PML Project

J ten Veen

4-3-2018

Introduction

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, your 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Load packages

```
##Load packages
library(caret)
```

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

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

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018c.
## 1.0/zoneinfo/Europe/Amsterdam'
```

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

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

```
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
```

Data processing

Load data when present in working directory.

```
training_pml <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing_pml <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
```

Show structure of the data.

```
dim(training_pml)
```

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

A data frame with 19622 observations of 160 variables.

Cleaning the data

Remove columns with NA missing values.

```
training_pml <- training_pml[, colSums(is.na(training_pml)) == 0]
testing_pml <- testing_pml[, colSums(is.na(testing_pml)) == 0]
```

Remove identification only variables (columns 1 to 5).

```
identification <- names(training_pml) %in% c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_pa
training_pml <- training_pml[!identification]
```

Data splitting

The test dataset does not contain the classe variable. These classes should be predicted by the model. Therefore we split the training data within 2 groups. Training (train_data) and testing (test_data).

```
set.seed(50)
inTrain <- createDataPartition(training_pml$classe, p=0.70, list=F)
train_data <- training_pml[inTrain, ]
test_data <- training_pml[-inTrain, ]
dim(train_data)
```

```
## [1] 13737    55
```

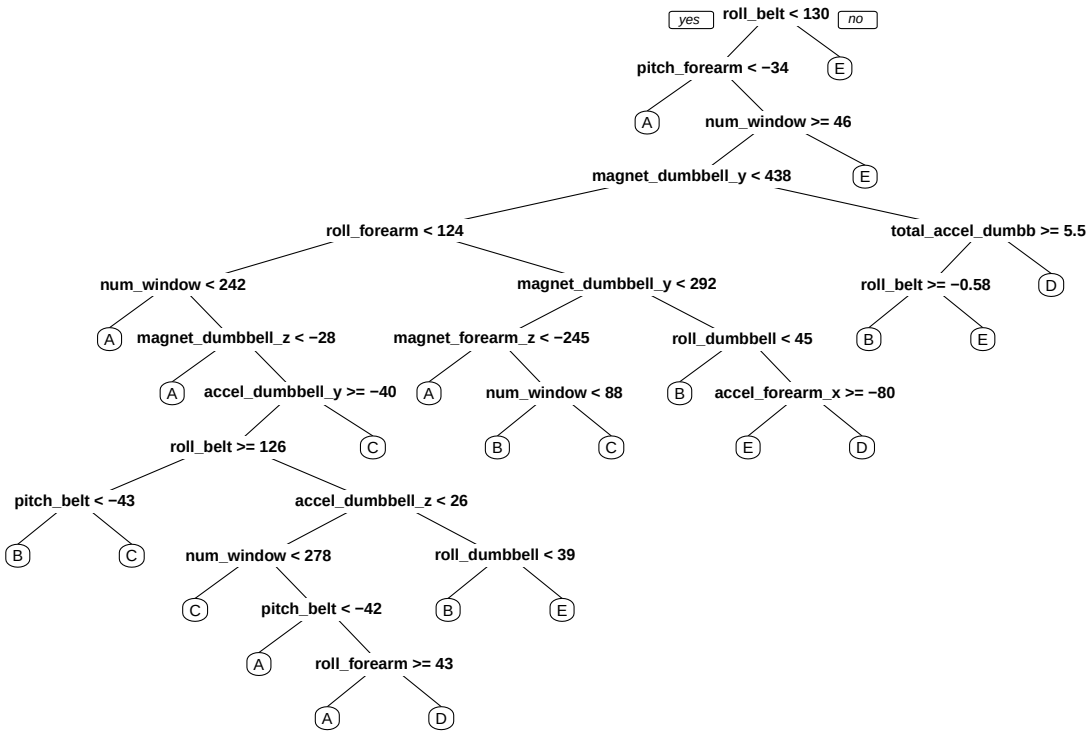
```
dim(test_data)
```

```
## [1] 5885   55
```

Model building

Decision Tree

```
set.seed(50)
modelfit_dt <- rpart(classe ~ ., data=train_data, method="class")
prp(modelfit_dt)
```



Predict on test_data

```
predict_dt <- predict(modelfit_dt, newdata=test_data, type="class", trControl = control)
results_dt <- confusionMatrix(predict_dt, test_data$classe)
results_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1560  266   42   86   37
##          B   46  651   34   33   26
##          C   12   88  831  142  108
##          D   37   60   52  561   45
##          E   19   74   67  142  866
##
## Overall Statistics
##
##                Accuracy : 0.7594
##                  95% CI : (0.7483, 0.7703)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6936
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9319   0.5716   0.8099  0.58195   0.8004
## Specificity            0.8976   0.9707   0.9280  0.96058   0.9371
```

```
## Pos Pred Value         0.7835   0.8241   0.7036  0.74305   0.7414
## Neg Pred Value         0.9707   0.9042   0.9585  0.92144   0.9542
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2651   0.1106   0.1412  0.09533   0.1472
## Detection Prevalence   0.3383   0.1342   0.2007  0.12829   0.1985
## Balanced Accuracy      0.9148   0.7711   0.8690  0.77126   0.8687
```

It is shown that the Accuracy is .7594. The expected out of sample error is 0.2406. Therefore, we'll try another method.

Use random forest

```
modelfit_rf <- randomForest(classe ~., data=train_data)
result_rf <- confusionMatrix(test_data$classe, predict(modelfit_rf, newdata=test_data))
result_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    5 1133    1    0    0
##          C    0    2 1024    0    0
##          D    0    0    6  958    0
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9954, 0.9983)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9982   0.9932   0.9969   1.0000
## Specificity            1.0000   0.9987   0.9996   0.9988   0.9994
## Pos Pred Value         1.0000   0.9947   0.9981   0.9938   0.9972
## Neg Pred Value         0.9988   0.9996   0.9986   0.9994   1.0000
## Prevalence             0.2853   0.1929   0.1752   0.1633   0.1833
## Detection Rate         0.2845   0.1925   0.1740   0.1628   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9985   0.9985   0.9964   0.9978   0.9997
```

It is shown that the Accuracy is .9963, which is better than the decision tree model. Therefore, the expected out of sample error rate is 0.0037.

Conclusion

Based on the two models. We can conclude that, in this case, the Random Forest Classification works better than the Decision Tree Method. The results obtained by the Random Forest Classification were highly accurate on the used dataset. A disadvantage of this technique is the loading time. However, we used the RF model to predict the 20 cases.