Prediction assignment writeup

Carlos Andres Pillajo

11/16/2020

Overview

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, using the 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.

The objective of this report is to demonstrate the process employed to arrive at a prediction algorithm, which aims to classify the manner in which the participants employed certain exercises. The data comes from accelerometers attached on the belt, forearm and dumbells.

Model building

The outcome variable is class, a 5 level of factor variable. In this dataset, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashion:

- Class A - exactly according to the specification
- Class B - throwing the elbows to the front
- Class C- lifting the dumbbell only halfway
- Class D - lowering the dumbbell only halfway
- Class E - throwing the hips to the front.

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

1. Decision tree will be used to create the model.
2. After the model have been developed. Cross-validation will be performed.
3. Two set of data will be created, original training data set (75%) and subtesting data set (25%).

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

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

```
## 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 Dataset

```
set.seed(100)
trainingData <- read.csv("training.csv", na.strings = c("NA", "#DIV/0!", ""))
testingData <- read.csv("testing.csv", na.strings = c("NA", "#DIV/0!", ""))
trainingData <- trainingData[, colSums(is.na(trainingData)) == 0]
testingData <- testingData[, colSums(is.na(testingData)) == 0]
# Delete variables that are not related
trainingData <- trainingData[, -c(1:7)]
testingData <- testingData[, -c(1:7)]
```

Cross Validation

Split the training data in training data set (75%) and testing data set (25%)

```
traningPartitionData <- createDataPartition(trainingData$classe,  p = 0.75, list = F)
trainingDataSet <- trainingData[traningPartitionData, ]
testingDataSet <- trainingData[-traningPartitionData, ]
dim(trainingData)
```

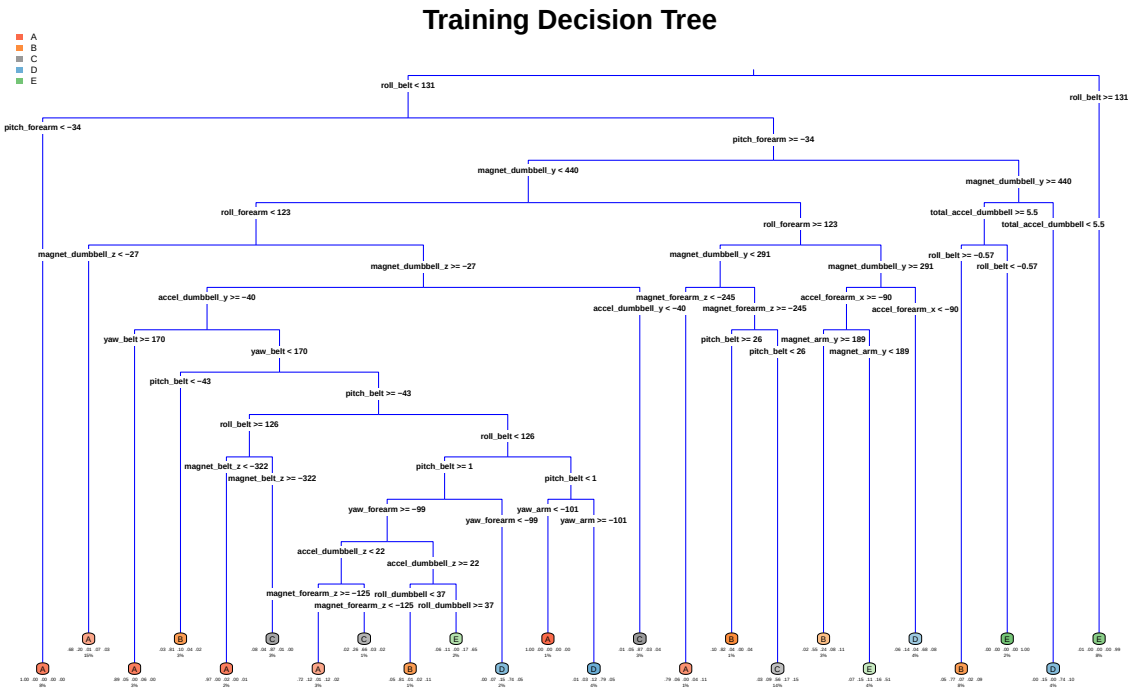
```
## [1] 19622    53
```

```
dim(testingDataSet)
```

```
## [1] 4904   53
```

```
decisionTreeModel <- rpart(classe ~ ., data = trainingDataSet, method = "class")
decisionTreePrediction <- predict(decisionTreeModel, testingDataSet, type = "class")
rpart.plot(decisionTreeModel, main = "Training Decision Tree",cex=0.6, under = TRUE, faclen = 0, compres
```

Prediction model 1 - Decision Tree



```
confusionMatrix(factor(decisionTreePrediction), factor(testingDataSet$classe))
```

Estimate the errors of the prediction algorithm in the Decision Tree model

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1276  196   14   80   30
##          B   31  548   72   32   63
##          C   35   86  682  118  111
##          D   18   68   65  516   51
##          E   35   51   22   58  646
##
## Overall Statistics
##
##                Accuracy : 0.748
##                  95% CI : (0.7356, 0.7601)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6797
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9147   0.5774   0.7977   0.6418   0.7170
## Specificity            0.9088   0.9499   0.9136   0.9507   0.9585
## Pos Pred Value         0.7995   0.7346   0.6609   0.7187   0.7956
## Neg Pred Value         0.9640   0.9036   0.9553   0.9312   0.9377
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2602   0.1117   0.1391   0.1052   0.1317
## Detection Prevalence   0.3254   0.1521   0.2104   0.1464   0.1656
## Balanced Accuracy      0.9118   0.7637   0.8556   0.7963   0.8378
```

```
randomForestModel <- randomForest(factor(classe) ~. , data = trainingDataSet, method = "class")
randomForestPrediction <- predict(randomForestModel, testingDataSet, type = "class")
```

Prediction model 2 - Random Forest

```
confusionMatrix(factor(randomForestPrediction), factor(testingDataSet$classe))
```

Estimate the errors of the prediction algorithm in the Random Forest

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    0    0    0    0
##          B    2  947    3    0    0
```

```
##          C    0    2  852    8    2
##          D    0    0    0  796    1
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9942, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9954
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9979   0.9965   0.9900   0.9967
## Specificity            1.0000   0.9987   0.9970   0.9998   1.0000
## Pos Pred Value         1.0000   0.9947   0.9861   0.9987   1.0000
## Neg Pred Value         0.9994   0.9995   0.9993   0.9981   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1931   0.1737   0.1623   0.1831
## Detection Prevalence   0.2841   0.1941   0.1762   0.1625   0.1831
## Balanced Accuracy      0.9993   0.9983   0.9968   0.9949   0.9983
```

Conclusion

From the results, Random Forest accuracy is higher than Decision tree which is $0.9963 > 0.7412$. Therefore, we will use random forest to answer the assignment.

Final Prediction

```
FinalPrediction <- predict(randomForestModel, testingData, 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
```