Course Project

student0129 (TJ)

March 3, 2017

Background

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, my goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

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).

Data Processing

In this section we load the data, clean the data, and create training and testing tests that can be used for the analysis.

```
#load data
setwd("~/MOOC - Data Science/Part 8")
training = read.csv("pml-training.csv", na.strings = c("NA", ""))
testing = read.csv("pml-testing.csv", na.strings = c("NA", ""))

#check if all attributes are the same
all.equal(names(training), names(testing))
```

```
## [1] "1 string mismatch"
```

```
#identify differences in attribute names
setdiff(names(training), names(testing))
```

```
## [1] "classe"
```

```
setdiff(names(testing), names(training))
```

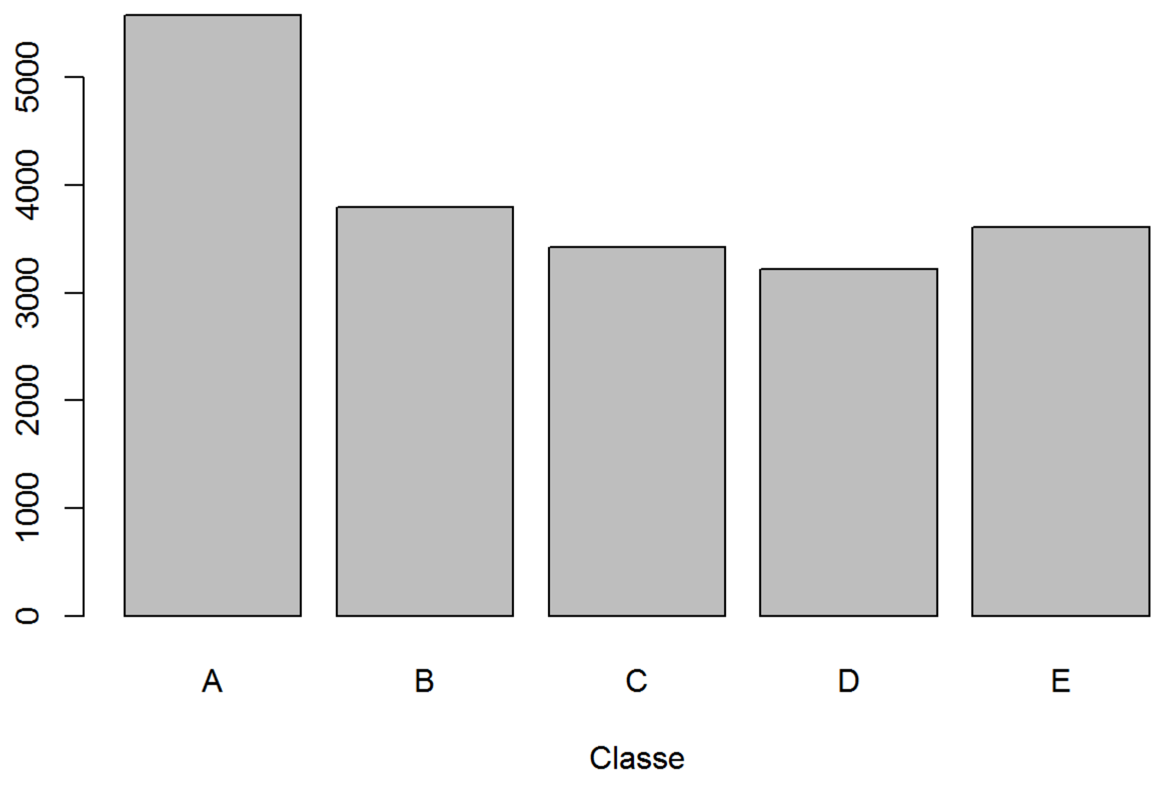
```
## [1] "problem_id"
```

```
#grab all attribute names
naColNames <- c()

#remove attrbiutes that have NULL values
for (i in names(training)) {
        if (sum(is.na(training[[i]])) != 0) {
                naColNames <- c(naColNames, i)
        }
}

#create clean training and resting sets
trainingClean <- training[,!(names(training) %in% naColNames)]
testingClean <- testing[,!(names(training) %in% naColNames)]

set.seed(3217)
trainingSet <- data.frame(trainingClean[,8:60])
testingSet <- data.frame(testingClean[,8:59])
```

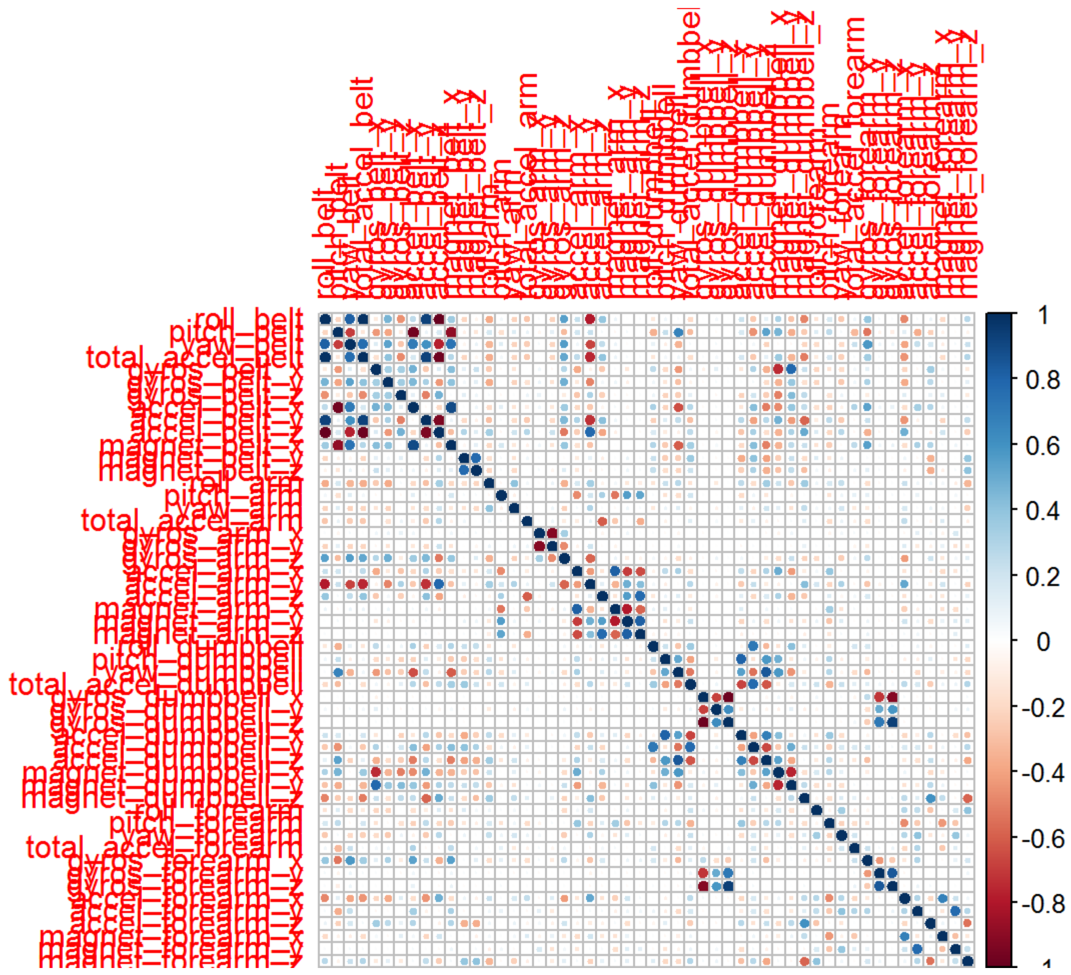
Data Exploration

In this section, I will look at the cleaned training set and examine number of *Classe* attributes as well as looking at the correlation between attrbiutes.

```
#plot a bar chart to examine how many instances of each classe we have
plot(trainingSet$classe, main = "", xlab = "Classe")
```



```
#corrleation plot for all attrbiutes
library(corrplot)
corrplot(cor(trainingSet[,1:52], use = "pairwise"))
```



Machine Learning

In this section, we create two machine learning models. In creating these models we perform cross validation for k-fold (K=10).

We will comapre these models and pick the best. Using the best model we will predict the *Classe* for the records in the testing set.

```
#first we will create a decision tree model. in order to find the best decision tree
we use 10-fold crross-validation.
library(caret)
```

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

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

```
modelRP <- train(classe ~ ., data = trainingSet, method = "rpart",
                 trControl = trainControl(method = "cv", number = 10))
```

```
## Loading required package: rpart
```

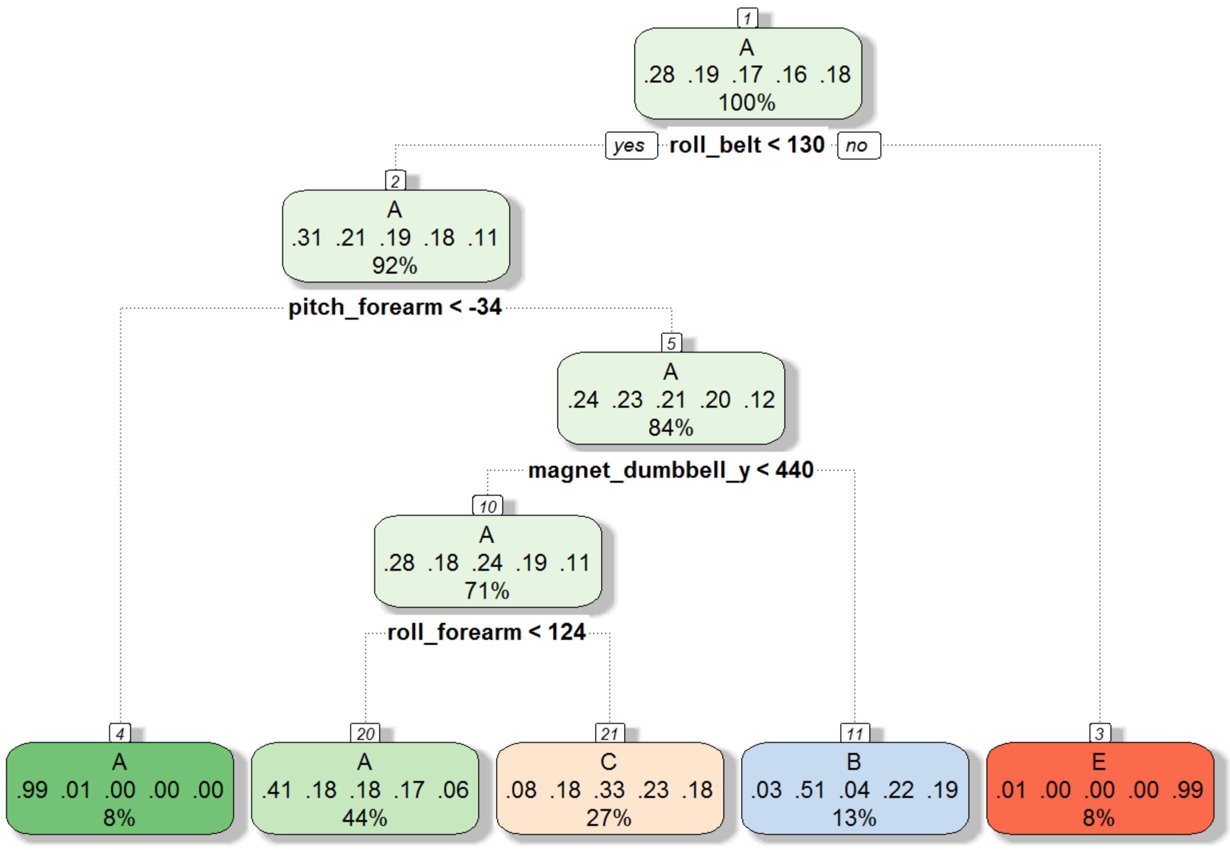
```
#show the output of the model
modelRP
```

```
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17660, 17661, 17659, 17661, 17659, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03567868  0.5034683  0.35143129
##   0.05998671  0.4288120  0.22969504
##   0.11515454  0.3155091  0.04742066
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03567868.
```

```
#plot the decsion tree
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(modelRP$finalModel, sub = "")
```



```
#examine the performance of the decision tree
confusionMatrix(trainingSet$classe, predict(modelRP$finalModel, trainingSet, type = "
class"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5080   81  405    0   14
##          B 1581 1286  930    0    0
##          C 1587  108 1727    0    0
##          D 1449  568 1199    0    0
##          E  524  486  966    0 1631
##
## Overall Statistics
##
##                Accuracy : 0.4956
##                  95% CI : (0.4885, 0.5026)
##     No Information Rate : 0.5209
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3407
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4970  0.50850  0.33040       NA  0.99149
## Specificity            0.9468  0.85310  0.88225   0.8361  0.89008
## Pos Pred Value         0.9104  0.33869  0.50468       NA  0.45218
## Neg Pred Value         0.6339  0.92145  0.78395       NA  0.99913
## Prevalence             0.5209  0.12889  0.26638   0.0000  0.08383
## Detection Rate         0.2589  0.06554  0.08801   0.0000  0.08312
## Detection Prevalence   0.2844  0.19351  0.17440   0.1639  0.18382
## Balanced Accuracy      0.7219  0.68080  0.60633       NA  0.94079
```

```
#create a random forest modee. as before, we use 10-fold cross validation to find the
best random forect model.
modelRF <- train(classe ~ ., data = trainingSet, method = "rf",
                 trControl = trainControl(method = "cv", number = 10))
```

```
## 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
```

```
#show the final model
modelRF
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17660, 17659, 17660, 17660, 17660, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9951588  0.9938760
##   27    0.9951081  0.9938118
##   52    0.9903684  0.9878153
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
#examine the performance of the random forest.
confusionMatrix(trainingSet$classe, predict(modelRF$finalModel, trainingSet, type = "
class"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

As you can see from the resutls above, the random forest does a much better job of predciting the outcomes. Therfore, we will use the random forect model in our next section to predict the *Classe* for records in the testing set.

Prediction

As mentiones above, we will use the random forest model to predict the outcomes.

```
#predict classe for records in the testing set
predict(modelRF$finalModel, newdata = testingSet)
```

```
##  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
```

Conclusion

In this project, I built to machine learning models. In doing so, I used cross-validation to find the best model in each case. I then, comapred the performance of these two models and picked the best (i.e., the random forest model). Finally, I used the random forest model to predict the *Classe* attribute of the testing set.