Barbell Lifts Classification

Lukas Kloucek

2023-04-13

Abstract

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 Setup

The initial setup consists of loading some R packages we will use, initializing some variables and preparing data/folders structure.

```r
#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")
#if(!IscaretInstalled){
#    install.packages("caret")
#    library("caret")
#    }
#IsrandomForestInstalled <- require("randomForest")
#if(!IsrandomForestInstalled){
#    install.packages("randomForest")
#    library("randomForest")
#    }
#IsRpartInstalled <- require("rpart")
#if(!IsRpartInstalled){
#    install.packages("rpart")
#    library("rpart")
#    }
#IsRpartPlotInstalled <- require("rpart.plot")
#if(!IsRpartPlotInstalled){
#    install.packages("rpart.plot")
#    library("rpart.plot")
#    }


### Install packages if necessary
#install.packages("caret")
#install.packages("randomForest")
#install.packages("rpart")
#install.packages("rpart.plot")

### Load packages
library(caret)
library(randomForest)
library(rpart)
library(rpart.plot)

# Set seed for reproducability
set.seed(777)
```

Data Preparation

In this section we will download and prepare the data. 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 and most importantly, `classe` variable will be transformed from character string to factor, which is necessary for model to be able to work with it.

The `pml-training.csv` data is used to create 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!", ""),stringsAsFactors = T)
testing    <-read.csv(test.cases.file , na.strings=c("NA", "#DIV/0!", ""),stringsAsFactors = T)
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

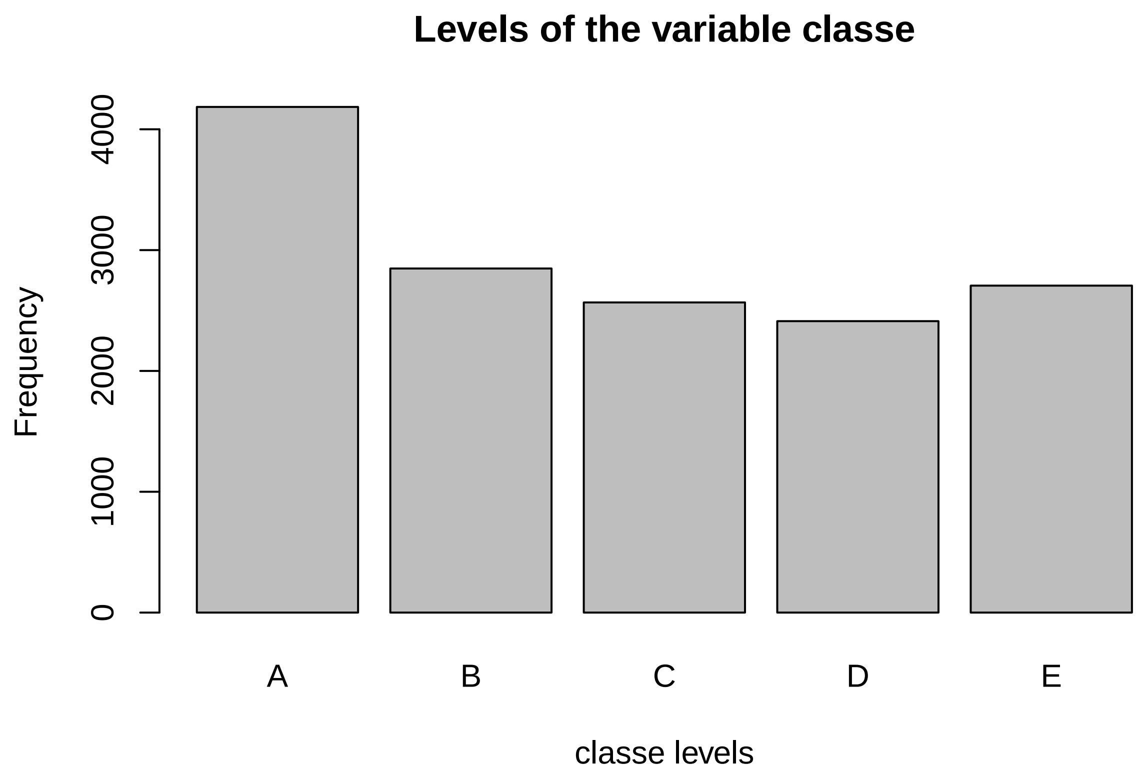
In this section cross-validation will be performed by splitting **the training data** in training (75%) and testing (25%) sub-data sets.

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

Exploratory analysis

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

```
plot(subTraining$classe, col="grey", main="Levels of the variable classe", xlab="classe levels", ylab="
```



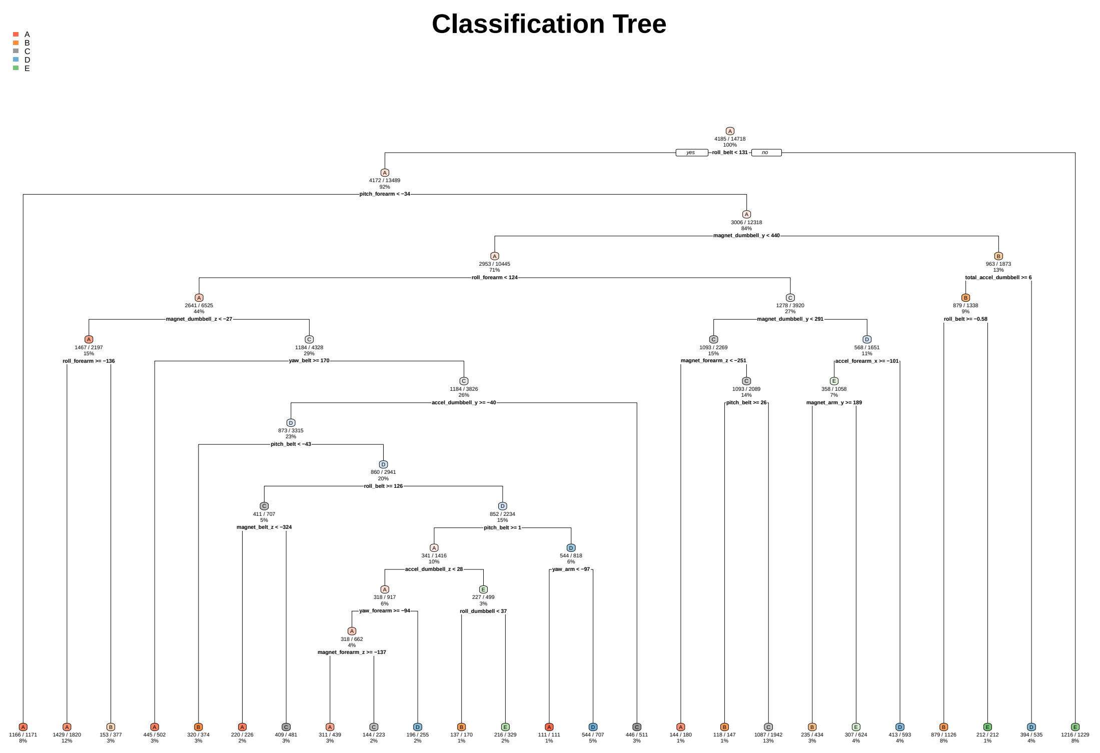
The plot above shows that Level A is the most frequent `classe`. Level D appears to be the least frequent one.Distribution of Levels B to E is quite uniform, no big outliers anywhere.

Prediction models

In this section **Decision Tree** and **Random Forest** will be applied to the data and results summarized.

Decision Tree

```r
# 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)
```



Following confusion matrix shows the errors of the **Decision Tree** prediction algorithm.

```
confusionMatrix(predictDT, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1272  154    6   47   14
##          B   54  570   78   75   78
##          C   30  107  696  122  112
##          D   17   71   51  498   42
##          E   22   47   24   62  655
##
## Overall Statistics
##
##                Accuracy : 0.7527
##                  95% CI : (0.7403, 0.7647)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6864
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9118   0.6006   0.8140   0.6194   0.7270
## Specificity            0.9370   0.9279   0.9084   0.9559   0.9613
## Pos Pred Value         0.8520   0.6667   0.6523   0.7334   0.8086
## Neg Pred Value         0.9639   0.9064   0.9586   0.9276   0.9399
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2594   0.1162   0.1419   0.1015   0.1336
## Detection Prevalence   0.3044   0.1743   0.2176   0.1385   0.1652
## Balanced Accuracy      0.9244   0.7643   0.8612   0.7876   0.8441
```

Random Forest

```
# Fit model
modFitRF <- randomForest(classe ~ ., data=subTraining, method="class")
# Perform prediction
predictRF <- predict(modFitRF, subTesting, type = "class")
```

Following confusion matrix shows the errors of the **Random Forrest** prediction algorithm.

```
confusionMatrix(predictRF, subTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    0  947    5    0    0
##          C    0    1  849    7    0
##          D    0    0    1  797    2
##          E    1    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            0.9993   0.9979   0.9930   0.9913   0.9978
## Specificity            0.9997   0.9987   0.9980   0.9993   0.9998
## Pos Pred Value         0.9993   0.9947   0.9907   0.9963   0.9989
## Neg Pred Value         0.9997   0.9995   0.9985   0.9983   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1931   0.1731   0.1625   0.1833
## Detection Prevalence   0.2845   0.1941   0.1748   0.1631   0.1835
## Balanced Accuracy      0.9995   0.9983   0.9955   0.9953   0.9988
```

Conclusion

Result

The confusion matrix clearly shows that the **Random Forest algorithm performs better** than Decision Tree. **The accuracy of the Random Forest model was 0.995 (95% CI: (0.993, 0.997))** compared to 0.739 (95% CI: (0.727, 0.752)) of Decision Tree model. **The random Forest model will be chosen.**

Expected out-of-sample error

The optimistic out-of-sample error is estimated at 0.005, or 0.5% (half percent). The expected out-of-sample error is calculated as foloows: 1 - accuracy for predictions made against the cross-validation set, but we should expect the error to be closer to the lower 95% CI, which is 0.007, or 0.7%. Our Test data set contains 20 cases and with accuracy above 99% on our cross-validation data, we can expect that very few, if any, of the Test samples will be misclassified.

Submission

In this section the files for the project submission are generated using the Random Forest algorithm on the Test data.

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

```
##  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
```

```r
# Write files for submission
write_submission_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)
  }
}
write_submission_files(predictSubmission)
```