Practical Machine Learning Assignment

Paul Martinez

2023-06-08

Executive Summary

The goal of this project is to predict the manner in which participants did exercise using devices such as Jawbone Up, Nike FuelBand, and Fitbit. This report describes in detail how the model was built, how it was cross validated, and what the expected sample error is.

First Step is to load libraries and dataset

```
library(lattice)
library(ggplot2)
library(plyr)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

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

```
training.raw <- read.csv('~/pml-training.csv')
testing.raw <- read.csv('~/pml-testing.csv')

dim(training.raw)
```

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

Given the amount of missing data (NAs). I will clean the data to not include missing on the variables of interest.

```
maxNAPerc = 20
maxNACount <- nrow(training.raw) / 100 * maxNAPerc
removeColumns <- which(colSums(is.na(training.raw) | training.raw=="") > maxNACount)
training.cleaned01 <- training.raw[,-removeColumns]
testing.cleaned01 <- testing.raw[,-removeColumns]
```

I will also remove data variable that are not being considered.

```
removeColumns <- grep("timestamp", names(training.cleaned01))
training.cleaned02 <- training.cleaned01[,-c(1, removeColumns )]
testing.cleaned02 <- testing.cleaned01[,-c(1, removeColumns )]
```

Converting all factors to integers

```
classeLevels <- levels(training.cleaned02$classe)
training.cleaned03 <- data.frame(data.matrix(training.cleaned02))
training.cleaned03$classe <- factor(training.cleaned03$classe, labels=1)
testing.cleaned03 <- data.frame(data.matrix(testing.cleaned02))
```

The following is the clean dataset.

```
training.cleaned <- training.cleaned03
testing.cleaned <- testing.cleaned03
```

Exploratory data analyses

```
set.seed(19791108)
library(caret)

classeIndex <- which(names(training.cleaned) == "classe")

partition <- createDataPartition(y=training.cleaned$classe, p=0.75, list=FALSE)
training.subSetTrain <- training.cleaned[partition, ]
training.subSetTest <- training.cleaned[-partition, ]
```

Here I am exploring what variables have high correlations with the classe:

```
correlations <- cor(training.subSetTrain[, -classeIndex], as.numeric(training.subSetTrain$classe))
bestCorrelations <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
bestCorrelations
```

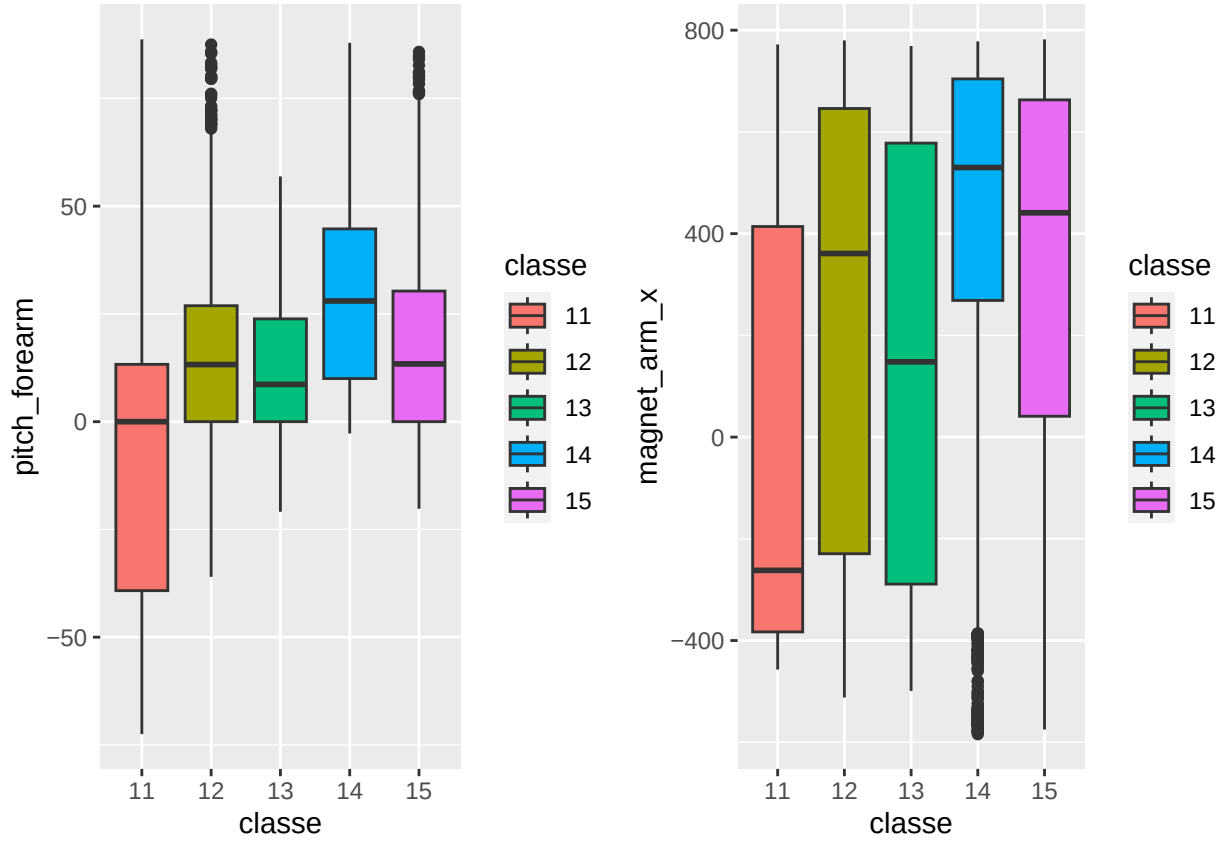
```
##             Var1 Var2     Freq
## 44 pitch_forearm    A 0.336018
```

```
library(Rmisc)
library(ggplot2)

p1 <- ggplot(training.subSetTrain, aes(classe,pitch_forearm)) +
  geom_boxplot(aes(fill=classe))

p2 <- ggplot(training.subSetTrain, aes(classe, magnet_arm_x)) +
  geom_boxplot(aes(fill=classe))

multiplot(p1,p2,cols=2)
```



No clear correlations can be established from the analysis above.

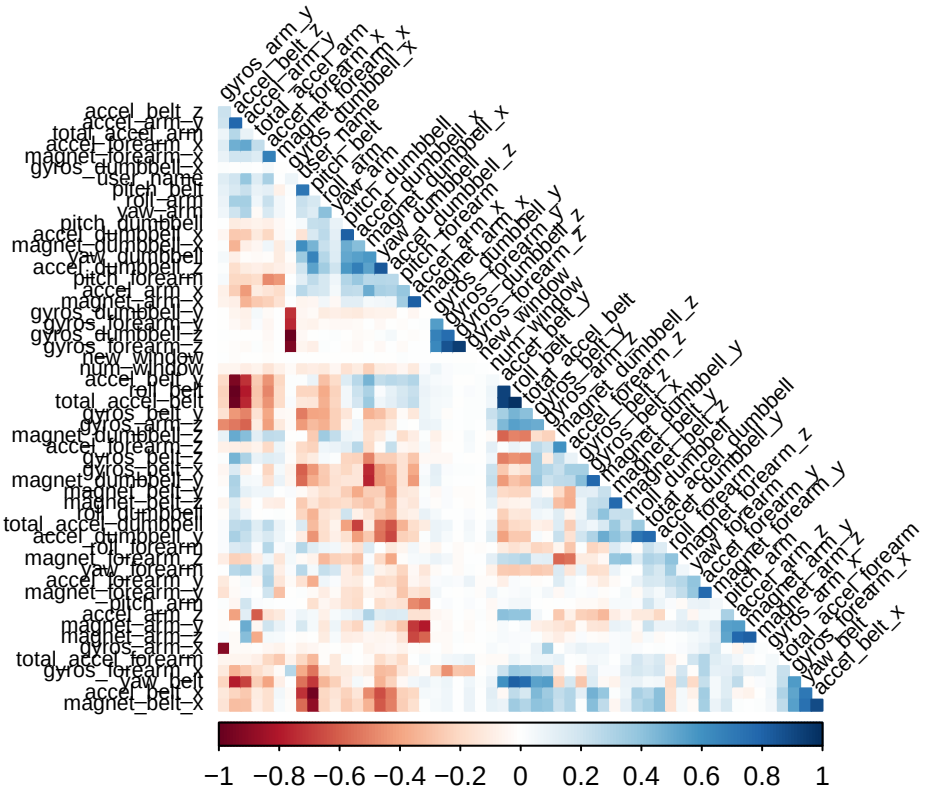
Model selection

I will identify variables with high correlations so they can be exclude them from the training.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
correlationMatrix <- cor(training.subSetTrain[, -classeIndex])
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.9, exact=TRUE)
excludeColumns <- c(highlyCorrelated, classeIndex)
corrplot(correlationMatrix, method="color", type="lower", order="hclust", tl.cex=0.70, tl.col="black",
```



Given the results we can see that some features are correlated.I will include a model that excludes these features.

```
pcaPreProcess.all <- preProcess(training.subSetTrain[, -classeIndex], method = "pca", thresh = 0.99)
training.subSetTrain.pca.all <- predict(pcaPreProcess.all, training.subSetTrain[, -classeIndex])
training.subSetTest.pca.all <- predict(pcaPreProcess.all, training.subSetTest[, -classeIndex])
testing.pca.all <- predict(pcaPreProcess.all, testing.cleaned[, -classeIndex])


pcaPreProcess.subset <- preProcess(training.subSetTrain[, -excludeColumns], method = "pca", thresh = 0.9
training.subSetTrain.pca.subset <- predict(pcaPreProcess.subset, training.subSetTrain[, -excludeColumns]
training.subSetTest.pca.subset <- predict(pcaPreProcess.subset, training.subSetTest[, -excludeColumns])
testing.pca.subset <- predict(pcaPreProcess.subset, testing.cleaned[, -classeIndex])
```

Next, we will include random forest training. We'll use 50 trees and time each of the 2 random forest models to see if when all else is equal.

```
library(randomForest)

ntree <- 50

start <- proc.time()
rfMod.cleaned <- randomForest(
  x=training.subSetTrain[, -classeIndex],
  y=training.subSetTrain$classe,
  xtest=training.subSetTest[, -classeIndex],
```

```
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start
```

```
##    user  system elapsed
##   20.95    0.37   21.33
```

```
start <- proc.time()
rfMod.exclude <- randomForest(
  x=training.subSetTrain[, -excludeColumns],
  y=training.subSetTrain$classe,
  xtest=training.subSetTest[, -excludeColumns],
  ytest=training.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE) #do.trace=TRUE
proc.time() - start
```

```
##    user  system elapsed
##   22.49    0.57   23.06
```

Model examination

Now that we have 2 trained models, we will check the accuracies of each.

```
rfMod.cleaned
```

```
##
## Call:
##  randomForest(x = training.subSetTrain[, -classeIndex], y = training.subSetTrain$classe,      xtest =
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.46%
## Confusion matrix:
##      11   12   13   14   15  class.error
## 11 4184    0    0    0    1 0.0002389486
## 12   10 2830    8    0    0 0.0063202247
## 13    0   14 2552    1    0 0.0058433970
## 14    0    0   22 2389    1 0.0095356551
## 15    1    1    0    8 2696 0.0036954915
##                 Test set error rate: 0.35%
## Confusion matrix:
##      11  12  13  14  15 class.error
## 11 1395   0   0   0   0 0.000000000
## 12    3 942   4   0   0 0.007376185
## 13    0   2 853   0   0 0.002339181
## 14    0   0   2 801   1 0.003731343
## 15    0   0   0   5 896 0.005549390
```

```
rfMod.cleaned.training.acc <- round(1-sum(rfMod.cleaned$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rfMod.cleaned.training.acc)
```

```
## [1] "Accuracy on training: 0.974"
```

```
rfMod.cleaned.testing.acc <- round(1-sum(rfMod.cleaned$test$confusion[, 'class.error']),3)
paste0("Accuracy on testing: ",rfMod.cleaned.testing.acc)
```

```
## [1] "Accuracy on testing: 0.981"
```

```
rfMod.exclude
```

```
##
## Call:
##  randomForest(x = training.subSetTrain[, -excludeColumns], y = training.subSetTrain$classe,      xtes
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.45%
## Confusion matrix:
##      11   12   13   14   15  class.error
## 11 4184    1    0    0    0 0.0002389486
## 12    6 2837    5    0    0 0.0038623596
## 13    1   17 2546    3    0 0.0081807557
## 14    0    0   23 2384    5 0.0116086235
## 15    0    0    0    5 2701 0.0018477458
##                 Test set error rate: 0.33%
## Confusion matrix:
##      11  12  13  14  15 class.error
## 11 1395   0   0   0   0 0.000000000
## 12    2 944   3   0   0 0.005268704
## 13    0   7 848   0   0 0.008187135
## 14    0   0   2 801   1 0.003731343
## 15    0   0   0   1 900 0.001109878
```

```
rfMod.exclude.training.acc <- round(1-sum(rfMod.exclude$confusion[, 'class.error']),3)
paste0("Accuracy on training: ",rfMod.exclude.training.acc)
```

```
## [1] "Accuracy on training: 0.974"
```

```
rfMod.exclude.testing.acc <- round(1-sum(rfMod.exclude$test$confusion[, 'class.error']),3)
paste0("Accuracy on testing: ",rfMod.exclude.testing.acc)
```

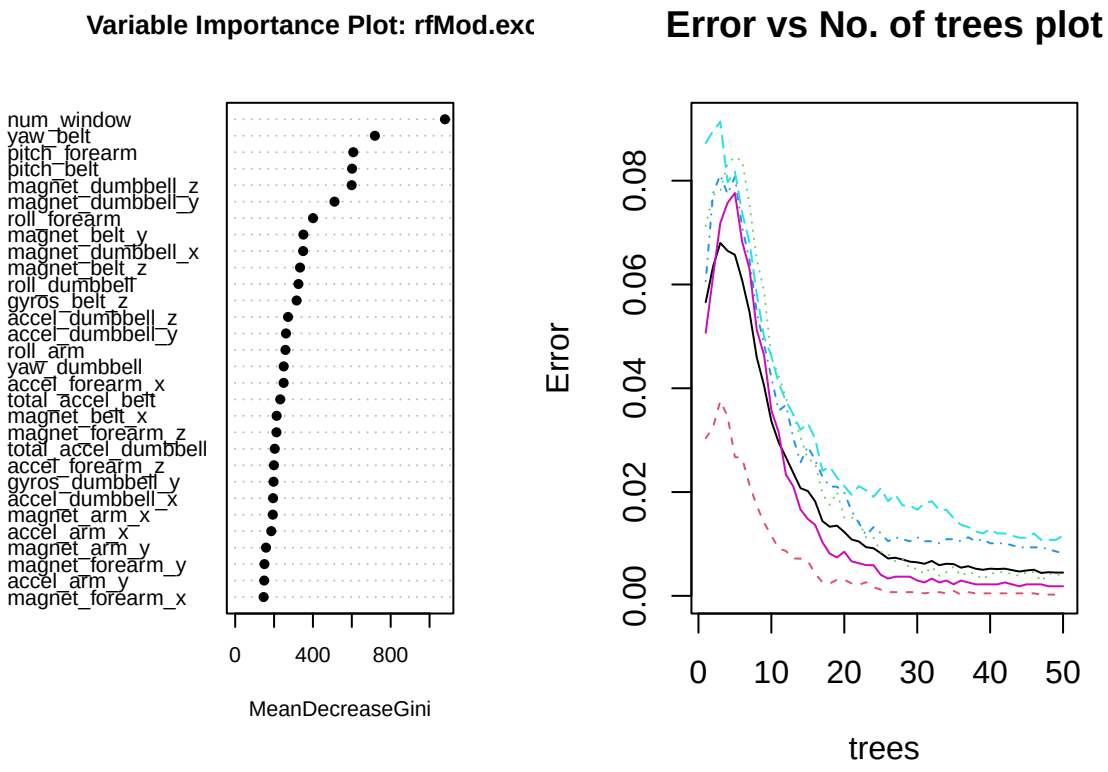
```
## [1] "Accuracy on testing: 0.982"
```

Conclusion

The `rfMod.exclude` perform's slightly better then the 'rfMod.cleaned'. I will use the `rfMod.exclude` model as the best model to use for predicting the test set.Because with an accuracy of 98.7%, the estimated OOB error rate of 0.23% this is the best model.

Before doing the final prediction we will examine the chosen model more in depth using some plots

```
par(mfrow=c(1,2))
varImpPlot(rfMod.exclude, cex=0.7, pch=16, main='Variable Importance Plot: rfMod.exclude')
plot(rfMod.exclude, cex=0.7, main='Error vs No. of trees plot')
```




```
par(mfrow=c(1,1))
```

To really look in depth at the distances between predictions we can use MDSplot and cluster prediction and results

Test results

Although we've chosen the `rfMod.exclude` it's still nice to see what the other model would predict on the final test set. Let's look at predictions for all models on the final test set.

```
predictions <- t(cbind(
    exclude=as.data.frame(predict(rfMod.exclude, testing.cleaned[, -excludeColumns]), optional=TRUE),
    cleaned=as.data.frame(predict(rfMod.cleaned, testing.cleaned), optional=TRUE)
))
predictions
```

```
##         1    2    3    4    5    6    7    8    9    10   11   12   13   14
```

```
## exclude "12" "11" "12" "11" "11" "15" "14" "12" "11" "11" "12" "13" "12" "11"
## cleaned "12" "11" "12" "11" "11" "15" "14" "12" "11" "11" "12" "13" "12" "11"
##         15   16   17   18   19   20
## exclude "15" "15" "11" "12" "12" "12"
## cleaned "15" "15" "11" "12" "12" "12"
```

Overall, `rfMod.exclude` is the best model.