Practical Machine Learning - Predicting Physical Exercise Executions

MeM

18 september 2016

Introduction

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, we will use data recorded from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

More information is available from the website http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which the participants did the exercise. This is the classe variable of the training set, which classifies the correct and incorrect outcomes into A, B, C, D, and E categories. This report describes how the model for the project was built, its cross validation, expected out of sample error calculation, and the choices made. It was used successfully to accurately predict 20 different test cases.

Exploratory data analysis

The data available from the source mentioned in the introduction is imported, while interpreting the miscellaneous NA, #DIV/0! and empty fields as NA.

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

A quick look at the data and particularly at classe (the variable to bed predicted) results in the following characteristics.

```
str(training, list.len=15)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 132308423
2 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
table(training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
prop.table(table(training$classe))
```

```
##
##         A         B         C         D         E
## 0.2843747 0.1935073 0.1743961 0.1638977 0.1838243
```

A first data clean-up is done by, for example, removing columns 1 to 6 (information and reference columns) and removing columns which consist for a great part of NA's.

```
## Remove columns 1 & 6
training <- training[, 7:160]
testing  <- testing[, 7:160]
## Remove NA columns
is_data  <- apply(!is.na(training), 2, sum) > 19621  # which is the number of observations
training <- training[, is_data]
testing  <- testing[, is_data]
```

The training set is split into two sets to be used in the cross validation. 60% of the training set is used for training purposes and the remaining 40% is used for the performance measurement of the predictive model.

```
library(caret)
set.seed(3141592)
inTrain <- createDataPartition(y=training$classe, p=0.60, list=FALSE)
train1  <- training[inTrain,]
train2  <- training[-inTrain,]
```

train1 is the training data set and contains 11776 observations and train2 is the testing data set and contains 7846 observations. train2 is the blind testset and is only going to be used for accuracy measurements.

```
dim(train1)
```

```
## [1] 11776    54
```

```
dim(train2)
```

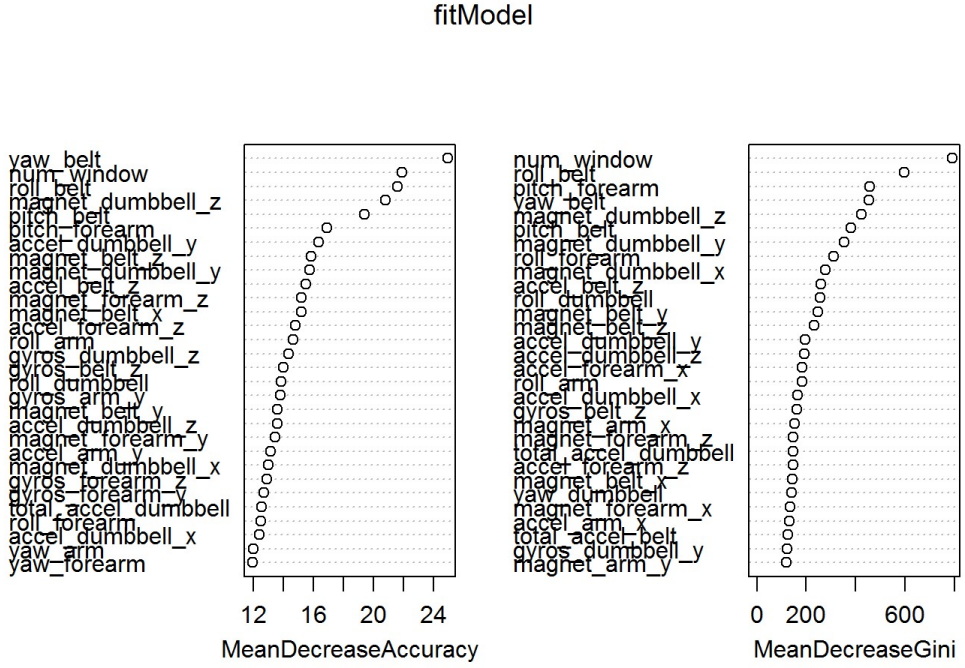
```
## [1] 7846   54
```

Both sets contain 53 clean covariates (columns) to build a model for the prediction of the classe variable (54th column).

Data manipulation

The relative importance of the 53 covariaties is checked using the output of a Random Forest algorithm and then plotted.

```
library(randomForest)
set.seed(3141592)
fitModel <- randomForest(classe~., data=train1, importance=TRUE, ntree=100)
varImpPlot(fitModel)
```



The Accuracy and Gini graphs above are used to select the top 10 variables which we'll be used for the predictive model. The 10 covariates are:

- yaw_belt
- num_window
- roll_belt
- magnet_dumbbell_z
- pitch_belt
- pitch_forearm
- magnet_dumbbell_y
- accel_dumbbell_y
- roll_arm
- roll_forearm.

The correlations between these 10 variables are researched. The output below shows the variables having an absolute value correlation above 75%.

```
correl = cor(train1[,c("yaw_belt","roll_belt","num_window","pitch_belt","magnet_dumbbell_z","magnet_dumbbell_
y","pitch_forearm","accel_dumbbell_y","roll_arm","roll_forearm")])
diag(correl) <- 0
which(abs(correl)>0.75, arr.ind=TRUE)
```

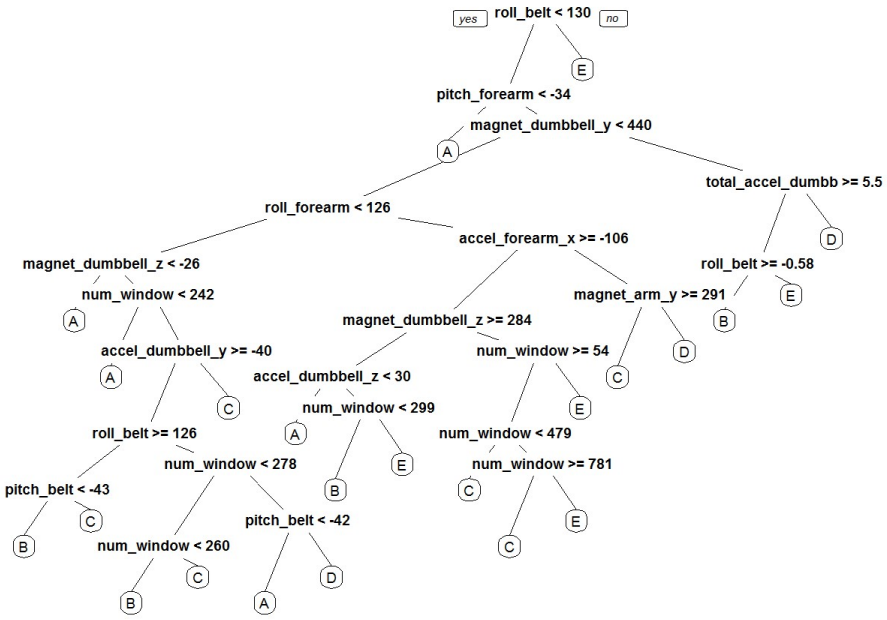
```
##           row col
## roll_belt   2   1
## yaw_belt    1   2
```

```
cor(train1$roll_belt, train1$yaw_belt)
```

```
## [1] 0.8152349
```

Inaccuracy could be introduced by the roll_belt and yaw_belt variables which are highly correlated with each other (above 75%). See output above.

```
library(rpart.plot)
fitModel <- rpart(classe~., data=train1, method="class")
prp(fitModel)
```



A quick tree classifier (see plot above) selects roll_belt as the first discriminant among all 53 covariates which indicates it would be better to eliminate yaw_belt, because roll_belt is a “more important” covariate.

To conclude: the model will be built on the remaining 9 variables. By re-running the correlation script, it can be seen that the maximum correlation among these 9 variables is 50.57% which is sufficiently low to ensure a relatively independent set of covariates.

Data modeling

A Random Forest algorithm is used to build the predictive model. The 9 variables (num_window, roll_belt, magnet_dumbbell_z, pitch_belt, pitch_forearm, magnet_dumbbell_y, accel_dumbbell_y, roll_arm, roll_forearm) determined in the previous step are used as input to predict the classe variable. The data set is large, thus using a small number of folds is justified. A 2-fold cross-validation control will be used in order to gain a relatively low computation time.

```
set.seed(3141592)
fitModel <- train(classe~roll_belt+num_window+pitch_belt+magnet_dumbbell_y+magnet_dumbbell_z+pitch_forearm+accel_
dumbbell_y+roll_arm+roll_forearm,
                  data=train1,
                  method="rf",
                  trControl=trainControl(method="cv",number=2),
                  prox=TRUE,
                  verbose=TRUE,
                  allowParallel=TRUE)
```

The model is saved for later use, which makes it possible for later use by referencing directly to a variable using one command.

```
saveRDS(fitModel, "modelRF.Rds")
fitModel <- readRDS("modelRF.Rds")
```

The accuracy of the model can be obtained by using caret’s confusion matrix function applied on train2 (the test set).

```
predictions <- predict(fitModel, newdata=train2)
confusionMat <- confusionMatrix(predictions, train2$classe)
confusionMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    1    0    0    0
##          B    1 1513    0    0    1
##          C    0    4 1368    6    1
##          D    0    0    0 1280    3
##          E    0    0    0    0 1437
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.9965, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9973
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9967   1.0000   0.9953   0.9965
## Specificity            0.9998   0.9997   0.9983   0.9995   1.0000
## Pos Pred Value         0.9996   0.9987   0.9920   0.9977   1.0000
## Neg Pred Value         0.9998   0.9992   1.0000   0.9991   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1928   0.1744   0.1631   0.1832
## Detection Prevalence   0.2845   0.1931   0.1758   0.1635   0.1832
## Balanced Accuracy      0.9997   0.9982   0.9992   0.9974   0.9983
```

The resulting accuracy of 99.78% is very high and validates the hypothesis made when eliminating most of the original training variables and only use 9 relatively independent variables. Because the train2 testset is blind (not used while building the predictive model) it gives an unbiased estimate of the out-of-sample error rate equal to 100%-99.78%=0.22%.