Practical Machine Learning - Course Project

Dr. Robert Bauer

April 10, 2020

Background

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, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.

Each participant was 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. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6NA7beuhc

Objective

The goal of this project is to predict the manner in which they did the exercise, represented by the "classe" variable in the training set. All other variables can be used as prediction variables.

This report describes the set up and cross validation of the modelling, quantifies the expected out of sample error. Finally, the model is being used to predict 20 different test cases.

Getting started

First we need to load the required packages and get the training and test data.

```
rm(list = ls())
library(caret)
## Loading required package: lattice
## Loading required package: ggplot2
```

```
# load data
train_file <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training <- read.csv(train_file)

test_file <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing <- read.csv(test_file)
```

Preparing the data

As a first checkup, we need to verify that both data frames have the same variables.

```
dim(training)
## [1] 19622   160
dim(testing)
## [1]  20 160
```

It appears that this is quite a long list of column names (variables). Let's check it with some R-code:

```
all(names(training) %in% names(testing))
## [1] FALSE
names(training)[which(!(names(training) %in% names(testing)))]
## [1] "classe"
```

classe-variable is the only missing variable in the testing dataframe.
Now let's check on the training data frame:

```
all(names(testing) %in% names(training))
## [1] FALSE
names(testing)[which(!(names(testing) %in% names(training)))]
## [1] "problem_id"
```

problem-id is the only missing variable in the training dataframe

```
testing[['problem_id']]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

While we obviously need to keep the classe-column in the training dataset as our response-variable, we can delete the problem_id-variable from the testing data set, since it's just returning the row lines.

```
testing[['problem_id']] <- c()
```

randomly split the full training data (training) into a smaller training set (Train_Set) and a cross validation set (Val_Set):

```
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
Train_Set <- training[inTrain, ]
Val_Set  <- training[-inTrain, ]
```

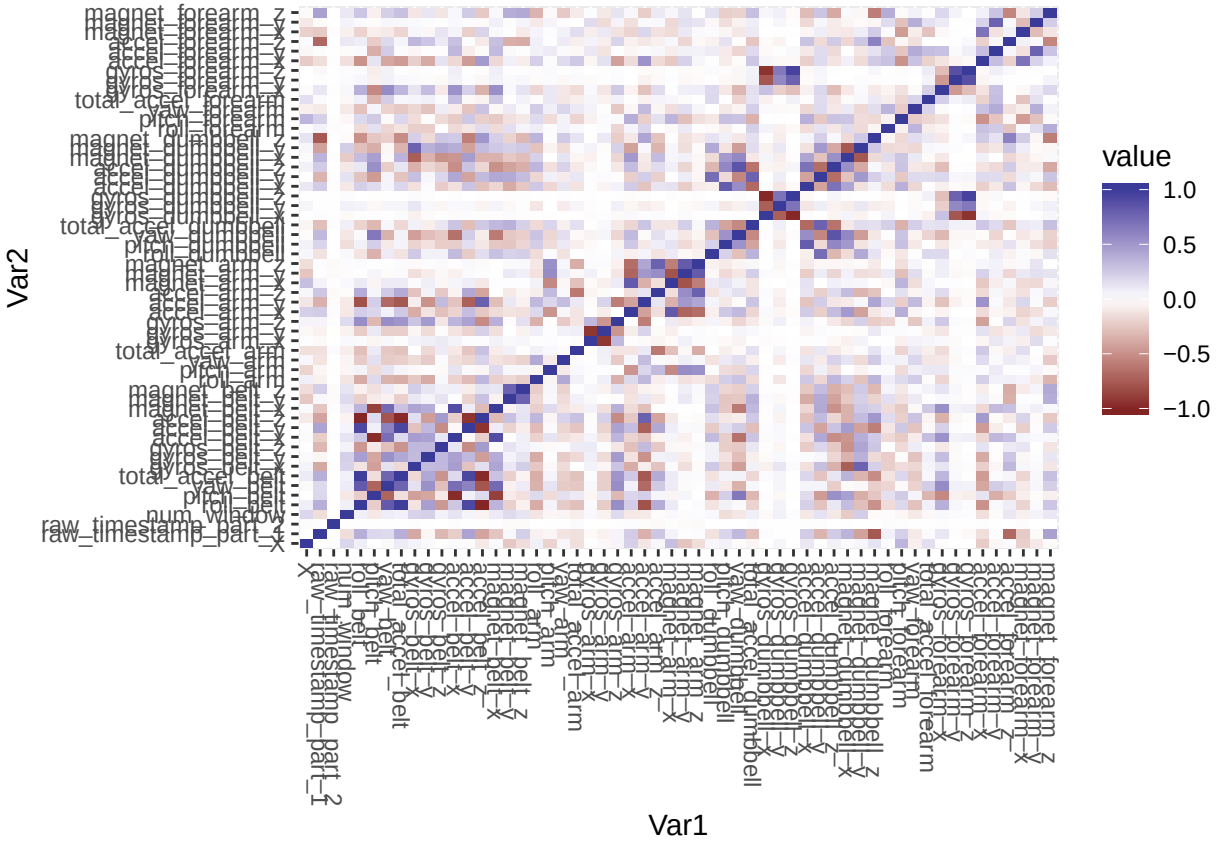
drop variables that are mostly NA or have low variance

```
NA_variables    <- which(sapply(Train_Set, function(x) length(which(is.na(x)))/length(x)) > 0.95)
near0_variables <- nearZeroVar(Train_Set)

dropvars  <- unique(c(NA_variables, near0_variables))
Train_Set <- Train_Set[, -dropvars]
Val_Set  <- Val_Set[, -dropvars]
```

check for colinear variables

```
library(data.table)
cor_matrix <- cor(Train_Set[sapply(Train_Set, is.numeric)])
ctable <- data.table::melt(cor_matrix)
qplot(x=Var1, y=Var2, data=ctable, fill=value, geom="tile") +
   scale_fill_gradient2(limits=c(-1, 1)) +
    theme(axis.text.x = element_text(angle=-90, vjust=0.5, hjust=0))
```



Search through the correlation matrix and return the columns to remove to reduce pair-wise correlations.

```
colinvars <-findCorrelation(cor_matrix, cutoff = .90)
Train_Set <- Train_Set[, -colinvars]
Val_Set  <- Val_Set[, -colinvars]
```

Subsample Training Set to work with (to speed up the modelling)

```
TSet <- Train_Set[sample(1:nrow(Train_Set),size = 1500,replace = F),]
```

This leaves the following models for fitting:

```
names(TSet)[names(TSet) != "classe"]
```

```
##  [1] "X"                    "user_name"            "raw_timestamp_part_1"
##  [4] "raw_timestamp_part_2" "num_window"           "roll_belt"
##  [7] "pitch_belt"           "yaw_belt"             "total_accel_belt"
## [10] "gyros_belt_x"         "accel_belt_y"         "accel_belt_z"
## [13] "magnet_belt_x"        "magnet_belt_y"        "magnet_belt_z"
## [16] "roll_arm"             "pitch_arm"            "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
```

```
## [25] "magnet_arm_x"          "magnet_arm_y"          "magnet_arm_z"
## [28] "roll_dumbbell"         "pitch_dumbbell"        "total_accel_dumbbell"
## [31] "gyros_dumbbell_y"      "gyros_dumbbell_z"      "accel_dumbbell_x"
## [34] "accel_dumbbell_y"      "accel_dumbbell_z"      "magnet_dumbbell_x"
## [37] "magnet_dumbbell_y"     "magnet_dumbbell_z"     "roll_forearm"
## [40] "pitch_forearm"         "yaw_forearm"           "total_accel_forearm"
## [43] "gyros_forearm_x"       "gyros_forearm_y"       "gyros_forearm_z"
## [46] "accel_forearm_x"       "accel_forearm_y"       "accel_forearm_z"
## [49] "magnet_forearm_x"      "magnet_forearm_y"      "magnet_forearm_z"
```

Model building

Set the seed to 2616 and predict "classe" with all the other variables using the following three models:

1. a random forest decision trees ("rf"),
2. decision trees with CART (rpart),
3. gradient boosting trees ("gbm") and
4. linear discriminant analysis ("lda") model.

random forest descision trees

Let's start training and predicting with the random forest decision trees:

```
set.seed(2616)

trControl <- trainControl(method="cv", number=5) # include cross validation as train control method.

modFit_rf <- train(classe ~ .,method="rf", data=TSet, trControl=trControl,verbose=FALSE) #, ntree=100)
cfm_rf <- confusionMatrix(Val_Set$classe, predict(modFit_rf, newdata = Val_Set))
print(cfm_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    4 1022    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9993
##                  95% CI : (0.9983, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9991
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   0.9992   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9961   1.0000   1.0000
## Neg Pred Value         1.0000   0.9992   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1942   0.1737   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1737   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   0.9983   0.9996   1.0000   1.0000
```

decision trees with CART (rpart)

```
set.seed(2616)

trControl <- trainControl(method="cv", number=5) # include cross validation as train control method.

modFit_rpart <- train(classe ~ .,method="rpart", data=TSet, trControl=trControl)
cfm_rpart <- confusionMatrix(Val_Set$classe, predict(modFit_rpart, newdata = Val_Set))
print(cfm_rpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0 1026    0    0    0
##          D    0  964    0    0    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.6619
##                  95% CI : (0.6496, 0.6739)
##     No Information Rate : 0.5317
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5678
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.3640       NA       NA   1.0000
## Specificity            1.0000   1.0000   0.8257   0.8362   1.0000
## Pos Pred Value         1.0000   1.0000       NA       NA   1.0000
## Neg Pred Value         1.0000   0.5807       NA       NA   1.0000
## Prevalence             0.2845   0.5317   0.0000   0.0000   0.1839
## Detection Rate         0.2845   0.1935   0.0000   0.0000   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
```

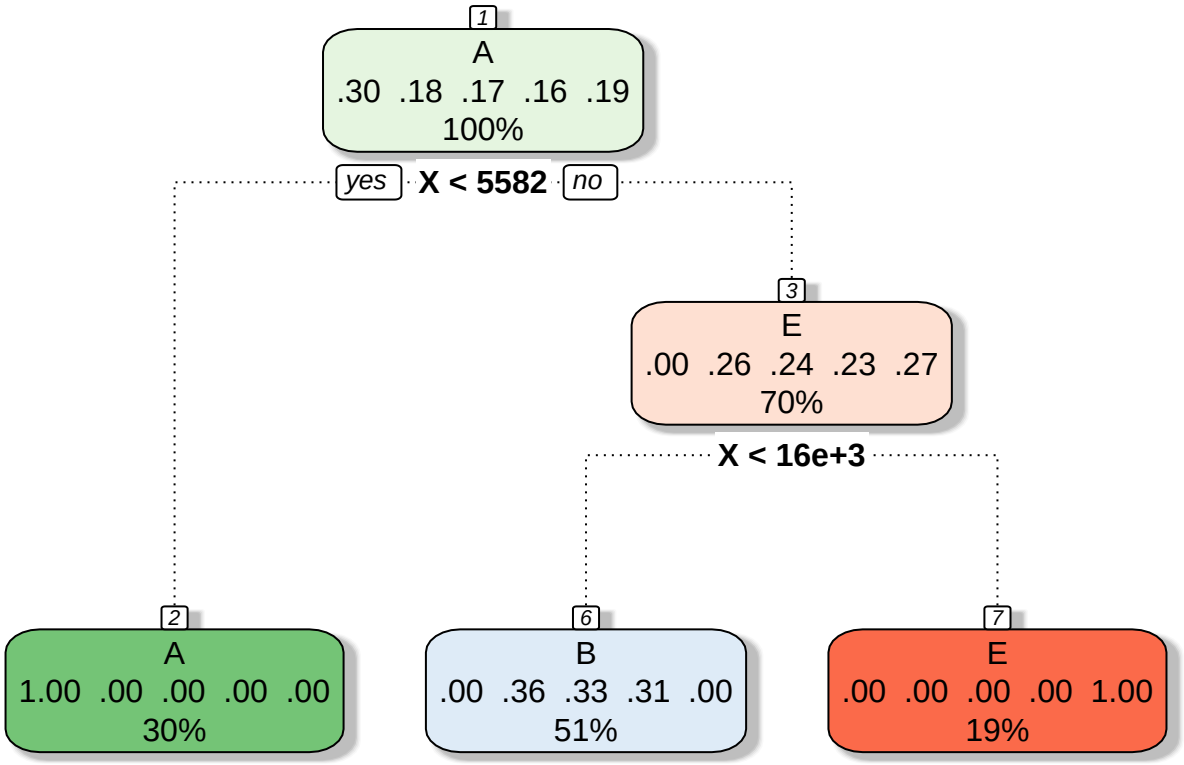
```
## Balanced Accuracy      1.0000   0.6820        NA        NA   1.0000
```

optional:

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(modFit_rpart$finalModel)
```



##

gradient boosting trees ("gbm")

```
set.seed(2616)

trControl <- trainControl(method="cv", number=5) # include cross validation as train control method.

modFit_gbm <- train(classe ~ .,method="gbm", data=TSet, trControl=trControl, verbose=FALSE)
cfm_gbm <- confusionMatrix(Val_Set$classe, predict(modFit_gbm, newdata = Val_Set))
print(cfm_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    0 1139    0    0    0
##          C    0    4 1022    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.998, 0.9997)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9989
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9956   1.0000   1.0000   1.0000
## Specificity            0.9998   1.0000   0.9992   1.0000   1.0000
## Pos Pred Value         0.9994   1.0000   0.9961   1.0000   1.0000
## Neg Pred Value         1.0000   0.9989   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1944   0.1737   0.1638   0.1839
## Detection Rate         0.2843   0.1935   0.1737   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9999   0.9978   0.9996   1.0000   1.0000
```

linear discriminant analysis ("lda")

```
trControl <- trainControl(method="cv", number=5) # include cross validation as train control method.

modFit_lda <- train(classe~., method = 'sparseLDA',data=TSet, trControl=trControl, verbose=F)
cfm_lda <- confusionMatrix(Val_Set$classe, predict(modFit_lda, newdata = Val_Set))
print(cfm_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B 1139    0    0    0    0
##          C 1026    0    0    0    0
##          D  964    0    0    0    0
##          E 1082    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.2845
##                  95% CI : (0.2729, 0.2962)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.2845       NA       NA       NA       NA
## Specificity                NA   0.8065   0.8257   0.8362   0.8161
## Pos Pred Value             NA       NA       NA       NA       NA
## Neg Pred Value             NA       NA       NA       NA       NA
## Prevalence             1.0000   0.0000   0.0000   0.0000   0.0000
## Detection Rate         0.2845   0.0000   0.0000   0.0000   0.0000
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy          NA       NA       NA       NA       NA
```

Get accuracy estimates from the confusion matrices of each model:

```
models <- c("modFit_rf","modFit_rpart", "modFit_gbm","modFit_lda")
n <- length(models)
accuracy <- rep(NA,n)

for(i in 1:n){
  cfm <- get(paste0("cfm_",gsub("modFit_","",models[i])))
  accuracy[i] <- cfm$overall[1]
}

comp <- data.frame(model=gsub("modFit_","",models),accuracy)
print(comp)
```

```
##   model  accuracy
## 1    rf 0.9993203
## 2 rpart 0.6618522
## 3   gbm 0.9991504
## 4   lda 0.2844520
```

Our test reveals that the random forest and gbm models are clearly performing best, with only slight differences between them.

Variable importance

Let's check for the five most important variables in the rf and gbm models and their relative importance values:

```
## random forest
vi <- varImp(modFit_rf)$importance
vi[head(order(unlist(vi), decreasing = TRUE), 5L), , drop = FALSE]
```

```
##                        Overall
## X                    100.000000
## roll_belt              7.297264
## raw_timestamp_part_1   2.370520
## magnet_belt_y          2.169245
## roll_dumbbell          2.088093
```

```
## gbm
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
vi <- varImp(modFit_gbm)$importance
vi[head(order(unlist(vi), decreasing = TRUE), 5L), , drop = FALSE]
```

```
##                       Overall
## X                1.000000e+02
## gyros_dumbbell_y 2.078172e-02
## roll_dumbbell    7.079085e-03
## num_window       9.177816e-04
## yaw_belt         6.019864e-04
```

Both models apparently rely mainly on the X-variable.

Predicting testing data

```
predicted_classe_rf <- predict(modFit_rf, newdata = testing)
cat("model preidction by the random forest model\n")
## model preidction by the random forest model
print(predicted_classe_rf)
##  [1] A A A A A A A A A A A A A A A A A A A A
## Levels: A B C D E

predicted_classe_gbm <- predict(modFit_gbm, newdata = testing)
cat("model preidction by the gbm model\n")
## model preidction by the gbm model
print(predicted_classe_gbm)
##  [1] A A A A A A A A A A A A A A A A A A A A
## Levels: A B C D E
```

Both models also predict the same classe-values on the testing variable.