Prediction of Exercise Class

Justin Rice

Executive Summary

This study was commissioned to predict the class of how a subject performed an exercise. The data used for this project was collected from a range of wireless activity wristbands. The wristbands were worn be the subjects as they performed excercies in 5 different ways. To study the data I broke the source data into a training and testing set. I experimented with 2 training models; Decision Tree and Random Forest. The decision tree performed very poorly which is likely to be caused by a lack of homogeneity in the data. We know that random forest is one of the most accurate models and it proved itself here with this data.

Getting and Cleaning

```r
library(knitr)
library(dplyr)
library(plyr)
library(ggplot2)
library(gridExtra)
library(caret)
library(party)
#library(e1071)

set.seed(2016)

#download the data if it doesn't already exist
if(!file.exists("pml-training.csv")) {
  trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
  download.file(trainURL, destfile="pml-training.csv")
}

if(!file.exists("pml-testing.csv")) {
  trainURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
  download.file(trainURL, destfile="pml-testing.csv")
}

#Load the data into R
  pmlVal <- read.csv("pml-testing.csv")
  pmlData <- read.csv("pml-training.csv")

#Remove all the columns that include NA values
NAcols <- colSums(is.na(pmlData)) == 0
Validcolumns <- sum(NAcols == TRUE)
pmlData <- pmlData[, NAcols]


#Remove columns that are not meaningful predictors
nzv <- nearZeroVar(pmlData, saveMetrics = T)
pmlData <- pmlData[, nzv$nzv==F]

#pmlData <- head(pmlData, 1000)
str(pmlData)
```

```
## 'data.frame':    19622 obs. of  59 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 1323084232
1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 44039
0 484323 484434 ...
##  $ cvtd_timestamp      : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9
9 9 ...
##  $ 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 -0.02
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 -0.0
3 ...
##  $ 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 -0.02
-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
...
```

After the data was loaded, it was important to remove the NA columns as they could mislead the results. I also removed the columns that did not contribute any variance to the prediction model.

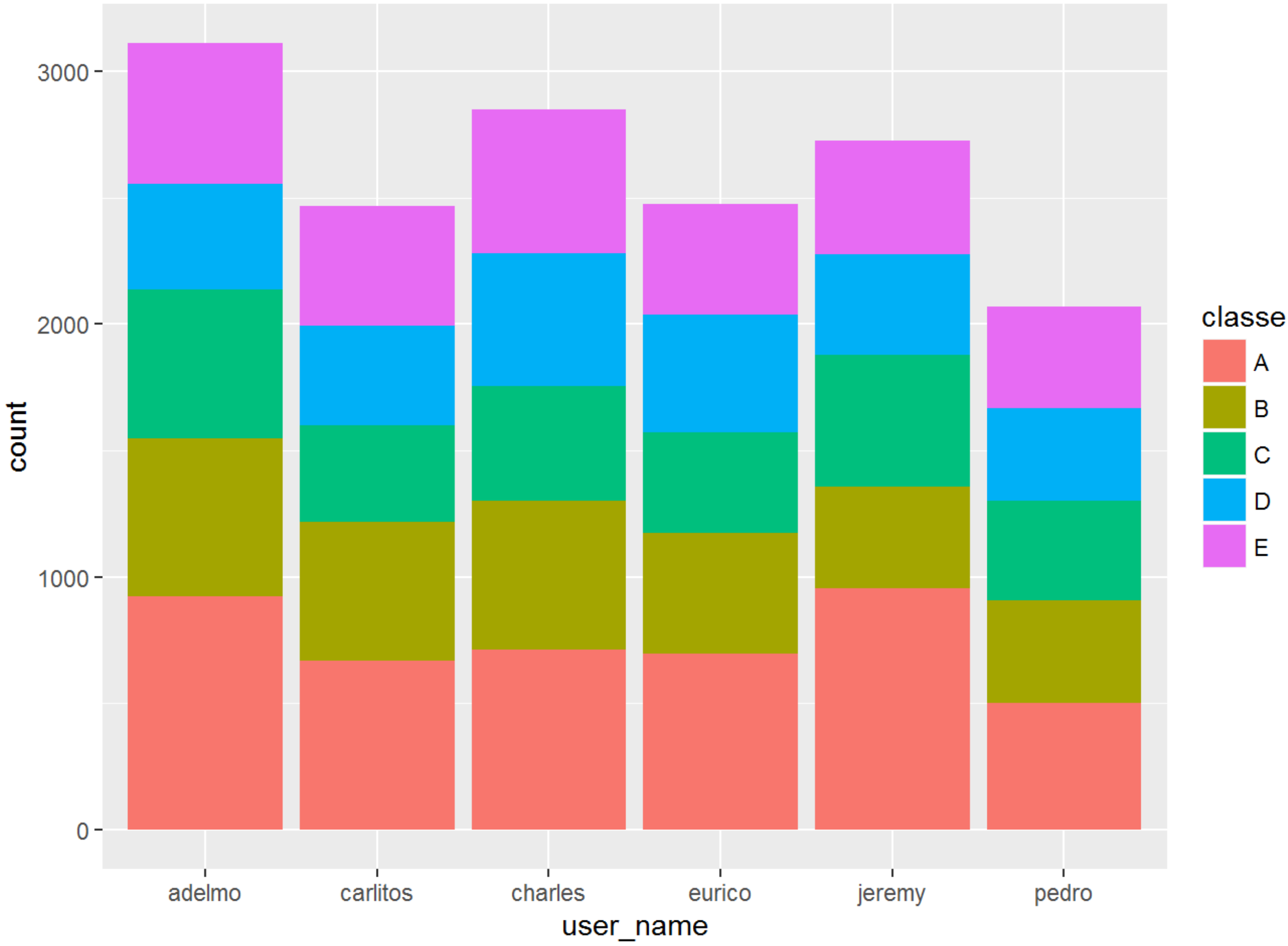
Setup Cross-Validation

```
partition <- createDataPartition(pmlData$classe, p=0.8, list=F)
inTrain <- pmlData[partition, ]
inTest <- pmlData[-partition, ]
i <- data.frame("Total"=c(nrow(inTrain), nrow(inTest)))
DT <- matrix(NA, nrow=2, ncol=5)
TR <- as.data.frame(table(inTrain$classe))
TE <- as.data.frame(table(inTest$classe))
DT[1, ] <- TR[,2]
DT[2, ] <- TE[,2]
g <- as.data.frame(DT)
colnames(g) <- c("A", "B", "C", "D", "E")
e<- cbind(g, i)
f <- colSums(e)
r <-  rbind(e, f)
rownames(r) <- c('Training Data','Testing Data','Total')
kable(r)
```

	A	B	C	D	E	Total
Training Data	4464	3038	2738	2573	2886	15699
Testing Data	1116	759	684	643	721	3923
Total	5580	3797	3422	3216	3607	19622

This shows the break-down of the outcome variable.

```
qplot(user_name, data=inTrain, fill=classe)
```



Build Decision Tree Model

In this section

```
treeFit <- train(classe ~., method="rpart", data=inTrain)
treePred <- predict(treeFit, newdata=inTest)
treeMatrix <- confusionMatrix(inTest$classe, treePred)
treeMatrix$overall['Accuracy']
```

```
##  Accuracy
## 0.6614836
```

```
treeMatrix$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    1  758    0    0    0
##          C    0    0    0    0  684
##          D    0    0    0    0  643
##          E    0    0    0    0  721
```

Random Forest

```
ctrl <- trainControl(allowParallel = T, method="cv", number=4)
rfFit <- train(classe ~., method="rf", data=inTrain, trControl=ctrl)
rfPred <- predict(rfFit, newdata=inTest)

rfMatrix <- confusionMatrix(inTest$classe, rfPred)
rfMatrix$overall['Accuracy']
```

```
##  Accuracy
## 0.9997451
```

```
rfMatrix$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    1  758    0    0    0
##          C    0    0  684    0    0
##          D    0    0    0  643    0
##          E    0    0    0    0  721
```

Out of Sample Error

```
err <- 1-rfMatrix$overall['Accuracy']
err
```

```
##    Accuracy
## 0.000254907
```

The expected Out of Sample Error is 0.02% which is very good for prediction. Therefore we will use the random forest.

Predict New Data

```
pmlVal <- pmlVal[, NAcols]
pmlVal <- pmlVal[, nzv$nzv==F]
testing <- pmlVal[, 1:58]

testPred <- predict(rfFit, newdata=testing)
```