practical machines

Binayak

17/02/2021

Background Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement ??? a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset)

###Data The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Library Loading

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## 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.
```

```
library(randomForest)
```

```
## 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:ggplot2':
##
##     margin
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

Data Loading

```
train1<-read.csv("C:/Users/binayak mishra/Downloads/pml-training.csv",header = TRUE)
test1<-read.csv("C:/Users/binayak mishra/Downloads/pml-testing.csv",header = TRUE)
```

Data Cleaning

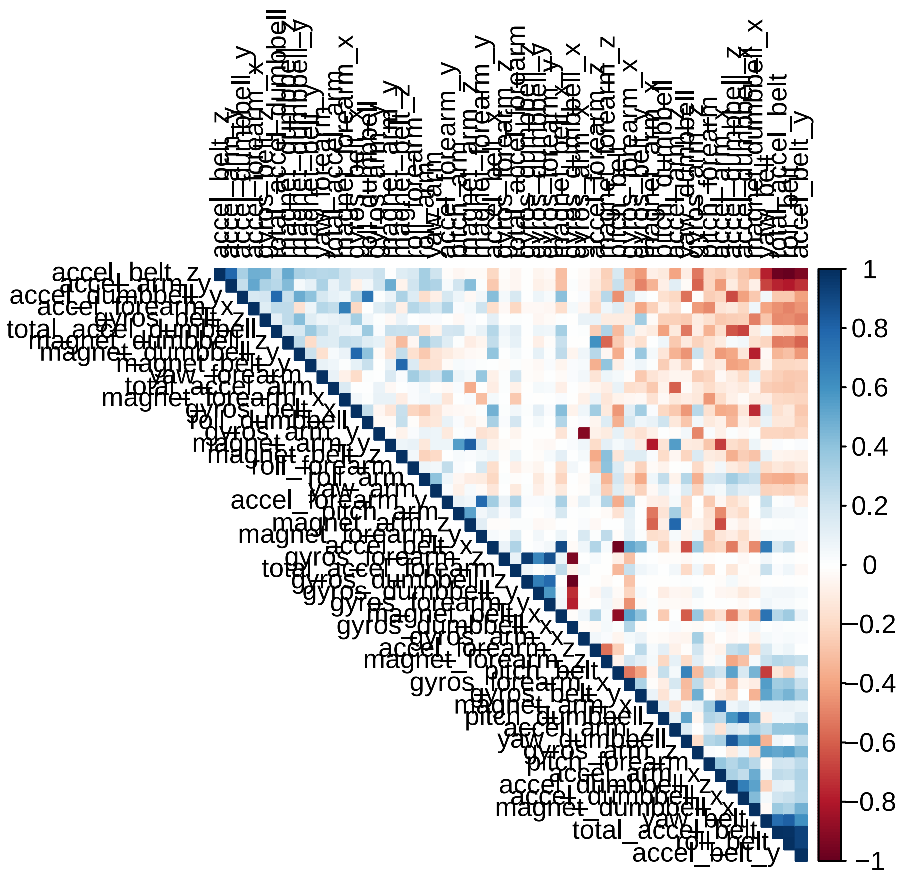
```
training<-train1[,colSums(is.na(train1))==0]
testing<-test1[,colSums(is.na(test1))==0]
training<-training[,-c(1:7)]
testing<-testing[,-c(1:7)]
```

Dataset for prediction

```
set.seed(12345)
intrain<-createDataPartition(training$classe,p =0.7,list = FALSE)
training<-training[intrain,]
testing<-training[-intrain,]
### Cleaning by removing nearly zero variance variable
Nz<-nearZeroVar(training)
training<-training[,-Nz]
testing<-testing[,-Nz]
```

Plotting the correlation Matrix

```r
cormat<-cor(training[,-53])
corrplot(cormat,order = "FPC",method = "color",type = "upper",tl.cex = 0.8,tl.col = rgb(0,0,0))
```

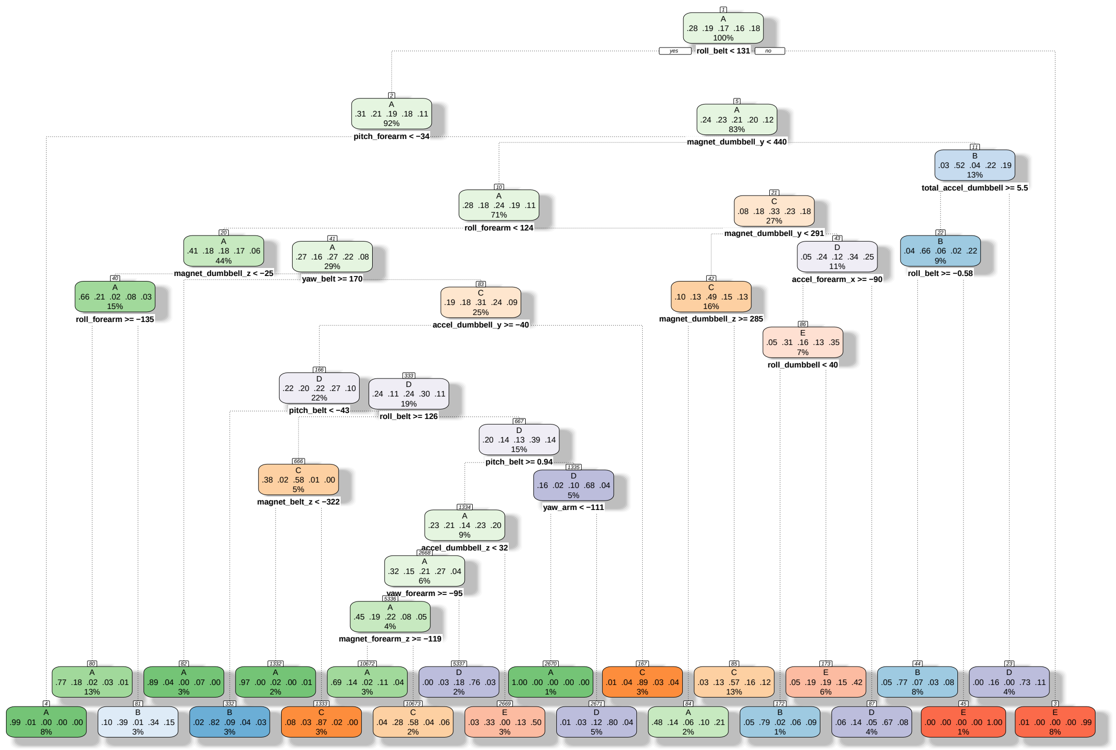


```r
highcorr<-findCorrelation(cormat,cutoff = 0.75)
names(training)[highcorr]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "accel_arm_y"       "total_accel_belt"  "accel_dumbbell_z"
##  [7] "accel_belt_x"      "pitch_belt"        "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y" "accel_dumbbell_x"
## [13] "accel_arm_x"       "accel_arm_z"       "magnet_arm_y"
## [16] "magnet_belt_z"     "accel_forearm_y"   "gyros_forearm_y"
## [19] "gyros_dumbbell_x"  "gyros_dumbbell_z"  "gyros_arm_x"
```

MODEL BUILDING USING CLASSIFICATION TREE

```r
Mod1Tr<-rpart(classe ~ .,data = training, method = "class")
fancyRpartPlot(Mod1Tr)
```



Rattle 2021−Feb−17 16:51:27 binayak mishra

```
predmod1Tr<-predict(Mod1Tr,testing,type = "class")
classes<- as.factor(testing$classe)
cmTr<-confusionMatrix(predmod1Tr,classes)
cmTr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1086  134   12   46   28
##          B   30  438   40   42   54
##          C   27  103  589  106   81
##          D   11   53   43  406   40
##          E   18   70   39   53  591
##
## Overall Statistics
##
##                Accuracy : 0.7512
##                  95% CI : (0.7377, 0.7643)
##     No Information Rate : 0.2831
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6841
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9266   0.5489   0.8147  0.62175   0.7443
## Specificity            0.9259   0.9503   0.9072  0.95784   0.9462
## Pos Pred Value         0.8315   0.7252   0.6501  0.73418   0.7665
## Neg Pred Value         0.9697   0.8982   0.9586  0.93114   0.9397
## Prevalence             0.2831   0.1928   0.1746  0.15773   0.1918
## Detection Rate         0.2623   0.1058   0.1423  0.09807   0.1428
## Detection Prevalence   0.3155   0.1459   0.2188  0.13357   0.1862
## Balanced Accuracy      0.9262   0.7496   0.8609  0.78979   0.8453
```

MODEL BUILDING USING RANDOM FOREST

```
contRf<-trainControl(method ="cv",number=5)
contRf
```

```
## $method
## [1] "cv"
##
## $number
## [1] 5
##
## $repeats
## [1] NA
##
## $search
## [1] "grid"
##
## $p
## [1] 0.75
##
## $initialWindow
## NULL
##
## $horizon
## [1] 1
##
## $fixedWindow
## [1] TRUE
##
## $skip
## [1] 0
##
## $verboseIter
## [1] FALSE
##
## $returnData
## [1] TRUE
##
## $returnResamp
## [1] "final"
```

```
##
## $savePredictions
## [1] FALSE
##
## $classProbs
## [1] FALSE
##
## $summaryFunction
## function (data, lev = NULL, model = NULL)
## {
##     if (is.character(data$obs))
##         data$obs <- factor(data$obs, levels = lev)
##     postResample(data[, "pred"], data[, "obs"])
## }
## <bytecode: 0x0000000012231be8>
## <environment: namespace:caret>
##
## $selectionFunction
## [1] "best"
##
## $preProcOptions
## $preProcOptions$thresh
## [1] 0.95
##
## $preProcOptions$ICAcomp
## [1] 3
##
## $preProcOptions$k
## [1] 5
##
## $preProcOptions$freqCut
## [1] 19
##
## $preProcOptions$uniqueCut
## [1] 10
##
## $preProcOptions$cutoff
## [1] 0.9
##
##
## $sampling
## NULL
##
## $index
## NULL
##
## $indexOut
## NULL
##
## $indexFinal
## NULL
##
## $timingSamps
## [1] 0
```

```
##
## $predictionBounds
## [1] FALSE FALSE
##
## $seeds
## [1] NA
##
## $adaptive
## $adaptive$min
## [1] 5
##
## $adaptive$alpha
## [1] 0.05
##
## $adaptive$method
## [1] "gls"
##
## $adaptive$complete
## [1] TRUE
##
##
## $trim
## [1] FALSE
##
## $allowParallel
## [1] TRUE
```

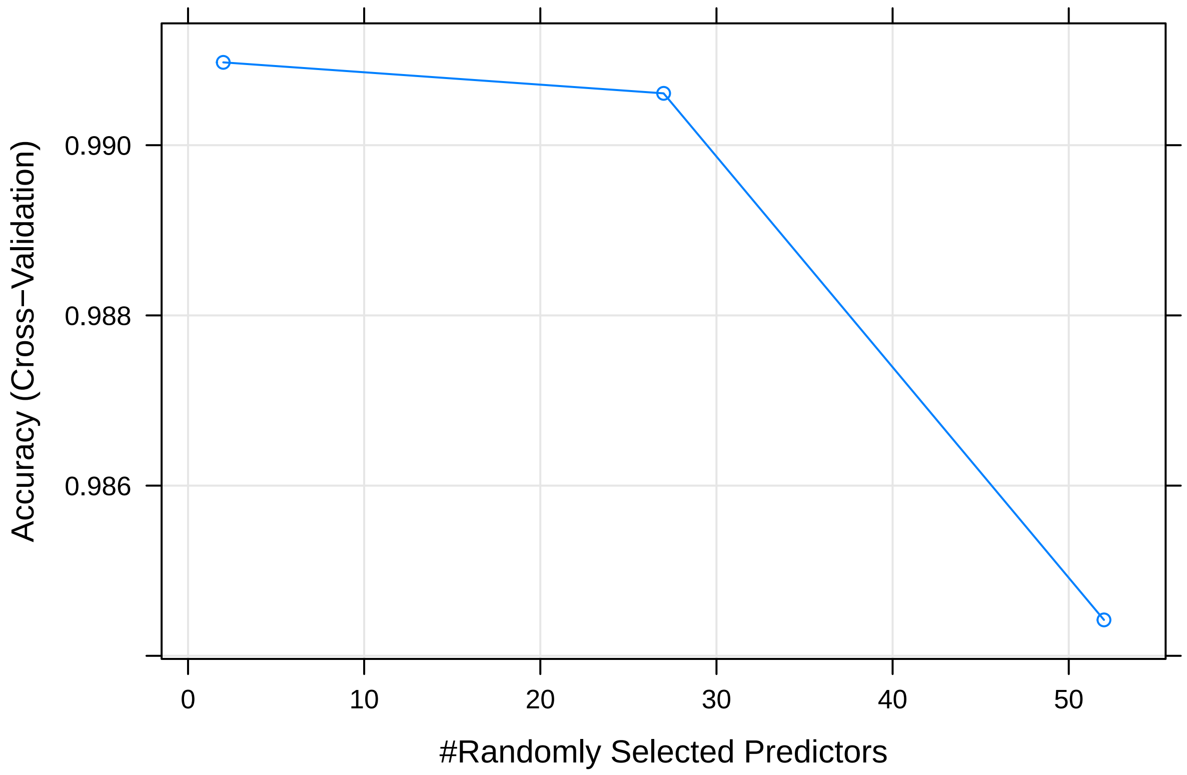
```
MODRF11<- train(classe~., data = training,method="rf",trControl=contRf,verbose= FALSE)
predictF1<-predict(MODRF11,newdata = testing)
```

```
classes<- as.factor(testing$classe)
cmrf1<-confusionMatrix(predictF1,classes)
cmrf1
```

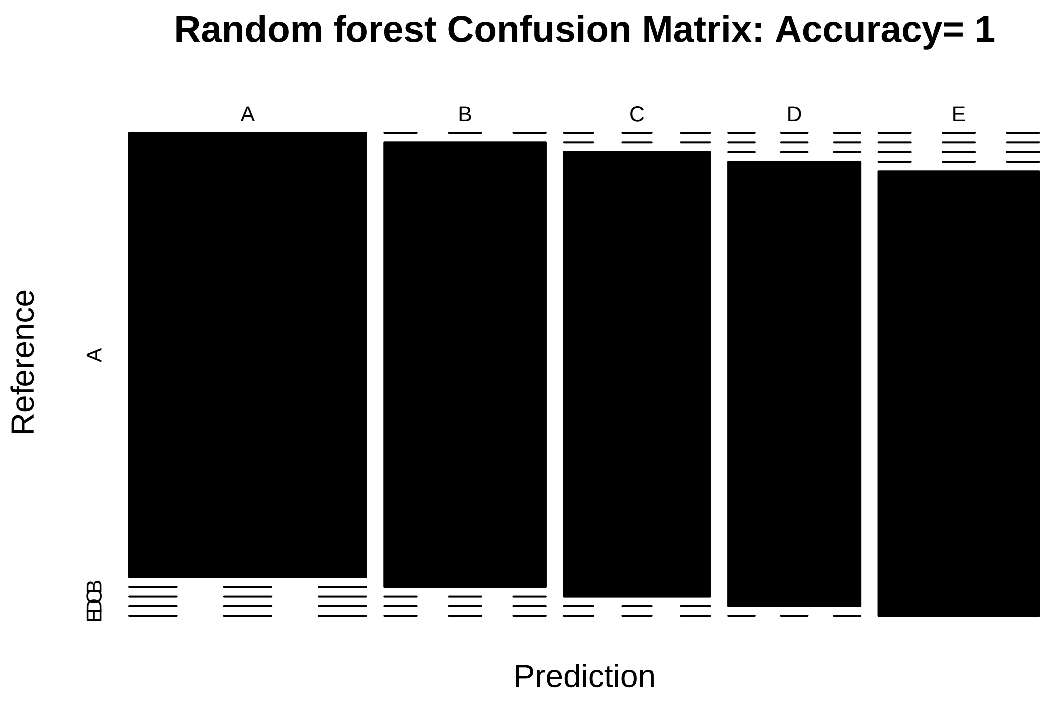
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1172    0    0    0    0
##          B    0  798    0    0    0
##          C    0    0  723    0    0
##          D    0    0    0  653    0
##          E    0    0    0    0  794
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2831
##     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             0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Rate         0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Prevalence   0.2831   0.1928   0.1746   0.1577   0.1918
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
### PLOTTING THE Random forest model
plot(MODRF11)
```



```
plot(cmrf1$table,col=cmrf1$byClass,main= paste("Random forest Confusion Matrix: Accuracy=",round(cmrf1$
```



### MODEL BUILDING USING GBM

```
set.seed(12345)
controlGBM<-trainControl(method = "repeatedcv",number = 5,repeats = 1)
modgbm<-train(classe ~.,data = training, method="gbm", trControl=controlGBM, verbose= FALSE)
modgbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 51 had non-zero influence.
```

```
print(modgbm)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10990, 10990, 10989, 10991, 10988
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7485616  0.6811069
##   1                  100      0.8226676  0.7755529
##   1                  150      0.8554984  0.8171937
##   2                   50      0.8564451  0.8181992
##   2                  100      0.9055825  0.8805462
##   2                  150      0.9299697  0.9114036
##   3                   50      0.8994676  0.8727568
##   3                  100      0.9407428  0.9250337
##   3                  150      0.9608349  0.9504599
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
predictGBM<-predict(modgbm, newdata = testing)
cmGBM<- confusionMatrix(predictGBM,classes)
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1164   15    0    1    1
##          B    8  771   20    1    6
##          C    0   11  699   24    3
##          D    0    0    4  625    6
##          E    0    1    0    2  778
##
## Overall Statistics
##
##                Accuracy : 0.9751
##                  95% CI : (0.9699, 0.9796)
##     No Information Rate : 0.2831
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9685
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9932   0.9662   0.9668   0.9571   0.9798
## Specificity            0.9943   0.9895   0.9889   0.9971   0.9991
## Pos Pred Value         0.9856   0.9566   0.9484   0.9843   0.9962
## Neg Pred Value         0.9973   0.9919   0.9929   0.9920   0.9952
## Prevalence             0.2831   0.1928   0.1746   0.1577   0.1918
## Detection Rate         0.2812   0.1862   0.1688   0.1510   0.1879
## Detection Prevalence   0.2853   0.1947   0.1780   0.1534   0.1886
## Balanced Accuracy      0.9937   0.9778   0.9778   0.9771   0.9895
```

Result

We use random forest algorithm to predict the test set

```
FinalResult<-predict(MODRF11,newdata = test1)
FinalResult
```

```
##  [1] 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
```

Conclusion

As can be seen that random forest gives us a prediction accuracy of 100% which is better than gbm(97.5% accuracy) and decision tree whose accuracy is (75%)