Practical Machine Learning Project

Vladimir Martinov

6/1/2019

Practical Machine Learning Project

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, 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://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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Setup

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

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

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## randomForest 4.6-14
```

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

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

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

Load the Training and Test data sets

```
training <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!", ""))
testingSet <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!", ""))

## exploratory data analysis
dim(training)
```

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

```
dim(testingSet)
```

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

```
## remove unnecesary columns
training <- training[, -c(1:7)]
testingSet <- testingSet[, -c(1:7)]

## remove columns with empty values
training<-training[,colSums(is.na(training)) == 0]
testingSet <-testingSet[,colSums(is.na(testingSet)) == 0]
```

Data Partition

```
## split the training set 75-25 into a new training set and a validation set

trainPartition <- createDataPartition(y = training$classe, p=0.75, list=FALSE)
trainingSet <- training[trainPartition, ]
validationSet <- training[-trainPartition, ]

## plot the variable "classe" from the training set
plot(trainingSet$classe, col = "maroon", main = "Classe variable in the training set", xlab = "Classe va
```

Random Forest Model

```
## set a seed for reproducibility
set.seed(86667)

forestModel <- randomForest(classe ~. , data = trainingSet, method="class")

forestPrediction <- predict(forestModel, validationSet, type = "class")
```

Test Results and Confusion Matrix

```
confusionMatrix(forestPrediction, validationSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    1  947    5    0    0
##          C    0    1  845    5    0
##          D    0    0    5  799    3
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9957
```

```
##                  95% CI : (0.9935, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9979   0.9883   0.9938   0.9967
## Specificity            0.9997   0.9985   0.9985   0.9980   1.0000
## Pos Pred Value         0.9993   0.9937   0.9929   0.9901   1.0000
## Neg Pred Value         0.9997   0.9995   0.9975   0.9988   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1931   0.1723   0.1629   0.1831
## Detection Prevalence   0.2845   0.1943   0.1735   0.1646   0.1831
## Balanced Accuracy      0.9995   0.9982   0.9934   0.9959   0.9983
```

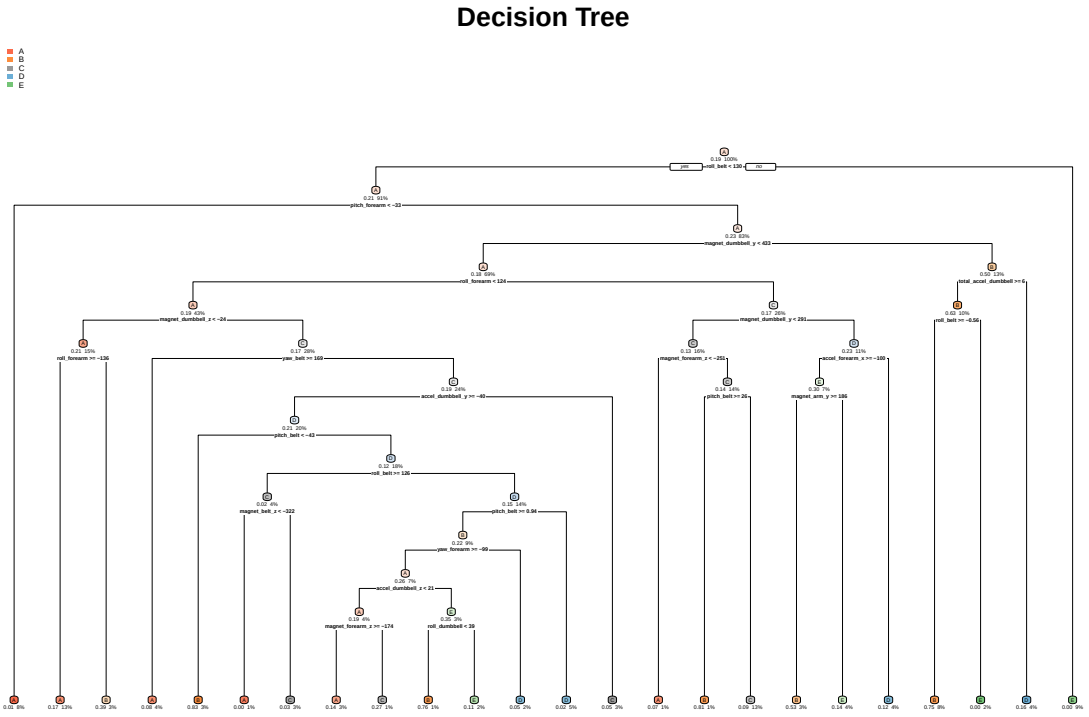
Decision Tree Model

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

treePrediction <- predict(treeModel, validationSet, type = "class")

rpart.plot(treeModel, main = "Decision Tree", extra = 106, under=TRUE, faclen=0)
```

```
## Warning: extra=106 but the response has 5 levels (only the 2nd level is
## displayed)
```



Test Results and Confusion Matrix

```
confusionMatrix(treePrediction, validationSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1205  142   20   49   12
##          B   45  607   84   75   80
##          C   36   93  674  115   93
##          D   54   69   51  512   45
##          E   55   38   26   53  671
##
## Overall Statistics
##
##                Accuracy : 0.7482
##                  95% CI : (0.7358, 0.7603)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6812
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8638   0.6396   0.7883   0.6368   0.7447
## Specificity            0.9364   0.9282   0.9168   0.9466   0.9570
## Pos Pred Value         0.8438   0.6813   0.6667   0.7004   0.7960
## Neg Pred Value         0.9453   0.9148   0.9535   0.9300   0.9434
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2457   0.1238   0.1374   0.1044   0.1368
## Detection Prevalence   0.2912   0.1817   0.2062   0.1491   0.1719
## Balanced Accuracy      0.9001   0.7839   0.8525   0.7917   0.8509
```

Final Result

The Random Forest algorithm had a higher accuracy than the Decision Tree Model, and we'll use it on the testing data set.

```
finalPrediction <- predict(forestModel, testingSet, type = "class")
finalPrediction
```

```
##  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
```