Predicting Exercise Performance with Accelerometer Data

vkrapp

Sunday, April 19, 2015

Executive summary

In this project, we 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: 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har. The goal of the project was to predict the manner in which they did the exercise. After cleaning the data, the random forest algorithm was applied to build the model.

Needed Packages

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

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

Reading and Cleaning Data

First, we read the data into R and remove all columns that include empty spaces or NAs, since these columns are not useful for fitting models.The data files are already downloaded and saved in the current working directory.

```
train <- read.csv("pml-training.csv", na.strings = c("NA", ""))
test <-  read.csv("pml-testing.csv", na.strings = c("NA", "")) # This data set is for the submission
dim(train) #19622   160 -> large sample
```

```
## [1] 19622   160
```

```
dim(test) #20 160
```

```
## [1]  20 160
```

```
# All variables that include NAs or empty space are removed:
complete <- complete.cases(t(train))
train <- train[,complete]
test <- test[,complete]
```

Next, variables that include timing information or the index of the observation are removed because they do not provide relevant information for our model.

```
train <- train[,c(-1,-3:-7)]
test <- test[,c(-1,-3:-7)]
dim(train) #19622    54
```

```
## [1] 19622    54
```

With this simplified data set we are going to build our model.

Cross Validation and Model Fitting

The data set is splitted into a training (70%) and a test (validation) set (30%). We perform the random forest algorithm only on the training set with a 5-fold cross-validation. To estimate the out-of-sample-error, we test our model‘s accuracy on an independent test set.

```
set.seed(2005)
inTrain <- createDataPartition(y=train$classe, p=0.7, list=FALSE)
training <- train[inTrain,]
testing <- train[-inTrain,]

# Fit model
set.seed(2005)
modelFit <- train(classe~., method="rf", data=training,
                  trControl = trainControl(method = "cv", number = 5))
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
modelFit
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10990, 10990, 10990, 10990, 10988
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9895178  0.9867391  0.004143798  0.005244772
##   29    0.9915559  0.9893183  0.002539450  0.003213169
##   57    0.9826753  0.9780843  0.004368032  0.005523664
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 29.
```

```r
modelFit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 29
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    3    0    0    0 0.0007680492
## B   14 2637    6    1    0 0.0079006772
## C    0    9 2372   15    0 0.0100166945
## D    1    2   27 2219    3 0.0146536412
## E    0    0    5    8 2512 0.0051485149
```

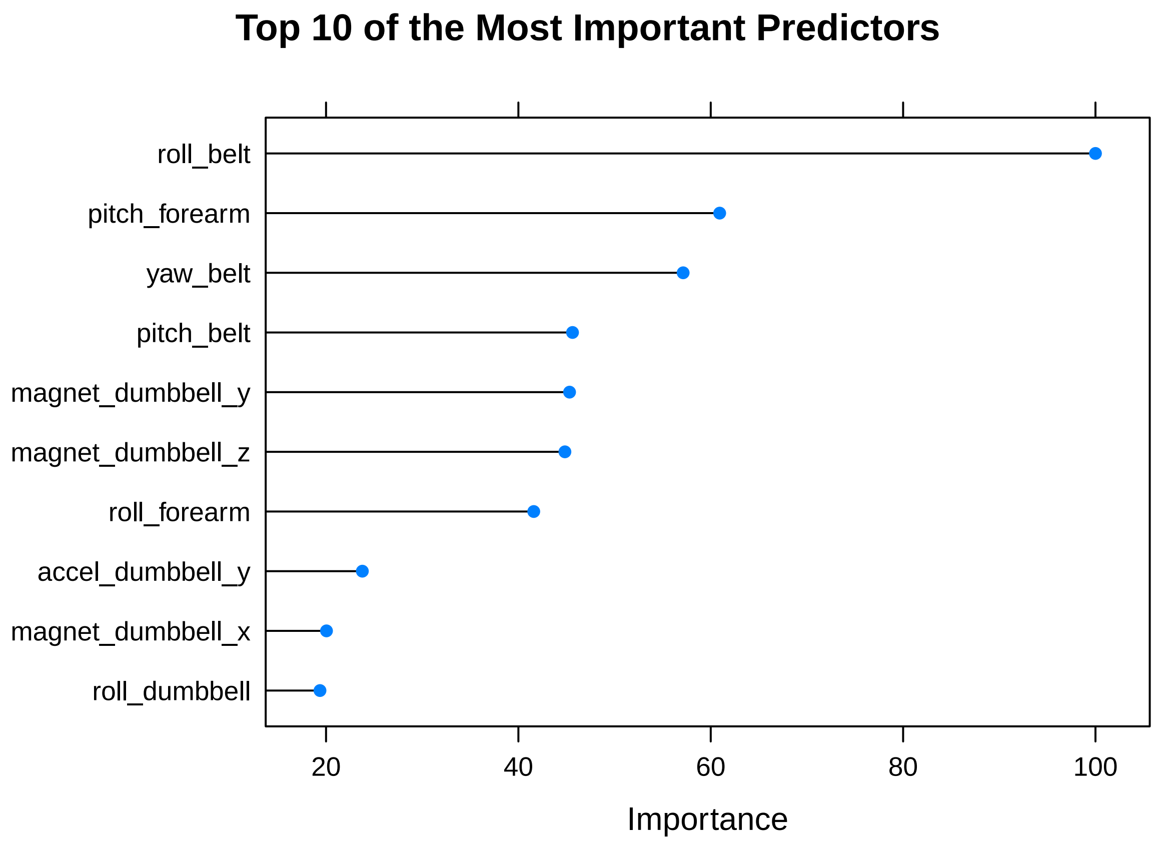
```r
# Test model on testing data
predictions <- predict(modelFit, testing)

# Evaluation of prediction
matrix <- confusionMatrix(predictions,testing$classe)
matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   11    0    0    0
##          B    1 1124    5    0    0
##          C    0    4 1019   11    3
##          D    0    0    2  953    4
##          E    1    0    0    0 1075
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9868   0.9932   0.9886   0.9935
## Specificity            0.9974   0.9987   0.9963   0.9988   0.9998
## Pos Pred Value         0.9935   0.9947   0.9826   0.9937   0.9991
## Neg Pred Value         0.9995   0.9968   0.9986   0.9978   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1910   0.1732   0.1619   0.1827
## Detection Prevalence   0.2860   0.1920   0.1762   0.1630   0.1828
## Balanced Accuracy      0.9981   0.9928   0.9947   0.9937   0.9967
```

So with this approach, we get a accuracy of 0.9928632 on our testing data set. This is a pretty good prediction. In a next step, the ten most important variables for the model are plotted.

```
# Get the variable importance
VarImportance <- varImp(modelFit)
# Plot the 10 most important predictors
plot(VarImportance, main = "Top 10 of the Most Important Predictors", top = 10)
```



Submission

In the end, we use our model to predict the classes of the 20 test cases.

```
answers = predict(modelFit, test)
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")
        write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
        }
    }
pml_write_files(answers)
```