest, newdata = UseTrain)  
confusionMatrix(Predict, UseTrain$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 4464 0 0 0 0  
## B 0 3038 0 0 0  
## C 0 0 2738 0 0  
## D 0 0 0 2573 0  
## E 0 0 0 0 2886  
##   
## Overall Statistics  
##   
## Accuracy : 1   
## 95% CI : (0.9998, 1)  
## No Information Rate : 0.2843   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 1.0000 1.0000 1.0000 1.0000  
## Specificity 1.0000 1.0000 1.0000 1.0000 1.0000  
## Pos Pred Value 1.0000 1.0000 1.0000 1.0000 1.0000  
## Neg Pred Value 1.0000 1.0000 1.0000 1.0000 1.0000  
## Prevalence 0.2843 0.1935 0.1744 0.1639 0.1838  
## Detection Rate 0.2843 0.1935 0.1744 0.1639 0.1838  
## Detection Prevalence 0.2843 0.1935 0.1744 0.1639 0.1838  
## Balanced Accuracy 1.0000 1.0000 1.0000 1.0000 1.0000

# As this model generates an accuracy of 1, all appropriate influencers must have been selected in developing the model. Applying this model we get following results.  
  
FinalTest <- predict(forest, newdata = TestSet)  
FinalTest

## 1 2 3 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