Coursera - Practical Machine Learning

Project description

Use data from accelometers on the belt forearm, arm and dumbell of 6 participants to predict the manner in which they did their exercises. For this purpose, variable “class” is to be used, together with other relevant variables.

Datasource

The training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data : https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

Experiment architecture

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).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)).

Prework

1.) Downloading data

2.) Installing packages such as caret, randomForest, rpart, rpart.plot, RColorBrewer, rattle, e1071

3.) setting seed in order to ensure reproducibility

Data cleaning

Loading packages and setting seed

```
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(e1071)

set.seed(777)
```

Loading data into memory and replacing all missing values with “NA”

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

Using general exploratory data analysis commands to understand nature of data.

```
#number of observations and columns
dim(training.data)
```

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

str(training.data) reveals that many variables are with “NA” values, additionaly first 7 variables are not relevant for our project assignment, thus we can remove them.

deleting columns if all values are “NA”

```
training.data<-training.data[,colSums(is.na(training.data)) == 0]
testing.data <-testing.data[,colSums(is.na(testing.data)) == 0]
```

```
training.data   <-training.data[,-c(1:7)]
testing.data <-testing.data[,-c(1:7)]
```

Split data for cross-validation

In order to verify functionality of the model, training.data are to be split into train.model data and test.model data in ration 80:20

```
subsamples <- createDataPartition(y=training.data$classe, p=0.8, list=FALSE)
train.model <- training.data[subsamples, ]
test.model <- training.data[-subsamples, ]
#number of observations and columns in train.model data
dim(train.model)
```

```
## [1] 15699    53
```

```
#number of observations and columns in test.model data
dim(test.model)
```

```
## [1] 3923   53
```

Look at the variable “classe” in the train.model dataset in order to decide if further adjustment needed.

```
summary(train.model$classe)
```

```
##    A    B    C    D    E
## 4464 3038 2738 2573 2886
```

Each classe A-E is represented by relativelly fair count of observations. Thus no further adjustment to perform.

Using prediction model: Random Forest

```
modelRF <- randomForest(classe ~. , data=train.model, method="class")

# Predicting - cross-validation
predictionRF <- predict(modelRF, test.model, type = "class")

# Test results on test.model data set - out of sample error
confusionMatrix(predictionRF, test.model$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    6    0    0    0
##          B    0  753    3    0    0
##          C    0    0  681    3    0
##          D    0    0    0  640    2
##          E    0    0    0    0  719
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9956   0.9953   0.9972
## Specificity            0.9979   0.9991   0.9991   0.9994   1.0000
## Pos Pred Value         0.9947   0.9960   0.9956   0.9969   1.0000
## Neg Pred Value         1.0000   0.9981   0.9991   0.9991   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1919   0.1736   0.1631   0.1833
## Detection Prevalence   0.2860   0.1927   0.1744   0.1637   0.1833
## Balanced Accuracy      0.9989   0.9956   0.9973   0.9974   0.9986
```

Using prediction model: Decision Tree

```
modelDT <- rpart(classe ~ ., data=train.model, method="class")

# Predicting - cross-validation
predictionDT <- predict(modelDT, test.model, type = "class")

# Test results on our test.model data set - out of sample error
confusionMatrix(predictionDT, test.model$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1034  135   14   29    5
##          B   34  435   74   61   77
##          C   30  102  522   60   52
##          D   11   41   46  430   47
##          E    7   46   28   63  540
##
## Overall Statistics
##
##                Accuracy : 0.7548
##                  95% CI : (0.741, 0.7682)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6888
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9265   0.5731   0.7632   0.6687   0.7490
## Specificity            0.9348   0.9223   0.9247   0.9558   0.9550
## Pos Pred Value         0.8496   0.6388   0.6815   0.7478   0.7895
## Neg Pred Value         0.9697   0.9001   0.9487   0.9364   0.9441
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2636   0.1109   0.1331   0.1096   0.1376
## Detection Prevalence   0.3102   0.1736   0.1953   0.1466   0.1744
## Balanced Accuracy      0.9307   0.7477   0.8439   0.8123   0.8520
```

Selecting prediction model

From above, the random forest model performed better than the decision tree.

Accuracy 99.64% vs. 75.48%

Out of sample error 0.36% vs.24.52%

Hence, we use the random forest model for the final prediction.

Model in practice

```
runModel <- predict(modelRF, testing.data, type="class")
runModel
```

```
##  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
```