Practical Machine Learning Project

Nicolas

Get and Clean Data

In this project, we are using data from the accelerometers on the belt, forearm, arm, and dumbell of six participants to predict how well they did the exercise.

```
library(knitr)
library(caret)
```

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

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

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

```
library(corrplot)
```

Download the Data

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.cs
v"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v"
trainCSV <- "./pml-training.csv"
testCSV  <- "./pml-testing.csv"
if (!file.exists(trainCSV)) {
  download.file(trainUrl, destfile=trainCSV, method="curl")
}
if (!file.exists(testCSV)) {
  download.file(testUrl, destfile=testCSV, method="curl")
}
```

Read the Data

```
trainRaw <- read.csv("./pml-training.csv")
testRaw <- read.csv("./pml-testing.csv")
dim(trainRaw)
```

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

```
dim(testRaw)
```

```
## [1]  20 160
```

The training dataset contains 19622 observations and 160 variables, while the testing dataset contains 20 observations and 160 variables. The “classe” variable in the training dataset is the outcome to predict. ### Clean the data Remove observations with missing values and non useful variables.

```
sum(complete.cases(trainRaw))
```

```
## [1] 406
```

Remove columns that contain NA missing values.

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

Remove columns that do not contribute to the accelerometer measurements.

```
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)]
```

Split the training data

Let’s split the clean training dataset into training dataset at 70% and a validation dataset at 30%. The validation dataset will be used to conduct cross validation before the testing dataset.

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

Data Modeling

Fit a predictive model for the activity recognition using **Random Forest** algorithm as it automatically selects important variables and it is robust to correlated covariates and outliers.

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

Estimate the performance of the model on the validation dataset.

```
predictRf <- predict(modelRf, valData)
confusionMatrix(valData$classe, predictRf)
```

```
## 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(predictRf, valData$classe)
accuracy
```

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

```
err <- 1 - as.numeric(confusionMatrix(valData$classe, predictRf)$overall[1])
err
```

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

The estimated accuracy of the model is 99.38% and the estimated out of sample error is 0.61%.

Predicting on Test Dataset

Apply the model to the original testing dataset.

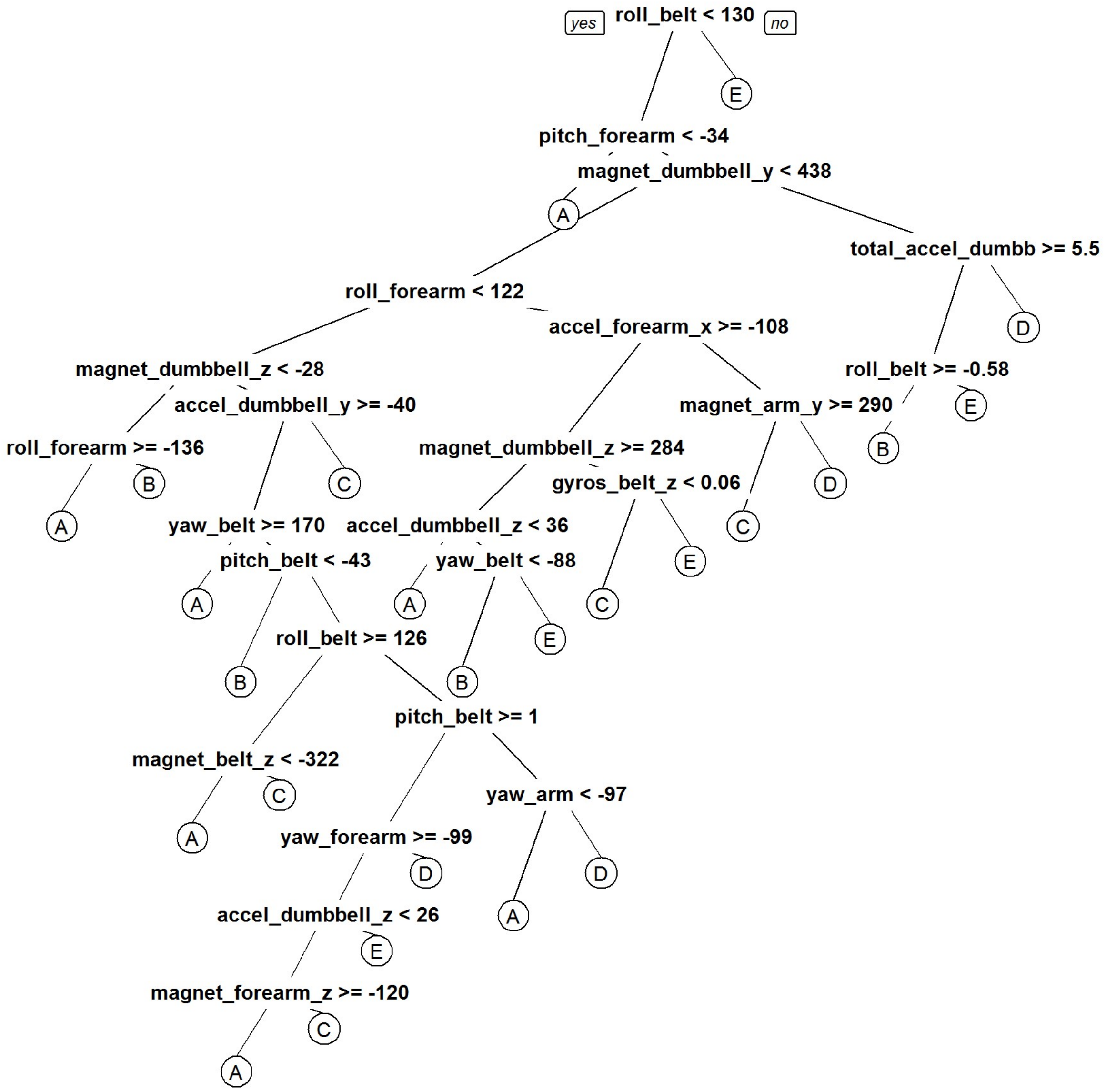
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
##  [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
```

Figures:

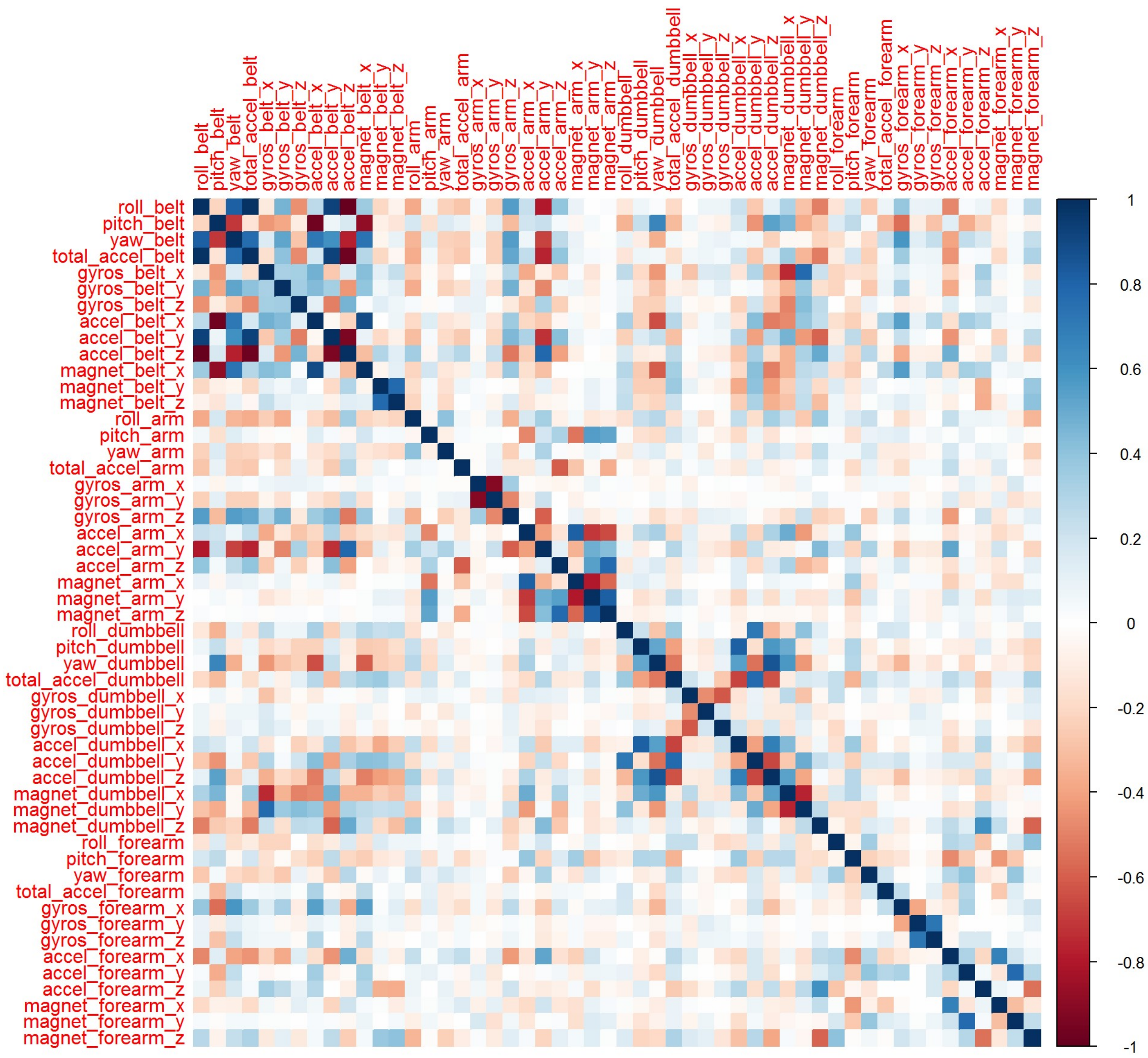
Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel)
```



Matrix Visualization of Correlation

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


roll_belt
pitch_belt
yaw_belt
total_accel_belt
gyros_belt_x
gyros_belt_y
gyros_belt_z
accel_belt_x
accel_belt_y
accel_belt_z
magnet_belt_x
magnet_belt_y
magnet_belt_z
roll_arm
pitch_arm
yaw_arm
total_accel_arm
gyros_arm_x
gyros_arm_y
gyros_arm_z
accel_arm_x
accel_arm_y
accel_arm_z
magnet_arm_x
magnet_arm_y
magnet_arm_z
roll_dumbbell
pitch_dumbbell
yaw_dumbbell
total_accel_dumbbell
gyros_dumbbell_x
gyros_dumbbell_y
gyros_dumbbell_z
accel_dumbbell_x
accel_dumbbell_y
accel_dumbbell_z
magnet_dumbbell_x
magnet_dumbbell_y
magnet_dumbbell_z
roll_forearm
pitch_forearm
yaw_forearm
total_accel_forearm
gyros_forearm_x
gyros_forearm_y
gyros_forearm_z
accel_forearm_x
accel_forearm_y
accel_forearm_z
magnet_forearm_x
magnet_forearm_y
magnet_forearm_z
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1