PracticalMachineLearningProject

Dxander

Saturday, January 24, 2015

Executive Summary

The purpose of this project is to analyze the activity of 6 participants to determine how well they perform activities. The data are from accelerometers on the belt, forearm, and dumbell use of the 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Information on other uses of the data for this project can be found at http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

This project involves classification. The goal is to classify the the manner in which the participant did the exercise.
If the exercise was performed exactly to specifications (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). See (VBGUF 2015)

Key points in the process are 1) Classify the manner in which the exercise was done. 2) The classification model should be built using features and cross-validataion. 3) As a performance measure calculate the out of sample error. 4) Finally, use the test data and the model to correctly classify 20 test cases.

A resource for this project is the paper by Velloso, E; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitive Activity Recognition of Weight Lifting Exercises (2013)

Data Setup

Load required packages and set random number generator to ensure reproduceability.

```r
library(Hmisc)
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: survival
## Loading required package: splines
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```r
library(caret)
```

```
## Loading required package: ggplot2
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:survival':
```

```
##
##     cluster
```

```r
library(kernlab)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:Hmisc':
##
##     combine
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.1.2
```

```r
set.seed(9237)
```

set the working directory

```r
setwd("E:/DataScientist/PracticalMachineLearning/Project/PML")
```

```r
downloadDataset <- function(URL="", destFile="data.csv"){
        if(!file.exists(destFile)){
                download.file(URL, destFile, method="curl")
        }else{
                message("You already downloaded the data!")
        }
}

trainURL<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

testURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
downloadDataset(trainURL, "pml-training.csv")
```

```
## You already downloaded the data!
```

```r
downloadDataset(testURL, "pml-testing.csv")
```

```
## You already downloaded the data!
```

Load the data into R

The original data set consist of lots of data marked "NA" and summary data that is not in the testing set. This data is removed from the training set. Same procedures are performed on both training and final testing set.

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

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

```
dim(final_testing)
```

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

```
# summary(training)
```

```
training <- training[ ,colSums(is.na(training)) == 0]
final_testing <- final_testing[ ,colSums(is.na(final_testing)) == 0]

for(i in c(8:ncol(training)-1)) {training[,i] = as.numeric(as.character(training[,i]))}
for(i in c(8:ncol(final_testing)-1)) {final_testing[,i] = as.numeric(as.character(final_testing[,i]))}

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

Examine the feature set and create the model data

```
feature_set <- colnames(training[colSums(is.na(training)) == 0])[-(1:7)]
model_data <- training[feature_set]
# feature_set

inTrain <- createDataPartition(y=model_data$classe, p=0.6, list=FALSE )
training <- model_data[inTrain,]
testing <- model_data[-inTrain,]
dim(training); dim(testing);
```

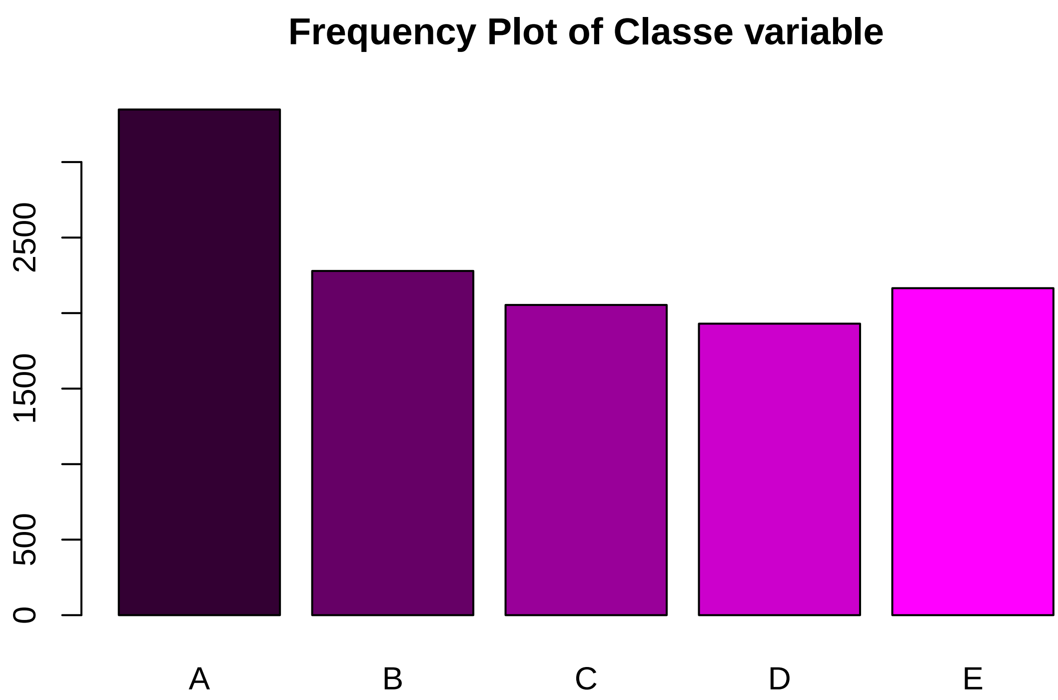
```
## [1] 11776    46
```

```
## [1] 7846   46
```

```
summary(training$classe)
```

```
##    A    B    C    D    E
## 3348 2279 2054 1930 2165
```

```r
plot(training$classe, col=rgb((1:5)/5,0,(1:5)/5), main="Frequency Plot of Classe variable")
```



Correlated Predictors

```r
M <- abs(cor(training[,-length(training)]))
diag(M) <- 0
which(M > 0.8, arr.ind=T)
```

```
##                  row col
## magnet_belt_x      4   1
## accel_belt_z       3   2
## accel_belt_y       2   3
## accel_belt_x       1   4
## gyros_arm_y       12  11
## gyros_arm_x       11  12
## magnet_arm_x      17  14
## accel_arm_x       14  17
## magnet_arm_z      19  18
## magnet_arm_y      18  19
## accel_dumbbell_x  27  21
## accel_dumbbell_z  29  22
## gyros_dumbbell_z  26  24
```

```
## gyros_forearm_z   39  24
## gyros_dumbbell_x  24  26
## gyros_forearm_z   39  26
## pitch_dumbbell    21  27
## yaw_dumbbell      22  29
## gyros_forearm_z   39  38
## gyros_dumbbell_x  24  39
## gyros_dumbbell_z  26  39
## gyros_forearm_y   38  39
```

```
correlMatrix <- cor(training[, -length(training)])
#corrplot(correlMatrix, order = "FPC", method = "circle", type = "lower", tl.cex = 0.8,  tl.col = rgb(0
```

Although there are many correlated predictors because I am not a specialist in sports physics I do not know of a way to combine the variables in a knowledgeable way. So I will continue with the main tasks.

Construct model with 4-fold cross validation

```
model <- train(training$classe ~., data= training, method="rf", prox=TRUE,
               trControl = trainControl(method = "cv", number =4, allowParallel=TRUE))

model
```

```
## Random Forest
##
## 11776 samples
##    45 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 8834, 8831, 8831, 8832
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      0.006        0.008
##   23    1         1      0.003        0.004
##   45    1         1      0.005        0.006
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

To calculate the prediction accuracy of the classification model

1st classify training set

```r
train_pred <- predict(model, training)
confusionMatrix(train_pred, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.194    0.174    0.164    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

For the training set the in sample accuracy is 1 or 100%

2nd classify testing set or out of sample accuracy

```r
test_pred <- predict(model, testing)
confusionMatrix(test_pred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   20    0    0    0
##          B    1 1486   13    0    1
##          C    0   12 1355   32    4
##          D    0    0    0 1253    3
##          E    0    0    0    1 1434
##
## Overall Statistics
##
##                Accuracy : 0.989
##                  95% CI : (0.986, 0.991)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.986
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.979    0.990    0.974    0.994
## Specificity             0.996    0.998    0.993    1.000    1.000
## Pos Pred Value          0.991    0.990    0.966    0.998    0.999
## Neg Pred Value          1.000    0.995    0.998    0.995    0.999
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.189    0.173    0.160    0.183
## Detection Prevalence    0.287    0.191    0.179    0.160    0.183
## Balanced Accuracy       0.998    0.988    0.992    0.987    0.997
```

For the test set the out of sample accuracy is 0.987 or 98.7%

Prediction Assignment

```
answers <- predict(model, final_testing)
answers <- as.character(answers)
answers
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

```
# Using the test program code provided
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)
        }
}
```

```r
pml_write_files(answers)
```