Practical Machine Learning Course Project - Prediction Assignment

Mishell Guerra

21 august, 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 built

The outcome variable is classe, 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%).

Load Dataset

```r
set.seed(2888)
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)
```

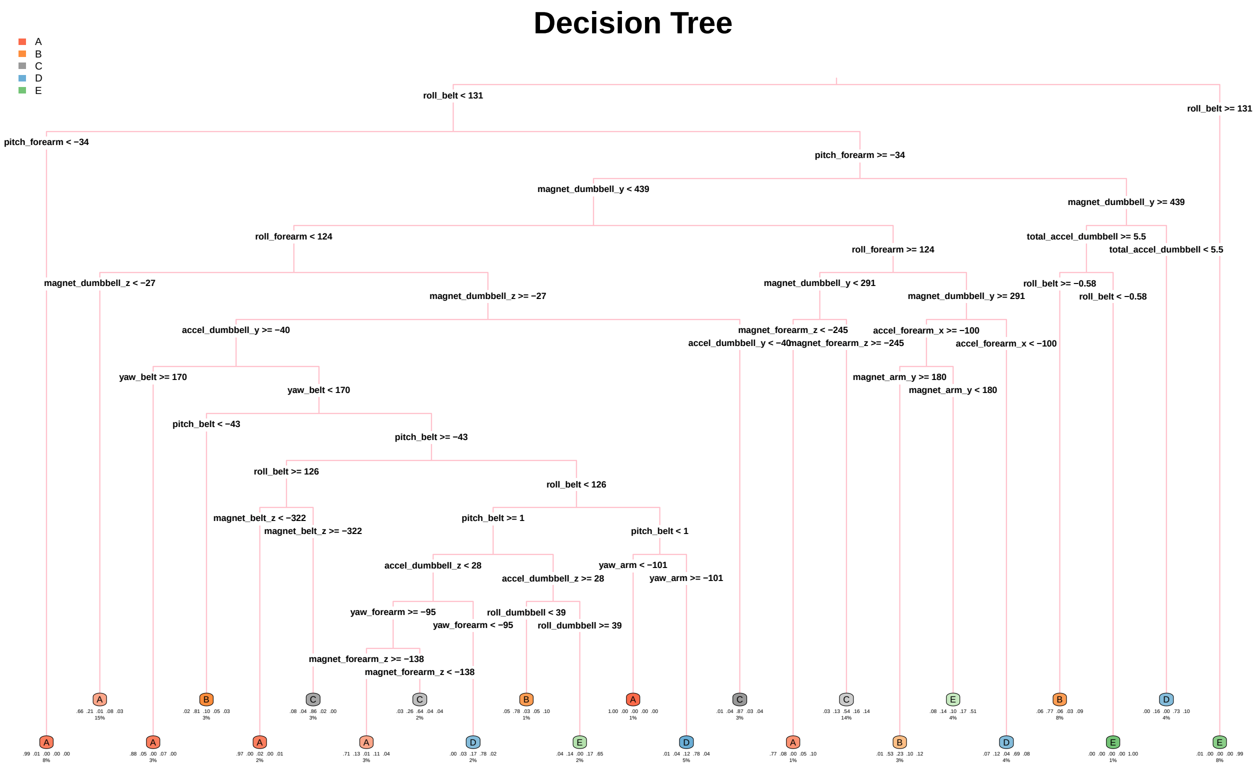
```
## [1] 19622    53
```

```
dim(testingDataSet)
```

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

Prediction model 1 - Decision Tree

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



```
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 1280  178   15   80   35
##          B   30  516   84   25   60
##          C   45  153  685  122  114
##          D   16   61   45  509   47
##          E   24   41   26   68  645
##
## Overall Statistics
##
##                Accuracy : 0.7412
##                  95% CI : (0.7287, 0.7534)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6712
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9176   0.5437   0.8012   0.6331   0.7159
## Specificity            0.9122   0.9497   0.8928   0.9588   0.9603
## Pos Pred Value         0.8060   0.7217   0.6122   0.7507   0.8022
## Neg Pred Value         0.9653   0.8966   0.9551   0.9302   0.9376
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2610   0.1052   0.1397   0.1038   0.1315
## Detection Prevalence   0.3238   0.1458   0.2282   0.1383   0.1639
## Balanced Accuracy      0.9149   0.7467   0.8470   0.7959   0.8381
```

Prediction model 2 - Random Forest

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

```r
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    3    0    0    0
##          B    2  946    2    0    0
##          C    0    0  848    4    0
##          D    0    0    5  800    2
##          E    0    0    0    0  899
##
## 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   0.9968   0.9918   0.9950   0.9978
## Specificity            0.9991   0.9990   0.9990   0.9983   1.0000
## Pos Pred Value         0.9979   0.9958   0.9953   0.9913   1.0000
## Neg Pred Value         0.9994   0.9992   0.9983   0.9990   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2841   0.1929   0.1729   0.1631   0.1833
## Detection Prevalence   0.2847   0.1937   0.1737   0.1646   0.1833
## Balanced Accuracy      0.9989   0.9979   0.9954   0.9967   0.9989
```

Conclusion

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

```
FinalPrediction <- predict(randomForestModel, testingDataSet, type = "class")
summary(FinalPrediction)
```

```
##    A    B    C    D    E
## 1395  950  853  807  899
```

```
head(FinalPrediction, n=20)
```

```
##  4  6  9 10 19 24 29 30 32 42 46 47 49 51 53 58 61 63 76 82
##  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A  A
## Levels: A B C D E
```