courseProject3

Shogo

1/10/2021

#Overview

This analysis corresponds to the Project Assignment for the Practical Machine Learning course of the John Hopkins Data Science Specialization at Coursera. The project uses data from the Weight Lifting Exercises (WLE) Dataset (see http://groupware.les.inf.puc-rio.br/har) According to the WLE website, six participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions, identified as classes A, B, C, D and E. Class A corresponds to a correct execution of the exercise, and the remaining five classes identify common mistakes in this weight lifting exercise. Several sensors were used to collect data about the quality of the exercise execution. The goal of this project is to obtain a prediction algorithm that takes such a set of sensor readings and correctly predicts the corresponding class (A to E).

#Load and pre-process the data

First we will read the training data. We can observe that there are 19622 rows and 160 columns (variables)

```
pml_training_data = read.table("data/pml-training.csv",
            header = TRUE, sep = ",",
            na.strings = c("NA", "#DIV/0!"))

dim(pml_training_data)
```

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

Most of the variables (152 out of 160) correspond to sensor readings for one of the four sensors. Those sensor-reading variable names (columns 8 to 159) include one of the following strings to identify the corresponding sensor: _belt _arm _dumbbell _forearm

#Only include sensor variables

The first 7 columns do not represent sensor readings, so we can remove these columns.

```
sensorColumns = grep(pattern = "_belt|_arm|_dumbbell|_forearm", names(pml_training_data))
length(sensorColumns)
```

```
## [1] 152
```

```
data = pml_training_data[, c(sensorColumns,160)]
dim(data)
```

```
## [1] 19622   153
```

We must remove variables where most values are NA

```
missingData = is.na(data)
omitColumns = which(colSums(missingData) > 19000)
data = data[, -omitColumns]
dim(data)
```

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

This leaves us with 53 variables. Let's check to see there are no missing values

```
table(complete.cases(data))
```

```
##
##  TRUE
## 19622
```

##Data Spliting

We will set the seed to ensure reproducibility, and create a training and test set

```
set.seed(2014)
library(caret)
```

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

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

```
inTrain <- createDataPartition(y=data$classe, p=0.75, list=FALSE)
training <- data[inTrain,]
dim(training)
```

```
## [1] 14718    53
```

```
testing <- data[-inTrain,]
dim(testing)
```

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

#Train the predictor

We will use the randomForest function (in the randomForest package) to fit the predictor to the training set.

```
library(randomForest)
```

```
## 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
```

```
time1 = proc.time()
training$classe=factor(training$classe)
(randForest = randomForest(classe~., data=training, ntree = 500))
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4184    0    0    0    1 0.0002389486
## B   11 2835    2    0    0 0.0045646067
## C    0   12 2553    2    0 0.0054538372
## D    0    0   27 2383    2 0.0120232172
## E    0    0    2    4 2700 0.0022172949
```

```
time2 = proc.time()
(time = time2 - time1)
```

```
##    user  system elapsed
##   37.79    0.54   38.66
```

As the above results show, the resulting predictor has a quite low OOB (out-of-bag) error estimate. The confusion matrix for the training set indicates that the predictor is accurate on that set.

#Apply Model to Test Sample

After training the predictor we use it on the testing subsample we constructed before, to get an estimate of its out of sample error.

```
predictionTesting = predict(randForest, newdata = testing)
```

The error estimate can be obtained with the confusionMatrix function of the caret package:

```
testing$classe=factor(testing$classe)
confusionMatrix(predictionTesting, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  943    5    0    0
##          C    0    1  850   10    0
```

```
##          D    0    0    0  793    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9937   0.9942   0.9863   0.9978
## Specificity            0.9986   0.9987   0.9973   0.9995   0.9998
## Pos Pred Value         0.9964   0.9947   0.9872   0.9975   0.9989
## Neg Pred Value         1.0000   0.9985   0.9988   0.9973   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1923   0.1733   0.1617   0.1833
## Detection Prevalence   0.2855   0.1933   0.1756   0.1621   0.1835
## Balanced Accuracy      0.9993   0.9962   0.9957   0.9929   0.9988
```