PML-Assignment

DrM

October 13, 2018

Study of Characteristics of Correct and Incorrect Ways to Perform Dumbbell Biceps Curls

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Load in appropriate libraries, read the data, label blanks and “#DIV/0!” characters as NA, and remove columns with NA values.

```
#Load libraries
library(lattice)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
set.seed(1234)

#Read in 2 data files:
training <- read.csv("pml-training.csv", na.strings=c("NA","","#DIV/0!"))
test_cases <- read.csv("pml-testing.csv", na.strings=c("NA","","#DIV/0!"))

# Remove columns that contain NA data
training <- training[,colSums(is.na(training))==0]
test_cases<- test_cases[,colSums(is.na(test_cases))==0]
```

Next, remove the first 7 columns (that do not contain data). Then segment the large training file into a training set (training_set) and a test set (test_set) as a 70/30 split.

```
#Remove first 7 columns that do not contain activity data
training <- training[,-c(1:7)]
test_cases <- test_cases[,-c(1:7)]

# Segment training set
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
training_set <- training[inTrain,]
test_set <- training[-inTrain,]
```

Use 3 different models on the training_set data: Recursive Partitioning (rpart), Random Forests (RF), and Generalized Boosted Model (gbm).

```
#Apply models
tr <- trainControl(method = "repeatedcv",number = 20)
rpart_mod <- train(classe~., data=training_set, method="rpart")
rf_mod <- train(classe~., data=training_set, method="rf", ntree=10)
gbm_mod <- train(classe~., data=training_set, method="gbm", verbose=FALSE, trControl=tr)
```

Now apply each model to the test_set data, and calculate a confusion matrix. Print the accuracies for each model.

```
rpart_pred <- predict(rpart_mod,newdata=test_set)
rpart_cm <- confusionMatrix(rpart_pred,test_set$classe)
rpart_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  486  493  452  168
##          B   35  379   31  164  145
##          C  105  274  502  348  302
##          D    0    0    0    0    0
##          E    4    0    0    0  467
##
## Overall Statistics
##
##                Accuracy : 0.489
##                  95% CI : (0.4762, 0.5019)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3311
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.3327   0.4893   0.0000  0.43161
## Specificity            0.6203   0.9210   0.7882   1.0000  0.99917
## Pos Pred Value         0.4890   0.5027   0.3279      NaN  0.99151
## Neg Pred Value         0.9478   0.8519   0.8797   0.8362  0.88641
## Prevalence             0.2845   0.1935   0.1743   0.1638  0.18386
## Detection Rate         0.2600   0.0644   0.0853   0.0000  0.07935
## Detection Prevalence   0.5317   0.1281   0.2602   0.0000  0.08003
## Balanced Accuracy      0.7671   0.6269   0.6388   0.5000  0.71539
```

```
rf_pred <- predict(rf_mod,newdata=test_set)
rf_cm <- confusionMatrix(rf_pred, test_set$classe)
rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   14    0    0    0
##          B    2 1120   10    6    2
##          C    1    3 1011   11    4
##          D    0    1    5  947    2
##          E    1    1    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9893
##                  95% CI : (0.9863, 0.9918)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9865
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9833   0.9854   0.9824   0.9926
## Specificity            0.9967   0.9958   0.9961   0.9984   0.9996
## Pos Pred Value         0.9917   0.9825   0.9816   0.9916   0.9981
## Neg Pred Value         0.9990   0.9960   0.9969   0.9966   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1903   0.1718   0.1609   0.1825
## Detection Prevalence   0.2862   0.1937   0.1750   0.1623   0.1828
## Balanced Accuracy      0.9971   0.9896   0.9907   0.9904   0.9961
```

```
gbm_pred <- predict(gbm_mod,newdata=test_set)
gbm_cm <- confusionMatrix(gbm_pred, test_set$classe)
gbm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1650   39    0    0    1
##          B   16 1071   32    7   10
##          C    1   23  977   23   11
##          D    5    5   15  924   11
##          E    2    1    2   10 1049
##
## Overall Statistics
##
##                Accuracy : 0.9636
##                  95% CI : (0.9585, 0.9683)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.954
##  Mcnemar's Test P-Value : 0.0002637
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9403   0.9522   0.9585   0.9695
## Specificity            0.9905   0.9863   0.9881   0.9927   0.9969
## Pos Pred Value         0.9763   0.9428   0.9440   0.9625   0.9859
## Neg Pred Value         0.9943   0.9857   0.9899   0.9919   0.9932
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2804   0.1820   0.1660   0.1570   0.1782
## Detection Prevalence   0.2872   0.1930   0.1759   0.1631   0.1808
## Balanced Accuracy      0.9881   0.9633   0.9702   0.9756   0.9832
```

```
print(c(rpart_cm$overall[1], rf_cm$overall[1], gbm_cm$overall[1]))
```

```
##  Accuracy  Accuracy  Accuracy
## 0.4890399 0.9892948 0.9636364
```

The random forest model has the highest accuracy of these 3.

These are the 20 most important variables in the random forest model:

```
varImp(rf_mod)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## yaw_belt               59.10
## pitch_forearm          56.14
## magnet_dumbbell_y      52.93
## roll_forearm           46.77
## pitch_belt             43.55
## magnet_dumbbell_z      43.13
## accel_forearm_x        18.42
## accel_dumbbell_y       18.24
## magnet_belt_y          16.72
## roll_dumbbell          15.53
## accel_belt_z           15.36
## accel_dumbbell_z       15.14
## magnet_belt_z          14.92
## total_accel_dumbbell   14.73
## magnet_forearm_z       14.09
## gyros_belt_z           12.46
## magnet_dumbbell_x      12.37
## accel_arm_x            11.74
## magnet_belt_x          11.53
```

Use the random forest model on the downloaded test data set, and use it to predict the Validation Set.

```
casesPrediction <- predict(rf_mod, newdata = test_cases)
casesPrediction
```

```
##  [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
```

Acknowledgement

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz5TpxtNLwa (http://groupware.les.inf.puc-rio.br/har#ixzz5TpxtNLwa) http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)