PracticalML\_CourseProject

Angela Frolov

March 14, 2019

## Overview

Devices such as Jawbone Up, Nike FuelBand, and Fitbit make possible to collect large amounts of moovement data. Mostly this data quantifies how much of a particular activity people do, but rarely quantifies how well they do it. In this project we use data collected from accelerometers on the belts, forearms, arms, and dumbells of 6 participants, who were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: <http://web.archive.org/web/20161224072740/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 ( the variable "classe") in which this participants did the exercise.

## Libraries

# Loading Libraries.  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(e1071)  
library(gbm)

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(rpart)

## Getting, cleaning and partitioning data

# Reading in data.  
  
train\_in <- read.csv(file = "pml-training.csv", header = T, na.strings = c("NA","#DIV/0!",""))  
test\_in <- read.csv(file = "pml-testing.csv", header = T, na.strings = c("NA","#DIV/0!",""))  
  
# Checking dimensions of sets  
  
dim(train\_in)

## [1] 19622 160

dim(test\_in)

## [1] 20 160

# Cleaning data: removing the first seven (not affecting our predictions) variables,   
# and variables with majority of NA values.  
train\_in <- train\_in[, -c(1:7)]  
test\_in <- test\_in[, -c(1:7)]  
  
# colMeans(is.na()) shows that all columns with NA values have more then 95% of them.  
  
mostly\_NA\_train <- colMeans(is.na(train\_in)) > 0.95  
train\_in <- train\_in[, mostly\_NA\_train==F]  
  
mostly\_NA\_test <- colMeans(is.na(test\_in)) > 0.95  
test\_in <- test\_in[, mostly\_NA\_test==F]  
  
# Testing if variables are the same in both sets.They are.  
names(train\_in) %in% names(test\_in)

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [12] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [23] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [34] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE  
## [45] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

# Partitioning train\_in data into testing and training sets  
  
inTrain <- createDataPartition(train\_in$classe, p=0.75, list = FALSE)  
myTrain <- train\_in[inTrain, ]  
myTest <- train\_in[-inTrain, ]  
  
# Checking dimensions of sets  
  
dim(myTrain)

## [1] 14718 53

dim(myTest)

## [1] 4904 53

## Setting seed

set.seed(46)

## Training models

The models, that will be used for training and testing on the training data set, are Classification Trees, Random Forests and Support Vector Machines. Then based on the performance results, the model with the highest accuracy will be used for prediction on the testing data set.

## Classification Trees

To train the model a 5-fold cross validation will be used.

# Predicting with trees.  
controlTR <- trainControl(method="cv", number=5, verboseIter=FALSE)  
  
modFit\_tree <- train(classe~., method="rpart", data=myTrain, trControl=controlTR)  
  
print(modFit\_tree$finalModel)

## n= 14718   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)   
## 2) roll\_belt< 130.5 13482 9308 A (0.31 0.21 0.19 0.18 0.11)   
## 4) pitch\_forearm< -33.95 1194 7 A (0.99 0.0059 0 0 0) \*  
## 5) pitch\_forearm>=-33.95 12288 9301 A (0.24 0.23 0.21 0.2 0.12)   
## 10) magnet\_dumbbell\_y< 439.5 10403 7476 A (0.28 0.18 0.24 0.19 0.11)   
## 20) roll\_forearm< 123.5 6484 3859 A (0.4 0.18 0.18 0.16 0.063) \*  
## 21) roll\_forearm>=123.5 3919 2622 C (0.077 0.18 0.33 0.24 0.18) \*  
## 11) magnet\_dumbbell\_y>=439.5 1885 936 B (0.032 0.5 0.044 0.22 0.2) \*  
## 3) roll\_belt>=130.5 1236 11 E (0.0089 0 0 0 0.99) \*

modPredict\_tree <- predict(modFit\_tree, newdata=myTest)  
  
confMatrix\_tree <- confusionMatrix(myTest$classe,modPredict\_tree)  
  
print(confMatrix\_tree)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1268 21 103 0 3  
## B 378 337 234 0 0  
## C 399 26 430 0 0  
## D 384 146 274 0 0  
## E 114 114 267 0 406  
##   
## Overall Statistics  
##   
## Accuracy : 0.4978   
## 95% CI : (0.4837, 0.5119)  
## No Information Rate : 0.5186   
## P-Value [Acc > NIR] : 0.9983   
##   
## Kappa : 0.3437   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.4986 0.52329 0.32875 NA 0.99267  
## Specificity 0.9462 0.85634 0.88181 0.8361 0.88988  
## Pos Pred Value 0.9090 0.35511 0.50292 NA 0.45061  
## Neg Pred Value 0.6366 0.92238 0.78316 NA 0.99925  
## Prevalence 0.5186 0.13132 0.26672 0.0000 0.08340  
## Detection Rate 0.2586 0.06872 0.08768 0.0000 0.08279  
## Detection Prevalence 0.2845 0.19352 0.17435 0.1639 0.18373  
## Balanced Accuracy 0.7224 0.68981 0.60528 NA 0.94127

Accuracy of classification trees model is 0.4977569, with the out-of-sample error 0.5.

## Random Forest

To train the model a 3-fold cross validation will be used.

# Predicting with random forests.  
  
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)  
modFit\_rf <- train(classe ~ ., data=myTrain, method="rf", trControl=controlRF)  
modFit\_rf$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: 27  
##   
## OOB estimate of error rate: 0.52%  
## Confusion matrix:  
## A B C D E class.error  
## A 4182 3 0 0 0 0.0007168459  
## B 16 2825 7 0 0 0.0080758427  
## C 0 7 2554 6 0 0.0050642774  
## D 0 2 21 2386 3 0.0107794362  
## E 0 1 4 7 2694 0.0044345898

modPred\_rf <- predict(modFit\_rf, newdata=myTest)  
confMatrix\_rf <- confusionMatrix(myTest$classe, modPred\_rf)  
  
print(confMatrix\_rf)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1393 1 0 0 1  
## B 6 942 1 0 0  
## C 0 4 849 2 0  
## D 0 0 11 791 2  
## E 0 0 0 3 898  
##   
## Overall Statistics  
##   
## Accuracy : 0.9937   
## 95% CI : (0.991, 0.9957)  
## No Information Rate : 0.2853   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.992   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9957 0.9947 0.9861 0.9937 0.9967  
## Specificity 0.9994 0.9982 0.9985 0.9968 0.9993  
## Pos Pred Value 0.9986 0.9926 0.9930 0.9838 0.9967  
## Neg Pred Value 0.9983 0.9987 0.9970 0.9988 0.9993  
## Prevalence 0.2853 0.1931 0.1756 0.1623 0.1837  
## Detection Rate 0.2841 0.1921 0.1731 0.1613 0.1831  
## Detection Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837  
## Balanced Accuracy 0.9976 0.9965 0.9923 0.9953 0.9980

Accuracy of random forests model is 0.9936786, with out-of-sample error 0.01.

## Support Vector Machines

# Predicting with SVM  
  
modFit\_svm <- svm(classe ~ ., data = myTrain, cross=3)  
print(modFit\_svm)

##   
## Call:  
## svm(formula = classe ~ ., data = myTrain, cross = 3)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 1   
## gamma: 0.01923077   
##   
## Number of Support Vectors: 6656

modPred\_svm <- predict(modFit\_svm, newdata = myTest)  
confMatrix\_svm <- confusionMatrix(myTest$classe, modPred\_svm)  
  
print(confMatrix\_svm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1392 1 1 0 1  
## B 61 873 15 0 0  
## C 4 22 819 9 1  
## D 1 0 65 738 0  
## E 0 8 28 28 837  
##   
## Overall Statistics  
##   
## Accuracy : 0.95   
## 95% CI : (0.9436, 0.956)  
## No Information Rate : 0.2973   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9367   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.9547 0.9657 0.8825 0.9523 0.9976  
## Specificity 0.9991 0.9810 0.9909 0.9840 0.9843  
## Pos Pred Value 0.9978 0.9199 0.9579 0.9179 0.9290  
## Neg Pred Value 0.9812 0.9922 0.9731 0.9910 0.9995  
## Prevalence 0.2973 0.1843 0.1892 0.1580 0.1711  
## Detection Rate 0.2838 0.1780 0.1670 0.1505 0.1707  
## Detection Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837  
## Balanced Accuracy 0.9769 0.9734 0.9367 0.9681 0.9909

Accuracy of Support Vector Machines Model is 0.9500408, with out-of-sample error 0.05.

Based on the data from confusion matrices, the Random Forest data model gives the highest accuracy, and, therefore, shall be used for prediction on our final test set.

## Predicting with Random Forests model

# Using random forests model on test data.  
  
finalResult\_rf <- predict(modFit\_rf, newdata = test\_in)  
print(finalResult\_rf)

## [1] 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