Predict activity quality from activity monitors

Tanya

07/02/2021

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.file <- './data/pml-testing.csv'
training.url    <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.url  <- 'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

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

library("caret")
```

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

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

```
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
```

```
library("rpart")
library("rpart.plot")
```

Data prepration

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.url,test.file )
```

Clean data

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

A: Removing Variables which are having nearly zero variance.

```
non_zero_var <- nearZeroVar(training)

training <- training[,-non_zero_var]
testing<- testing[,-non_zero_var]

dim(training)
```

```
## [1] 19622   124
```

B: Removing Variables which are having NA values. Our threshhold is 95%.

```
na_val_col <- sapply(training, function(x) mean(is.na(x))) > 0.95

training <- training[,na_val_col == FALSE]
testing <- testing[,na_val_col == FALSE]

dim(training)
```

```
## [1] 19622    59
```

C: We also remove col1 to col7 because they are not related to the model

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

Data Partitioning

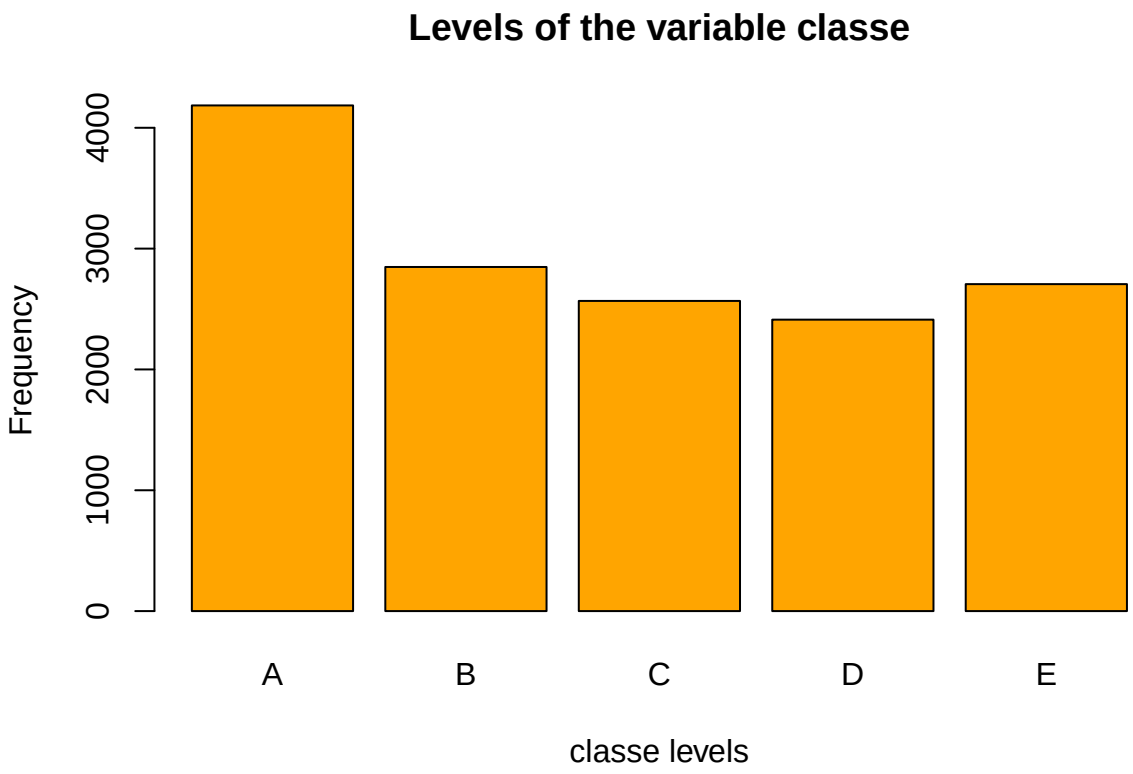
As per recommendation of the course ___ Practical Machine Learning___ , we will be segregating our org_training_data into 2 different parts, one is the training set (consisting 75% of the total data) and validation set (consisting 25% of the total data)

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

Exploratory analysis

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

```
barplot(table(subTraining$classe), col="orange", main="Levels of the variable classe", xlab="classe leve
```



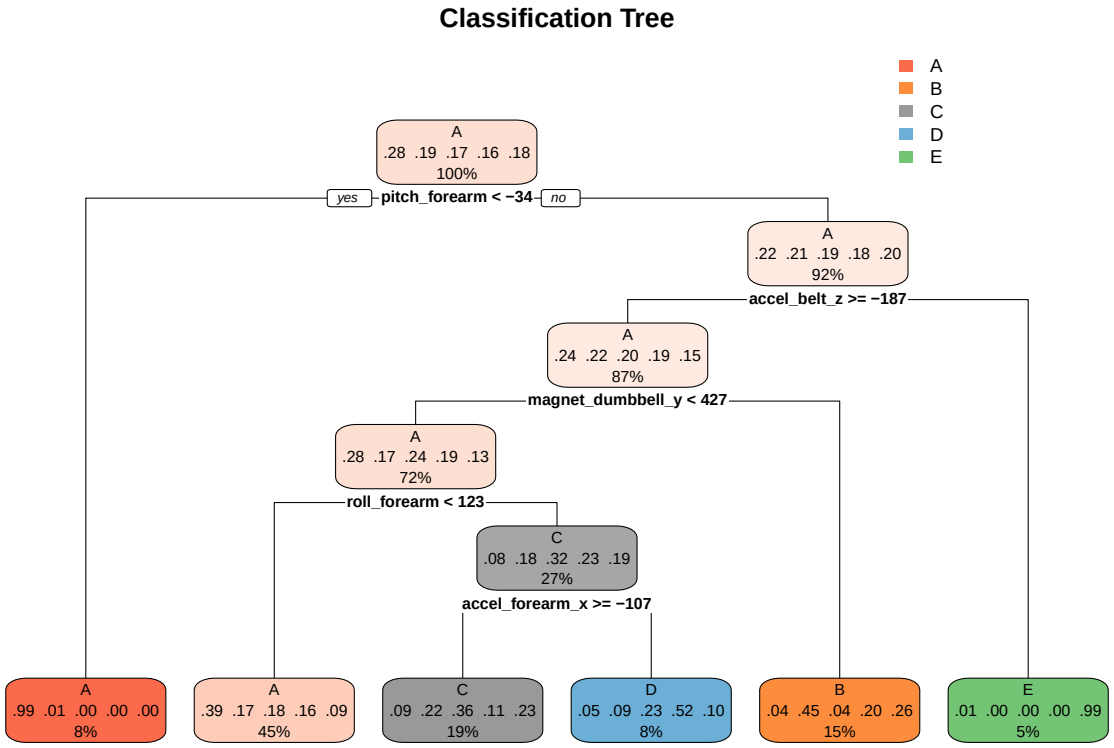
The plot above shows that Level A is the most frequent classe. D appears to be the least frequent one.

Prediction models

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

Decision tree

```
# Fit model
modFitDT <- train(classe ~ ., data = subTraining, method="rpart")
# Perform prediction
predictDT <- predict(modFitDT, subValidating)
# Plot result
rpart.plot(modFitDT$finalModel, main="Classification Tree", roundint=FALSE)
```



Following confusion matrix shows the errors of the prediction algorithm.

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1279  393  367  355  202
##          B   26  338   37  161  200
##          C   65  190  366   97  195
##          D   23   28   85  191   48
##          E    2    0    0    0  256
##
## Overall Statistics
##
##                Accuracy : 0.4955
##                  95% CI : (0.4814, 0.5096)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3403
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9168  0.35616  0.42807  0.23756  0.28413
## Specificity            0.6247  0.89279  0.86490  0.95512  0.99950
## Pos Pred Value         0.4927  0.44357  0.40088  0.50933  0.99225
## Neg Pred Value         0.9497  0.85249  0.87747  0.86465  0.86117
## Prevalence             0.2845  0.19352  0.17435  0.16395  0.18373
## Detection Rate         0.2608  0.06892  0.07463  0.03895  0.05220
## Detection Prevalence   0.5294  0.15538  0.18617  0.07647  0.05261
## Balanced Accuracy      0.7708  0.62448  0.64649  0.59634  0.64181
```

We can see that the prediction accuracy is 50% which is not upto the desired level.

Random forest

```
# Fit model
modFitRF <- randomForest(as.factor(classe) ~ ., data=subTraining, method="class")
# Perform prediction
predictRF <- predict(modFitRF, subValidating, type = "class")
# Fit model
#modFitRF <- train(classe ~ ., data = subTraining, method = "rf", ntree=100)
# Perform prediction
#predictRF <- predict(modFitRF, subValidating)
```

Following confusion matrix shows the errors of the prediction algorithm.

```
confusionMatrix(predictRF, as.factor(subValidating$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    5    0    0    0
##          B    1  941    7    0    0
##          C    0    3  848   10    0
##          D    0    0    0  793    0
##          E    0    0    0    1  901
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9916   0.9918   0.9863   1.0000
## Specificity            0.9986   0.9980   0.9968   1.0000   0.9998
## Pos Pred Value         0.9964   0.9916   0.9849   1.0000   0.9989
## Neg Pred Value         0.9997   0.9980   0.9983   0.9973   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1919   0.1729   0.1617   0.1837
## Detection Prevalence   0.2853   0.1935   0.1756   0.1617   0.1839
## Balanced Accuracy      0.9989   0.9948   0.9943   0.9932   0.9999
```

From the Confusion Matrix, we can clearly see that the prediction accuracy of Random Forest model is 99% which is satisfactory.

Prediction

Now we use it to predict the test set

```
predict(modFitRF, testing)
```

```
##  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
```

Conclusion

As we can we from the result, the random forest algorithem far outperforms the decision tree in terms of accuracy. We are getting 99.25% in sample accuracy, while the decision tree gives us only nearly 50% in sample accuracy