Weight Lifting Classification

M Hamersma

14 April 2020

Introduction

This is a HAR (Human Activity Recognition) model. The objective is to analyse the x, y and z direction movement data captured from wearable fitness devices while doing bicep curls with a 1.25kg dumbell, and classify new exercises as having been done correctly or not. Data and concept is from Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements.

Read more: http://groupware.les.inf.puc-rio.br/work.jsf?p1=10335#ixzz6JZBFyhzW

The model uses a machine learning algorithm, based on the captured data. The data will be used for training the model against a 70% subset, then validatinig it against the remaining 30% of data to estimate an Out of Sample error. Finally it is tested against a test data set of 20 exercises supplied by the John Hopkins Data Science Specialization course material.

Data and Environment

Prepare the environment

```
setwd("~/R/PML/Project")
library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

## Loading required package: ggplot2

library(readr)
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.6.3

## corrplot 0.84 loaded

library(rattle)

## Warning: package 'rattle' was built under R version 3.6.3

## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.6.3

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.6.3

## 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
```

Loading Data

The data was downloaded to two csv files:

```
training <- read.csv("pml-training1.csv", sep = ";", na.strings = c("NA",
"#DIV/0!", ""))
testing <- read.csv("pml-testing1.csv", sep = ";", na.strings = c("NA",
"#DIV/0!", ""))

dim(training)

## [1] 19622   160

dim(testing)

## [1]  20 160
```

Tidying up

The data contains many NA and near zero variables. Due to the nature of the Machine Learning algorithms, the amount of data, and the fact that the exercises has already been classified, there is no need to dig too deep - rather focus on detecting the variables that influence the classification.

The first five columns are just name identfiers and can be removed.

```
names(training[, 1:5])
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"

training <- training[, -(1:5)]
dim(training)

## [1] 19622   155
```

Remove variables from training with near zero variance, using the train set as reference

```
NearZV <- nearZeroVar(training)
training <- training[, -NearZV]
dim(training)

## [1] 19622   119
```

remove variables that are mostly NA

```
AllNA    <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, AllNA==FALSE]
dim(training)

## [1] 19622    54
```

Modelling

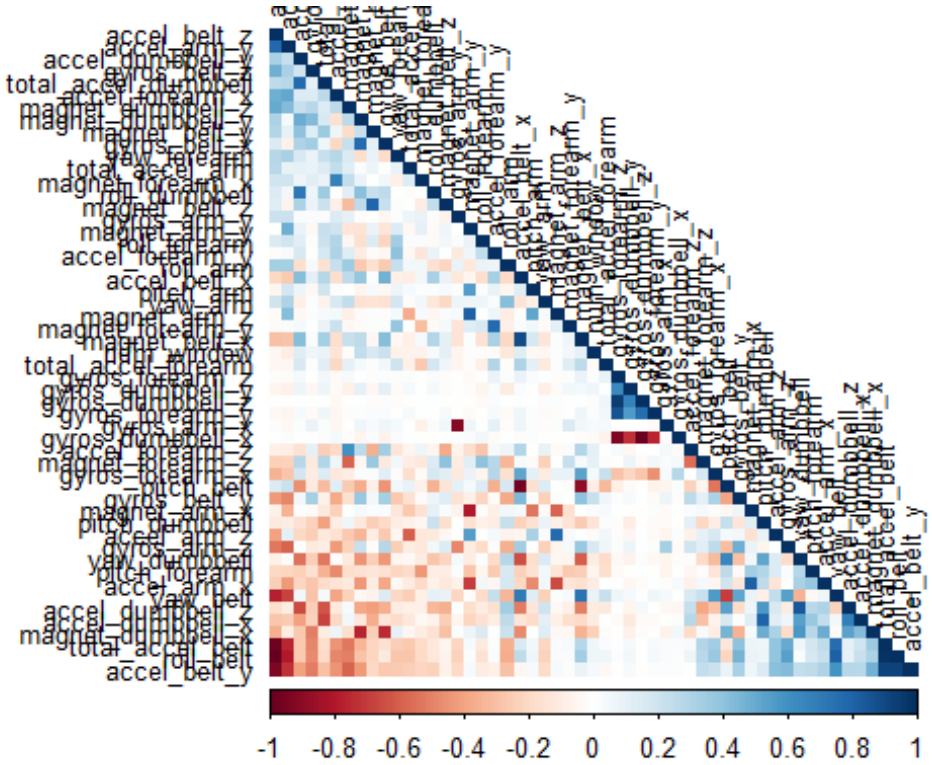
Divide the training data into a train and validation set

```
set.seed(100)
inTrain  <- createDataPartition(training$classe, p=0.75, list=FALSE)
Train <- training[inTrain, ]
Validate  <- training[-inTrain, ]
dim(Train)

## [1] 14718    54

dim(Validate)

## [1] 4904   54
```

Inter-variable correlation

If too many variables in the training model correlate it can be confusing without contributing anything.

```
corMatrix <- cor(Train[, -54])
```



There are not many that correlate, so we can proceed with classifier modelling. The models of to choose are Random Forests, Decision Tree and Gradient Boost models. The one with the best Accuracy will be chosen for the Test excercise.

Random Forest

```
set.seed(100)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=Train, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel
```

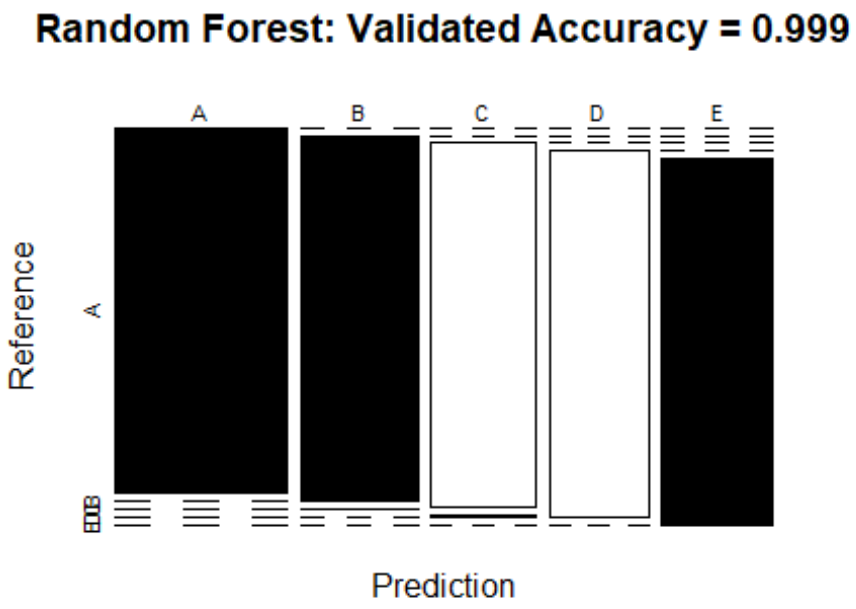
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.22%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4181    4    0    0    0 0.0009557945
## B    5 2839    3    1    0 0.0031601124
## C    0    3 2563    1    0 0.0015582392
## D    0    0   13 2398    1 0.0058043118
## E    0    0    0    2 2704 0.0007390983
```

Validate against the Validation Set:

```
predictRandForest <- predict(modFitRandForest, newdata=Validate)
confMatRandForest <- confusionMatrix(predictRandForest, Validate$classe)
confMatRandForest
```

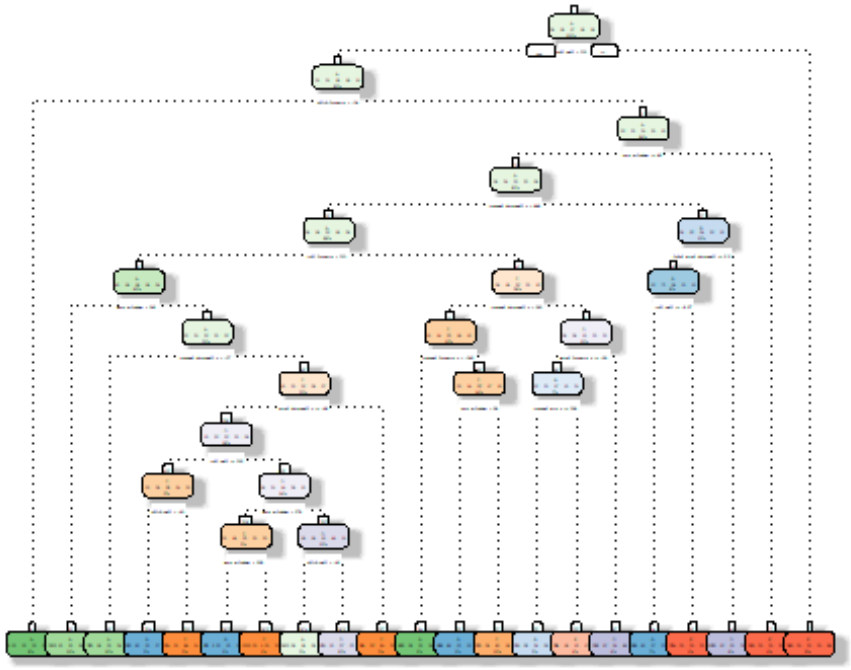
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    2    0    0
##          C    0    0  853    3    0
##          D    0    0    0  801    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.9976, 0.9997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9977   0.9963   1.0000
## Specificity            1.0000   0.9995   0.9993   1.0000   1.0000
## Pos Pred Value         1.0000   0.9979   0.9965   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   0.9995   0.9993   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1739   0.1633   0.1837
## Detection Prevalence   0.2845   0.1939   0.1746   0.1633   0.1837
## Balanced Accuracy      1.0000   0.9997   0.9985   0.9981   1.0000
```

Random Forest Validation Results = : ** 0.999**



Decision Tree model

```
set.seed(100)
modFitDecTree <- rpart(classe ~ ., data=Train, method="class")
fancyRpartPlot(modFitDecTree)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



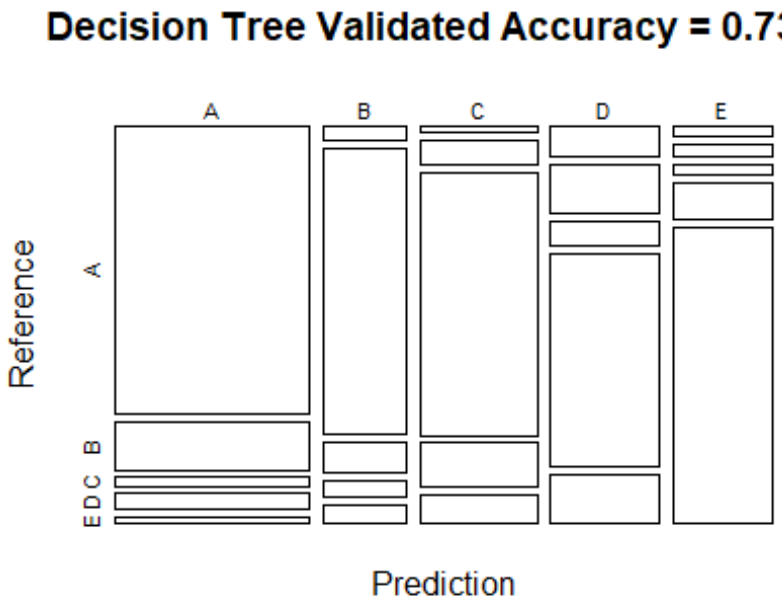
Valdidate against Validate data set:

```
predictDecTree <- predict(modFitDecTree, newdata=Validate, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, Validate$classe)
confMatDecTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1247  209   36   68   23
##          B   29  530   57   30   34
##          C   17   65  678  112   73
##          D   77  117   62  514  118
##          E   25   28   22   80  653
##
## Overall Statistics
##
##                Accuracy : 0.7386
##                  95% CI : (0.726, 0.7508)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6682
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8939   0.5585   0.7930   0.6393   0.7248
## Specificity            0.9042   0.9621   0.9341   0.9088   0.9613
## Pos Pred Value         0.7877   0.7794   0.7175   0.5788   0.8082
## Neg Pred Value         0.9554   0.9008   0.9553   0.9278   0.9395
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2543   0.1081   0.1383   0.1048   0.1332
## Detection Prevalence   0.3228   0.1387   0.1927   0.1811   0.1648
## Balanced Accuracy      0.8991   0.7603   0.8635   0.7740   0.8430
```

Decision Tree Accuracy = ** 0.7386 **



Boost model

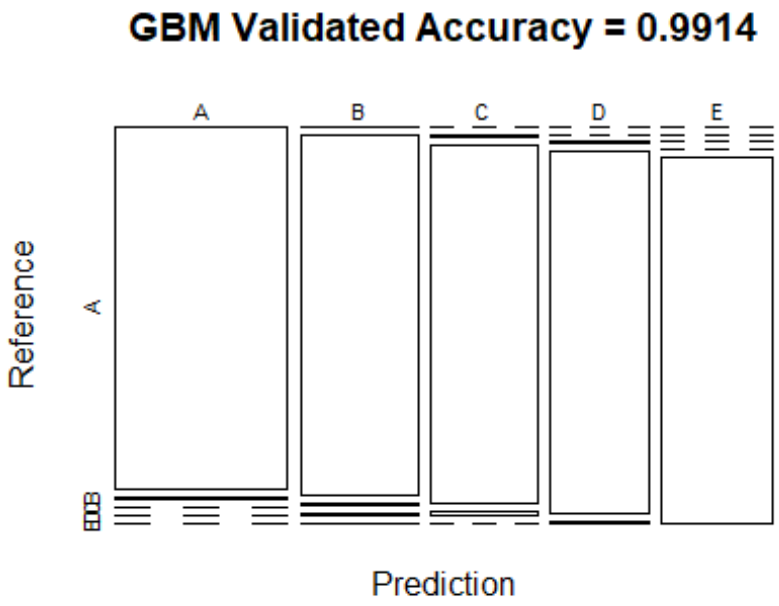
```
set.seed(100)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitGBM  <- train(classe ~ ., data=Train, method = "gbm",
                    trControl = controlGBM, verbose = FALSE)
modFitGBM$finalModel

## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

Valdiation on Validate dataset

```
predictGBM <- predict(modFitGBM, newdata=Validate)
confMatGBM <- confusionMatrix(predictGBM, Validate$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    9    0    0    0
##          B    3  933    8    2    1
##          C    0    7  846    8    0
##          D    0    0    1  794    3
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9914
##                  95% CI : (0.9884, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9892
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9831   0.9895   0.9876   0.9956
## Specificity            0.9974   0.9965   0.9963   0.9990   1.0000
## Pos Pred Value         0.9936   0.9852   0.9826   0.9950   1.0000
## Neg Pred Value         0.9991   0.9960   0.9978   0.9976   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1903   0.1725   0.1619   0.1829
## Detection Prevalence   0.2857   0.1931   0.1756   0.1627   0.1829
## Balanced Accuracy      0.9976   0.9898   0.9929   0.9933   0.9978
```

GBM Model Accurcay : ** 0.9914



Choose the best model:

Random Forest Accuracy: **0.999** Decision Tree Accuracy: ** 0.7386 **Gradient Boost Accuracy:** 0.9914 **

Run against Test data 20 records

```
predictTEST <- predict(modFitRandForest, newdata=testing)
predictTEST

##  [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
```