Machine Learning Project

Cassie

October 9, 2017

This report will look at athletic tech wear data and, using training data, build a model to predict which exercise is being performed based on the other metrics.

Set working directory

```
setwd("~/Coursera/R Repository/Machine Learning")
```

Load Packages

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.3.3
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.3.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.3.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

```
## 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(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

Reading in the training and testing data

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

Finding columns with NAs

```
NAs <- colMeans(is.na(training))
table(NAs)
```

```
## NAs
##                0 0.979308938946081 0.979359902150647 0.979410865355213
##               60                67                 1                 1
## 0.979512791764346 0.979563754968912 0.979767607787178 0.979818570991744
##                1                 4                 1                 4
##  0.97986953419631 0.980939761492203 0.983233105697686 0.983284068902253
##                2                 2                 1                 1
## 0.983385995311385 0.983538884925084  0.98358984812965 0.983640811334217
##                2                 1                 4                 2
##                1
##                6
```

Removing columns with NAs

```
sum(as.logical(NAs))
```

```
## [1] 100
```

```
colNAs <- !NAs
sum(colNAs)
```

```
## [1] 60
```

```
train.NA <- training[colNAs]
str(train.NA)
```

```
## 'data.frame':    19622 obs. of  60 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name           : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2
2 2 2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 132308423
2 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 4403
90 484323 484434 ...
##  $ cvtd_timestamp      : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9
9 9 9 ...
##  $ new_window          : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window          : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.
4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.0
2 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.
03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.0
2 -0.02 ...
```

```
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.
8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1
1 ...
```

Getting rid of other unecessary columns

```
others <- grep("^X$|user_name|timestamp|window", names(train.NA))
train.clean <- train.NA[-others]
```

Data partioning for training and testing

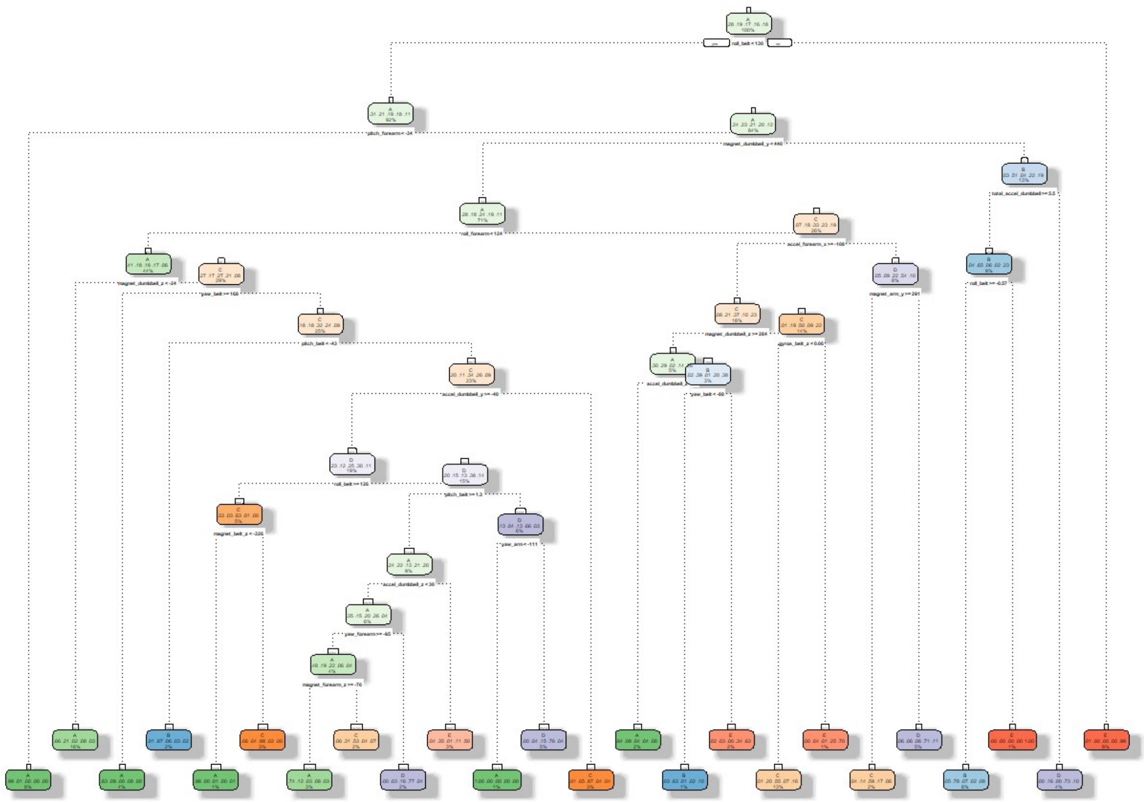
```
set.seed (1)
inTrain <- createDataPartition(train.clean$classe, p=0.6, list=FALSE)
train <- train.clean[inTrain, ]
test <- train.clean[-inTrain, ]
```

Testing different prediction models to determine which one has the highest accuracy.

1. Prediction with decision tree

```
model.1 <- rpart(classe~., data=train, method="class")
fancyRpartPlot(model.1)
```

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



Rattle 2017-Oct-09 15:19:54 Cassi

```
predict.1 <- predict(model.1, test, type ="class")
confusionMatrix(predict.1, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2064  313   34  154   44
##          B   49  708   60   23   71
##          C   70  305 1168  125  188
##          D   31   93  100  863  106
##          E   18   99    6  121 1033
##
## Overall Statistics
##
##                Accuracy : 0.7438
##                  95% CI : (0.734, 0.7534)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6744
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9247  0.46640   0.8538   0.6711   0.7164
## Specificity            0.9029  0.96792   0.8938   0.9497   0.9619
## Pos Pred Value         0.7911  0.77717   0.6293   0.7234   0.8089
## Neg Pred Value         0.9679  0.88320   0.9666   0.9364   0.9377
## Prevalence             0.2845  0.19347   0.1744   0.1639   0.1838
## Detection Rate         0.2631  0.09024   0.1489   0.1100   0.1317
## Detection Prevalence   0.3325  0.11611   0.2366   0.1521   0.1628
## Balanced Accuracy      0.9138  0.71716   0.8738   0.8104   0.8391
```

2. Prediction with random forest

```
model.2 <- randomForest(classe~., data=train, method = "class")
predict.2 <- predict(model.2, test, type="class")
confusionMatrix(predict.2, test$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    5    0    0    0
##          B    5 1509    6    0    0
##          C    1    4 1359   14    0
##          D    0    0    3 1271    5
##          E    0    0    0    1 1437
## 
## Overall Statistics
## 
##                Accuracy : 0.9944          
##                  95% CI : (0.9925, 0.9959)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9929          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9973   0.9941   0.9934   0.9883   0.9965
## Specificity            0.9991   0.9983   0.9971   0.9988   0.9998
## Pos Pred Value         0.9978   0.9928   0.9862   0.9937   0.9993
## Neg Pred Value         0.9989   0.9986   0.9986   0.9977   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2837   0.1923   0.1732   0.1620   0.1832
## Detection Prevalence   0.2843   0.1937   0.1756   0.1630   0.1833
## Balanced Accuracy      0.9982   0.9962   0.9952   0.9936   0.9982
```

The random forest model has a higher accuracy rate so we will use that for the prediction model.