Predict activity quality from activity monitors

Sarvagya Ranjan

Oct 13 2023

##Synopsis

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, your goal will be to use 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 goal of this project is to predict the manner in which they did the exercise. This is the `classe` variable in the training set.

Data description

The outcome variable is `classe` , a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 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)
- throwing the hips to the front (Class E)

Initial configuration

The initial configuration consists of loading some required packages and initializing some variables.

```
#Data variables
training.file   <- './data/pml-training.csv'
test.cases.file <- './data/pml-testing.csv'
training.url    <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.cases.url  <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

#Directories
if (!file.exists("data")){
  dir.create("data")
}
if (!file.exists("data/submission")){
  dir.create("data/submission")
}

#R-Packages
IscaretInstalled <- require("caret")
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

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

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

```
if(!IscaretInstalled){
    install.packages("caret")
    library("caret")
    }

IsrandomForestInstalled <- require("randomForest")
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 4.1.3
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
if(!IsrandomForestInstalled){
    install.packages("randomForest")
    library("randomForest")
    }

IsRpartInstalled <- require("rpart")
```

```
## Loading required package: rpart
```

```
if(!IsRpartInstalled){
    install.packages("rpart")
    library("rpart")
    }

IsRpartPlotInstalled <- require("rpart.plot")
```

```
## Loading required package: rpart.plot
```

```
## Warning: package 'rpart.plot' was built under R version 4.1.3
```

```
if(!IsRpartPlotInstalled){
    install.packages("rpart.plot")
    library("rpart.plot")
    }

# Set seed for reproducability
set.seed(9999)
```

Data processing

In this section the data is downloaded and processed. Some basic transformations and cleanup will be performed, so that `NA` values are omitted. Irrelevant columns such as `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`, `new_window`, and `num_window` (columns 1 to 7) will be removed in the subset.

The `pml-training.csv` data is used to devise training and testing sets. The `pml-test.csv` data is used to predict and answer the 20 questions based on the trained model.

```
# Download data
download.file(training.url, training.file)
download.file(test.cases.url,test.cases.file )

# Clean data
training   <-read.csv(training.file, na.strings=c("NA","#DIV/0!", ""))
testing <-read.csv(test.cases.file , na.strings=c("NA", "#DIV/0!", ""))
training<-training[,colSums(is.na(training)) == 0]
testing <-testing[,colSums(is.na(testing)) == 0]

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

Cross-validation

In this section cross-validation will be performed by splitting the training data in training (75%) and testing (25%) data.

```
subSamples <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
subTraining <- training[subSamples, ]
subTesting <- training[-subSamples, ]
```

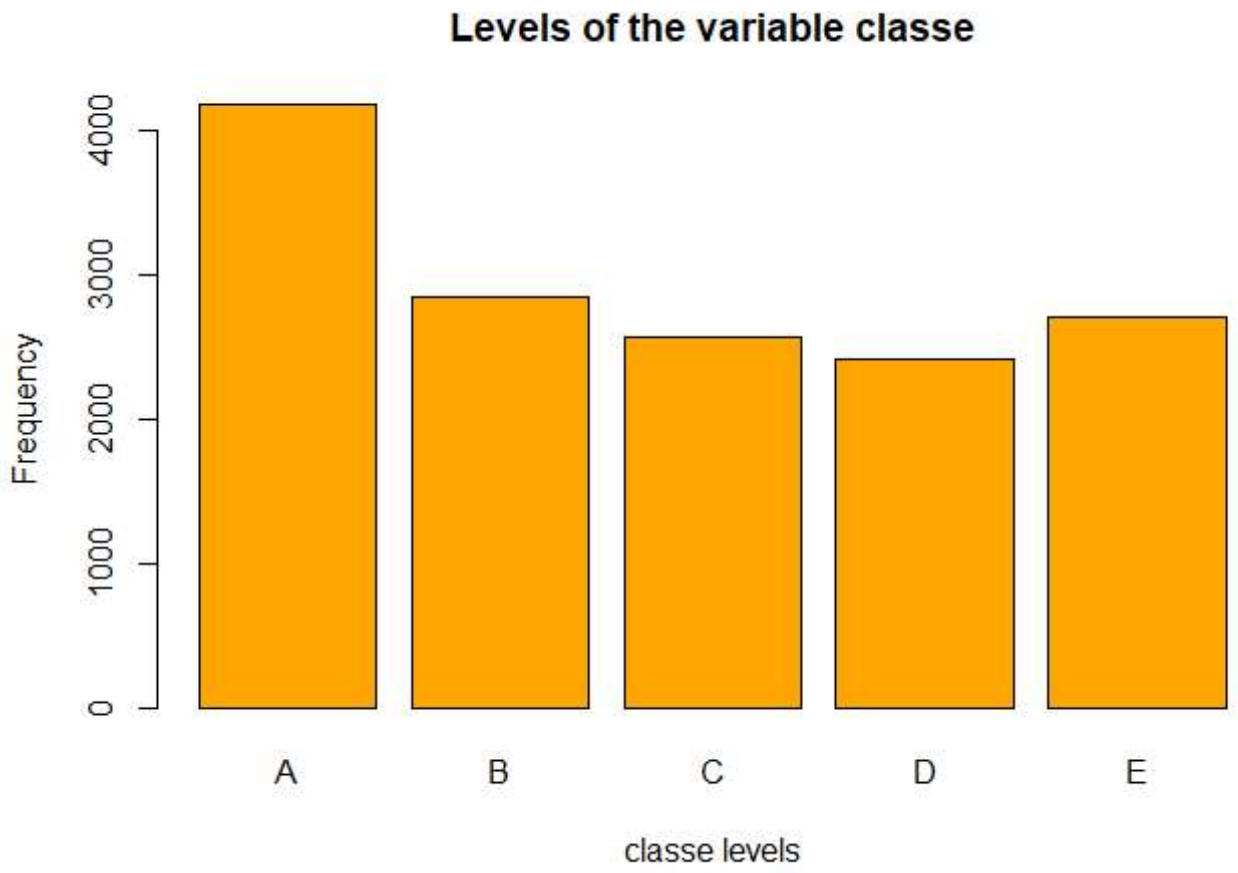
Expected out-of-sample error

The expected out-of-sample error will correspond to the quantity: 1-accuracy in the cross-validation data. Accuracy is the proportion of correct classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of missclassified observations/total observations in the Test data set, which is the quantity: 1-accuracy found from the cross-validation data set.

Exploratory analysis

The variable `classe` contains 5 levels. The plot of the outcome variable shows the frequency of each levels in the subTraining data.

```
#plot(subTraining$classe, col="green", main="Levels of the variable classe", xlab="classe level
s", ylab="Frequency")
barplot(table(subTraining$classe), col = "orange", main = "Levels of the variable classe", xlab
= "classe levels", ylab = "Frequency")
```



The plot above shows that Level A is the most frequent classe. D appears to be the least frequent one.

Prediction models

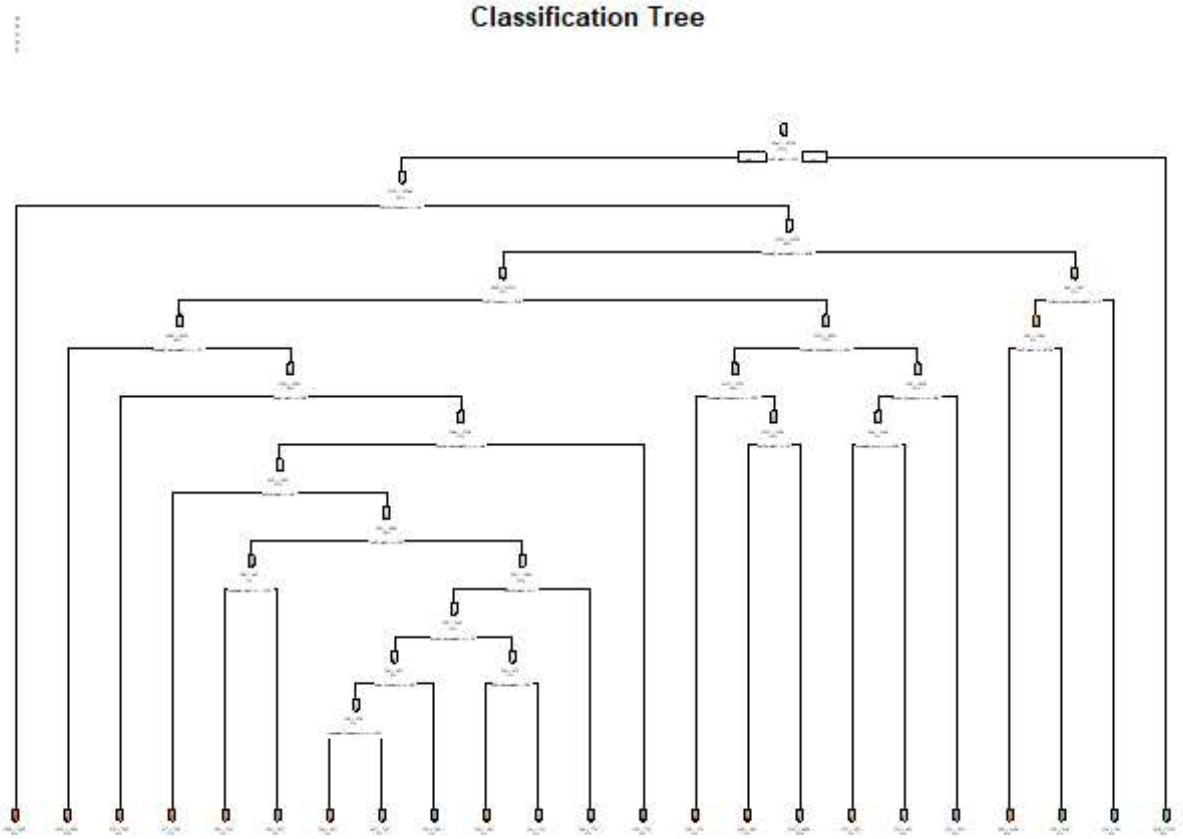
In this section a decision tree and random forest will be applied to the data.

Decision tree

```
# Fit model
modFitDT <- rpart(classe ~ ., data=subTraining, method="class")

# Perform prediction
predictDT <- predict(modFitDT, subTesting, type = "class")

# Plot result
rpart.plot(modFitDT, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```




```
confusionMatrix(predictDT, as.factor(subTesting$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1247  212   23   83   30
##          B   32  530   73   23   73
##          C   35   96  695  112  121
##          D   60   66   46  532   46
##          E   21   45   18   54  631
##
## 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            0.8939   0.5585   0.8129   0.6617   0.7003
## Specificity            0.9008   0.9492   0.9101   0.9468   0.9655
## Pos Pred Value         0.7818   0.7250   0.6563   0.7093   0.8205
## Neg Pred Value         0.9553   0.8996   0.9584   0.9345   0.9347
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2543   0.1081   0.1417   0.1085   0.1287
## Detection Prevalence   0.3252   0.1491   0.2159   0.1529   0.1568
## Balanced Accuracy      0.8974   0.7538   0.8615   0.8043   0.8329
```

Conclusion

Result

The confusion matrices show, that the Random Forest algorithm performens better than decision trees. The accuracy for the Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to 0.739 (95% CI: (0.727, 0.752)) for Decision Tree model. The random Forest model is choosen.

Expected out-of-sample error

The expected out-of-sample error is estimated at 0.005, or 0.5%. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be missclassified.

Submission

In this section the files for the project submission are generated using the random forest algorithm on the testing data.

```
# Perform prediction
#predictSubmission <- predict(modFitRF, testing, type="class")
#predictSubmission

# Write files for submission
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("./data/submission/problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

#pml_write_files(predictSubmission)
```