Practical Machine Learning

Background

Using devices such as JawboneUp, NikeFuelBand, and Fitbitit 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.

The goal of this project is to use the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict where the data came from.

Get Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

```
library(caret)
```

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

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

```
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
target <- "pml_training.csv"
download.file(url, destfile = target)
trainRaw <- read.csv(target)
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
target <- "pml_testing.csv"
download.file(url, destfile = target)
testRaw <- read.csv(target)
```

Clean Data

We need to split the data up in order to have a pure data set and a validation data set. Use only relevent columns with nonzero colums

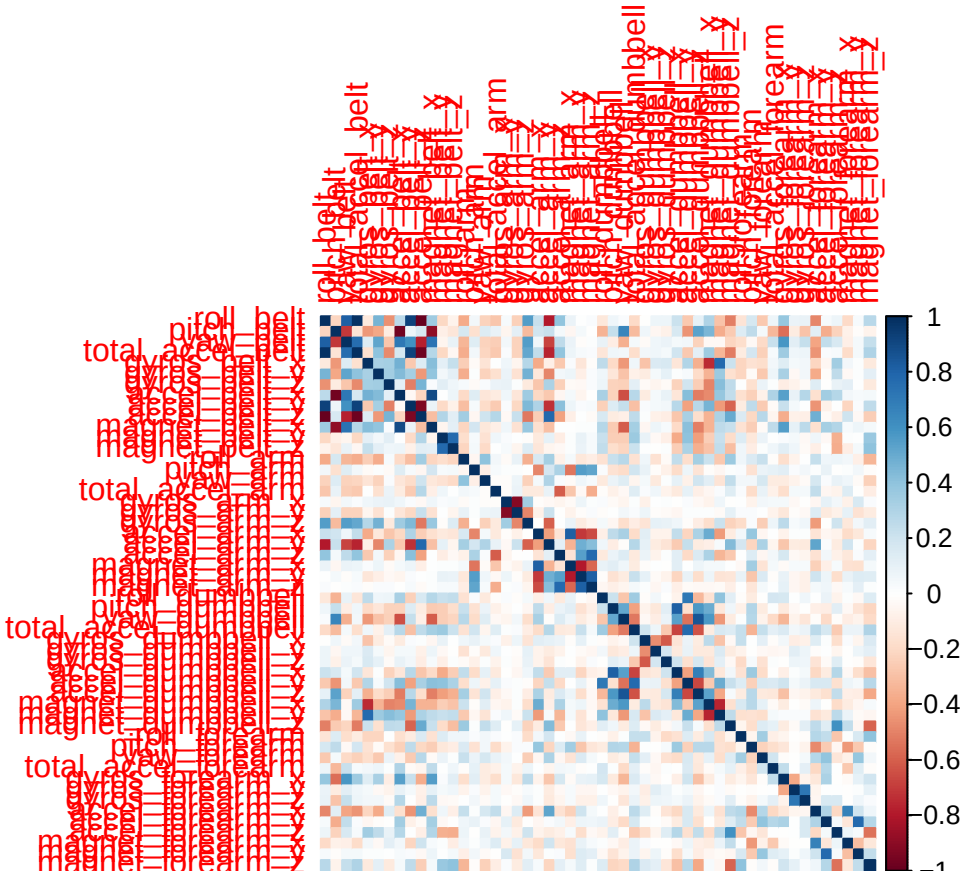
```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
```

```
set.seed(22519) # For reproducibile purpose
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

Data Manipulation

First lets visualize correlation

```
library(corrplot)
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color")
```



Our first model will be done using a random forest. Beware this takes a couple of minutes to run...

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=250)
```

```
## Loading required package: randomForest
```

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

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

```
##
## Attaching package: 'randomForest'
```

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

```
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9901727  0.9875673
##   27    0.9917015  0.9895017
##   52    0.9840572  0.9798282
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Model Results

```
predict <- predict(modelRf, testData)
confusionMatrix(testData$classe, predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    0    1
##          B    5 1131    3    0    0
##          C    0    0 1021    5    0
##          D    0    0   13  949    2
##          E    0    0    1    6 1075
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9915, 0.9957)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9970   1.0000   0.9836   0.9885   0.9972
## Specificity            0.9998   0.9983   0.9990   0.9970   0.9985
## Pos Pred Value         0.9994   0.9930   0.9951   0.9844   0.9935
## Neg Pred Value         0.9988   1.0000   0.9965   0.9978   0.9994
## Prevalence             0.2851   0.1922   0.1764   0.1631   0.1832
## Detection Rate         0.2843   0.1922   0.1735   0.1613   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9992   0.9913   0.9927   0.9979
```

```
accuracy <- postResample(predict, testData$classe)
accuracy
```

```
##  Accuracy     Kappa
## 0.9938828 0.9922620
```

```
o <- 1 - as.numeric(confusionMatrix(testData$classe, predict)$overall[1])
o
```

```
## [1] 0.006117247
```

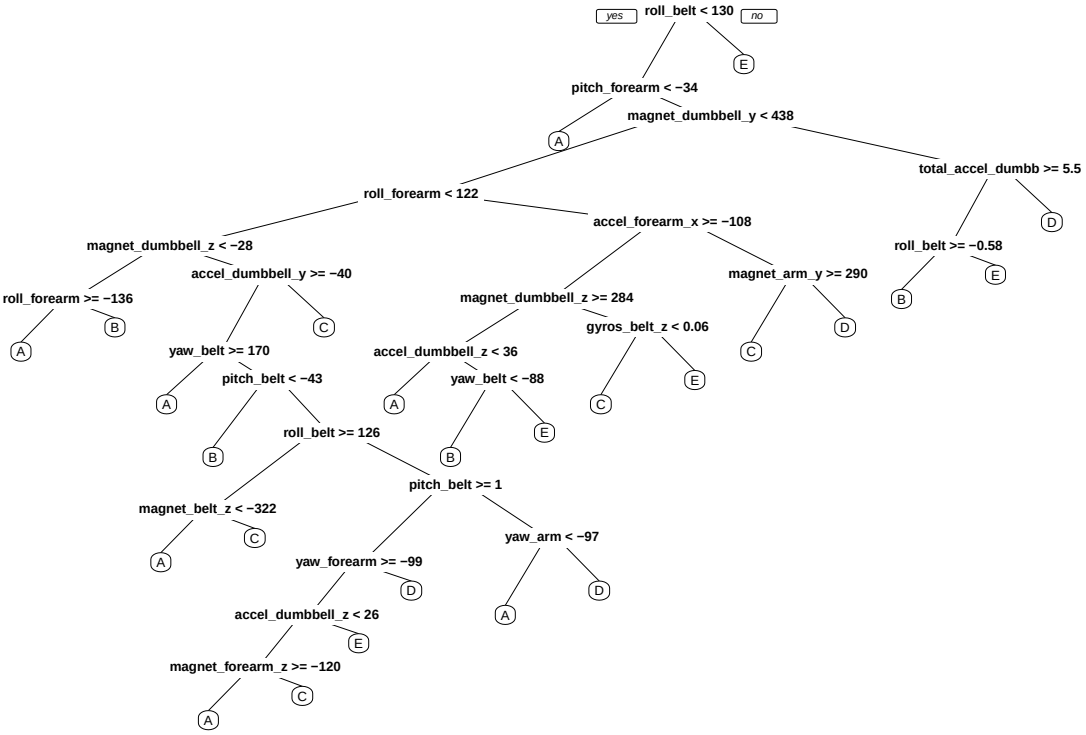
Now, with the above information we can predict...

```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Decision Tree

```
library(rpart)
library(rpart.plot)
library(randomForest)
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel) # fast plot
```



Conclusion

The Random Forest model has an accuracy of about 99 percent.