Machine Learning Peer-graded Assignment

A Steenekamp

08 November 2016

Setup environments

Executive Summary

Background

BY 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, 6 participants produce data from accelerometers on the belt, forearm, arm, and dumbell.They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Read more on HAR and the Weight-lifting dataset: http://groupware.les.inf.puc-rio.br/har#ixzz4PagQx474

Objective

The goal of this project is to predict the manner in which the participants did the exercise. This is the "classe" variable in the training set.A report describing how the model was built, how cross validation was used, what the expected out of sample error is, and why certain choices were made. The prediction model will be used to predict 20 different test cases.

Conclusion

Setup and record environment

Load libraries

```r
suppressMessages(library(randomForest))
suppressMessages(library(knitr))
#suppressMessages(library(plyr))
#suppressMessages(library(dplyr))
suppressMessages(library(ggplot2))
suppressMessages(library(datasets))
suppressMessages(library(corrplot))
#suppressMessages(library(AppliedPredictiveModeling))
suppressMessages(library(caret))
suppressMessages(library(rpart.plot))
suppressMessages(library(rattle))
suppressMessages(library(rpart))
#suppressMessages(library(pgmm))
```

```r
suppressMessages(library(gbm))
#suppressMessages(library(lubridate))
#suppressMessages(library(forecast))
#suppressMessages(library(e1071))
#suppressMessages(library(ElemStatLearn))
```

Read the data

```r
cache=TRUE
Work_Dir <- "C:/Users/Ada/Documents/Coursera/Mod8MachineLearning"
setwd(Work_Dir)
if (!file.exists("pml-training.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                  destfile = paste(Work_Dir, "pml-training.csv", sep="/"))
}
if (!file.exists("pml-testing.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                  destfile = paste(Work_Dir, "pml-testing.csv", sep="/"))
}
traindata <- data.frame(read.csv("pml-training.csv",na.strings=c("", "NA", "NULL")))
testdata <- data.frame(read.csv("pml-testing.csv",na.strings=c("", "NA", "NULL")))
```

Exploratory data analysis

```r
dim(traindata)
```

```
## [1] 19622   160
```

```r
dim(testdata)
```

```
## [1]  20 160
```

There is a very large number of variables and we need to determine which are the most important ones.

Create a new dataset with only the most important variables

Manual inspection

The first 7 columns can be removed as they are identifying data and will not contribute to the correctness of the exercise

```r
traindata<-traindata[-c(1:7)]
dim(traindata)
```

```
## [1] 19622   153
```

Remove columns with too many NA's

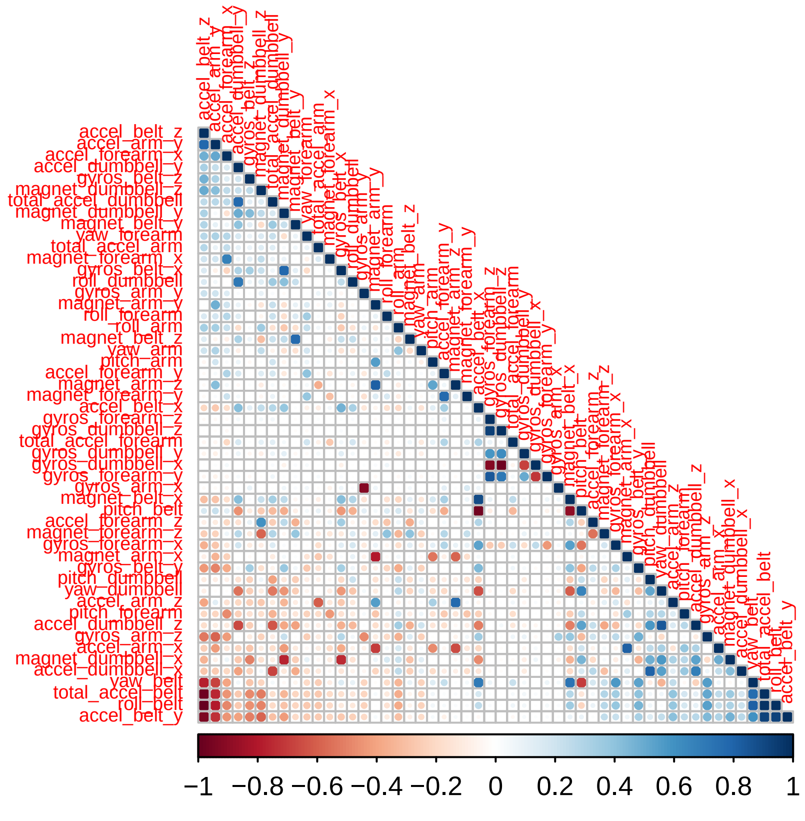
```
set.seed(12345)
trainNA <- traindata[ , colSums(is.na(traindata)) == 0]
dim(trainNA)
```

```
## [1] 19622    53
```

Determine highly correlated values

```
# first we need to remove the "classe" variable, since the cor function can only be performed on numeri
trainnum <- trainNA[,-c(53)]

cortrain<-cor(trainnum)
corrplot(cortrain,method="circle",order = "FPC", type = "lower", tl.cex = 0.6)
```



## Remove highly correlated values

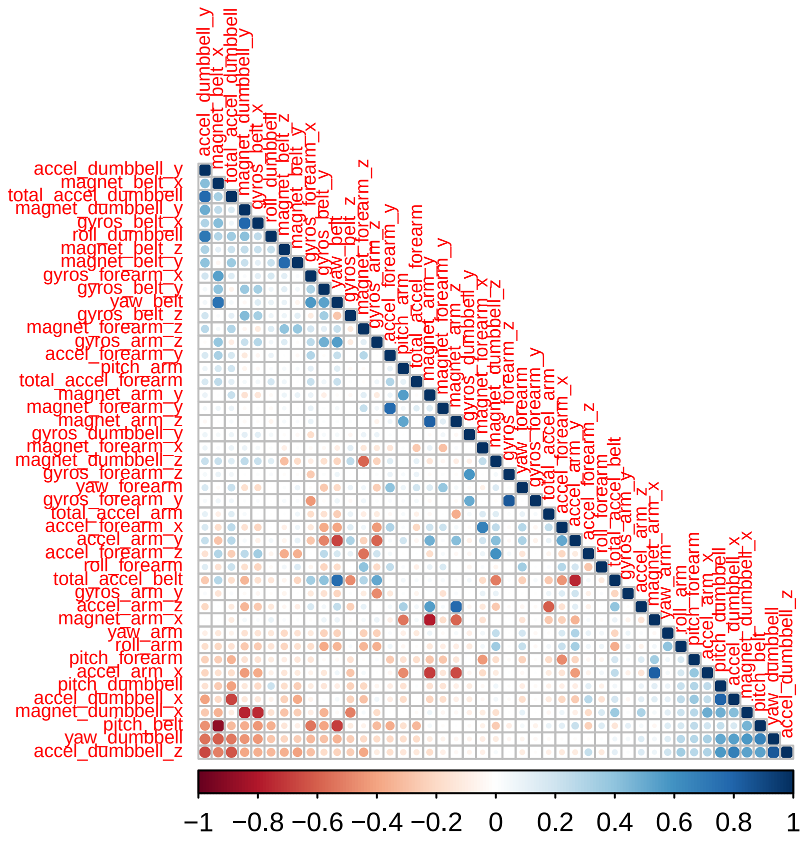
```
removecor = findCorrelation(cortrain, cutoff = .90, verbose = TRUE)
```

```
## Compare row 10  and column  1 with corr  0.992
##   Means:  0.27 vs 0.168 so flagging column 10
## Compare row 1  and column  9 with corr  0.925
##   Means:  0.25 vs 0.164 so flagging column 1
## Compare row 9  and column  4 with corr  0.928
##   Means:  0.233 vs 0.161 so flagging column 9
## Compare row 8  and column  2 with corr  0.966
##   Means:  0.245 vs 0.157 so flagging column 8
## Compare row 19  and column  18 with corr  0.918
##   Means:  0.091 vs 0.158 so flagging column 18
## Compare row 46  and column  31 with corr  0.914
##   Means:  0.101 vs 0.161 so flagging column 31
## Compare row 46  and column  33 with corr  0.933
##   Means:  0.083 vs 0.164 so flagging column 33
## All correlations <= 0.9
```

```r
trainrem = trainnum[,-removecor]
dim(trainrem)
```

```
## [1] 19622    45
```

```r
cortrain<-cor(trainrem)
corrplot(cortrain,method="circle",order = "FPC", type = "lower", tl.cex = 0.6)
```



```r
trainfin=trainNA[,-removecor]
dim(trainfin)
```

```
## [1] 19622    46
```

Create final training and testing datasets

```
inTrain = createDataPartition(y=trainfin$classe,p = 3/4,list=FALSE)
training=trainfin[inTrain,]
testing=trainfin[-inTrain,]
dim(training)
```

```
## [1] 14718    46
```

```
dim(testing)
```

```
## [1] 4904   46
```

Apply machine learning algorithms

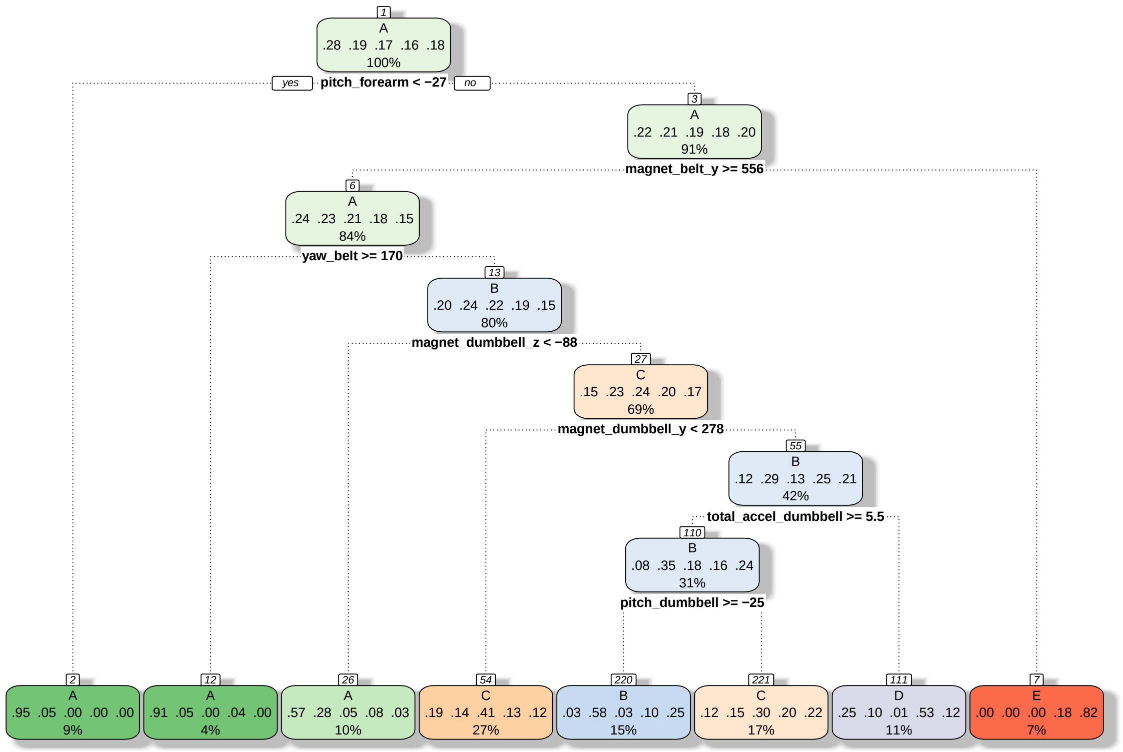
Decision Tree (method="rpart")

Note: Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

```
set.seed(12345)
fit1<-train(classe~.,method="rpart",data=training)
fit1$finalModel
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##     2) pitch_forearm< -26.65 1308    61 A (0.95 0.047 0 0 0) *
##     3) pitch_forearm>=-26.65 13410 10472 A (0.22 0.21 0.19 0.18 0.2)
##       6) magnet_belt_y>=555.5 12320  9385 A (0.24 0.23 0.21 0.18 0.15)
##        12) yaw_belt>=169.5 602    55 A (0.91 0.047 0 0.045 0) *
##        13) yaw_belt< 169.5 11718  8962 B (0.2 0.24 0.22 0.19 0.15)
##          26) magnet_dumbbell_z< -88.5 1502   649 A (0.57 0.28 0.046 0.076 0.033) *
##          27) magnet_dumbbell_z>=-88.5 10216  7720 C (0.15 0.23 0.24 0.2 0.17)
##            54) magnet_dumbbell_y< 277.5 4038  2366 C (0.19 0.14 0.41 0.13 0.12) *
##            55) magnet_dumbbell_y>=277.5 6178  4412 B (0.12 0.29 0.13 0.25 0.21)
##             110) total_accel_dumbbell>=5.5 4622  3011 B (0.083 0.35 0.18 0.16 0.24)
##               220) pitch_dumbbell>=-24.77322 2136   891 B (0.034 0.58 0.028 0.1 0.25) *
##               221) pitch_dumbbell< -24.77322 2486  1736 C (0.12 0.15 0.3 0.2 0.22) *
##             111) total_accel_dumbbell< 5.5 1556   734 D (0.25 0.1 0.0096 0.53 0.12) *
##       7) magnet_belt_y< 555.5 1090   199 E (0.0028 0.0028 0.0018 0.18 0.82) *
```

```
fancyRpartPlot(fit1$finalModel)
```



Rattle 2016−Nov−10 11:58:23 Ada

```
predrpart<-predict(fit1,testing)
confusionMatrix(predrpart,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 893 182  29  46  18
##          B  23 392  15 106 161
##          C 352 322 802 335 348
##          D 127  52   9 242  75
##          E   0   1   0  75 299
##
## Overall Statistics
##
##                Accuracy : 0.5359
##                  95% CI : (0.5218, 0.5499)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4177
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6401  0.41307   0.9380  0.30100  0.33185
## Specificity            0.9216  0.92288   0.6649  0.93585  0.98101
## Pos Pred Value         0.7646  0.56241   0.3715  0.47921  0.79733
## Neg Pred Value         0.8656  0.86760   0.9807  0.87224  0.86708
## Prevalence             0.2845  0.19352   0.1743  0.16395  0.18373
## Detection Rate         0.1821  0.07993   0.1635  0.04935  0.06097
## Detection Prevalence   0.2382  0.14213   0.4403  0.10298  0.07647
## Balanced Accuracy      0.7809  0.66797   0.8014  0.61842  0.65643
```

Random Forest (method="rf")

```
set.seed(12345)
fit2<-randomForest(classe~.,data=training,ntree=100, importance=TRUE)
#fit2$finalModel
predrf<-predict(fit2,testing)
confusionMatrix(predrf,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    7    0    0    0
##          B    1  938    5    0    0
##          C    0    4  848    6    2
##          D    0    0    2  798    2
##          E    0    0    0    0  897
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9915, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9884   0.9918   0.9925   0.9956
## Specificity            0.9980   0.9985   0.9970   0.9990   1.0000
## Pos Pred Value         0.9950   0.9936   0.9860   0.9950   1.0000
## Neg Pred Value         0.9997   0.9972   0.9983   0.9985   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1913   0.1729   0.1627   0.1829
## Detection Prevalence   0.2857   0.1925   0.1754   0.1635   0.1829
## Balanced Accuracy      0.9986   0.9934   0.9944   0.9958   0.9978
```

Boosting with trees (method = "gbm")

```r
set.seed(12345)
controlgbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
fit3<-train(classe~.,method="gbm",data=training,verbose=FALSE,trControl = controlgbm)
```

```
## Loading required package: plyr
```

```r
#fit3$finalModel
predgbm<-predict(fit3,testing)
confusionMatrix(predgbm,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1376   29    0    1    4
##          B   13  880   23    5    5
##          C    4   36  818   27   10
##          D    2    2   13  766   15
##          E    0    2    1    5  867
## 
## Overall Statistics
##                                          
##                Accuracy : 0.9598         
##                  95% CI : (0.954, 0.9652)
##     No Information Rate : 0.2845         
##     P-Value [Acc > NIR] : < 2.2e-16      
##                                          
##                   Kappa : 0.9492         
##  Mcnemar's Test P-Value : 5.378e-05      
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9864   0.9273   0.9567   0.9527   0.9623
## Specificity            0.9903   0.9884   0.9810   0.9922   0.9980
## Pos Pred Value         0.9759   0.9503   0.9140   0.9599   0.9909
## Neg Pred Value         0.9946   0.9827   0.9908   0.9907   0.9916
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2806   0.1794   0.1668   0.1562   0.1768
## Detection Prevalence   0.2875   0.1888   0.1825   0.1627   0.1784
## Balanced Accuracy      0.9883   0.9578   0.9689   0.9725   0.9801
```

Predict the test cases

The accuracy rate for the three methods are:

1) Decision Tree 49%, 51% expected out of error rate
2) Random Forest 99%, 1% expected out of error rate

3) Boosting 96%, 4% expected out of error rate

The most accurate method is Random Forest

```r
predtest <- predict(fit2, newdata=testdata)
predtest
```

```
##  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
```