Practical machine learning - Wearables sensors analysis

Eran Shlomo

February 15, 2017

## Objective

In this project we will train predictor to identify activity type, based on wearable sensor information collected. To run the code you will need the datasets,The training data for this project are available here:

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

The test data are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

## requirements

1.Download the datasets into code execution folder. 2.Install rpart,e1071,randomForest and caret packages.

## Preparing the data

Setting working dir and loading:

library(rpart)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(e1071)  
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

#uncomment if running outside of r markdown   
#this.dir <- dirname(parent.frame(2)$ofile)  
#setwd(this.dir)  
org\_testing = read.csv("./pml-testing.csv",na.strings=c("NA","#DIV/0!",""))  
org\_training = read.csv("./pml-training.csv",na.strings=c("NA","#DIV/0!",""))

Lets split the training data into training/validation (testing)

inTrain = createDataPartition(org\_training$classe, p = 3/4, list=FALSE)  
training = org\_training[ inTrain,]  
testing = org\_training[ -inTrain,]

Now lets clean the data: 1. Remove zero variance variables (A LOT OF NA in the dataset) 2. remove id col 3. clear mostly NA fields (creates a lot of un explained variance)

zvars <- nearZeroVar(training)#find meaningless   
training <- training[,-zvars] #remove meaningless  
training <- training[c(-1)] #remove id col  
  
trainingTemp <- training  
for(i in 1:length(training)) { #for every column in the training dataset  
 if( sum( is.na( training[, i] ) ) /nrow(training) >= .6 ) {   
 for(j in 1:length(trainingTemp)) {  
 if( length( grep(names(training[i]), names(trainingTemp)[j]) ) ==1) { #if the columns are the same:  
 trainingTemp <- trainingTemp[ , -j] #Remove that column  
 }   
 }   
 }  
}  
training<-trainingTemp  
clean\_cols <- colnames(training)  
clean\_cols\_no\_classe<-clean\_cols[-which(clean\_cols %in% c("classe"))]  
org\_testing<-org\_testing[,clean\_cols\_no\_classe]  
testing <- testing[,clean\_cols]

And finally alinging classes and levels: Classes:

for (i in 1:length(org\_testing) ) {  
 for(j in 1:length(training)) {  
 if (names(org\_testing[i])==names(training[j]))  
 {  
 class(org\_testing[i]) <- class(training[j])  
 }  
 #if( length( grep(names(training[i]), names(names(training[2]))[j]) ) ==1) {  
 # class(org\_testing[j]) <- class(training[i])  
 #}   
 }   
}

levels:

common <- intersect(names(training), names(org\_testing))  
for (p in common)  
{   
 if (class(training[[p]]) == "factor")   
 {   
 levels(org\_testing[[p]]) <- levels(training[[p]])   
 }   
}

# Lets test some prediction models

## Recursive Partitioning and Regression Trees

regressionTreeModel <- rpart(classe ~ ., data=training, method="class")  
regressionTreePrediction <- predict(regressionTreeModel, testing, type = "class")  
confusionMatrix(regressionTreePrediction, testing$classe)

## $positive  
## NULL  
##   
## $table  
## Reference  
## Prediction A B C D E  
## A 1342 47 3 1 0  
## B 41 785 57 35 0  
## C 12 114 783 134 31  
## D 0 3 8 500 46  
## E 0 0 4 134 824  
##   
## $overall  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.8633768 0.8270871 0.8534460 0.8728729 0.2844617   
## AccuracyPValue McnemarPValue   
## 0.0000000 NaN   
##   
## $byClass  
## Sensitivity Specificity Pos Pred Value Neg Pred Value Precision  
## Class: A 0.9620072 0.9854659 0.9633884 0.9849046 0.9633884  
## Class: B 0.8271865 0.9663717 0.8551198 0.9588560 0.8551198  
## Class: C 0.9157895 0.9281304 0.7290503 0.9812010 0.7290503  
## Class: D 0.6218905 0.9860976 0.8976661 0.9300667 0.8976661  
## Class: E 0.9145394 0.9655259 0.8565489 0.9804668 0.8565489  
## Recall F1 Prevalence Detection Rate  
## Class: A 0.9620072 0.9626973 0.2844617 0.2736542  
## Class: B 0.8271865 0.8409213 0.1935155 0.1600734  
## Class: C 0.9157895 0.8118196 0.1743475 0.1596656  
## Class: D 0.6218905 0.7347539 0.1639478 0.1019576  
## Class: E 0.9145394 0.8845947 0.1837276 0.1680261  
## Detection Prevalence Balanced Accuracy  
## Class: A 0.2840538 0.9737366  
## Class: B 0.1871941 0.8967791  
## Class: C 0.2190049 0.9219599  
## Class: D 0.1135808 0.8039941  
## Class: E 0.1961664 0.9400326  
##   
## $mode  
## [1] "sens\_spec"  
##   
## $dots  
## list()  
##   
## attr(,"class")  
## [1] "confusionMatrix"

## Support vector machine - linear kernel

svmModel<-svm(classe~.,data=training,kernel="linear")  
svmPrediction <- predict(svmModel, testing)  
accuracy=sum(svmPrediction==testing$classe)/dim(testing)[[1]]  
print(accuracy)

## [1] 0.9070147

## Support vector machine - radial kernel

svmModel<-svm(classe~.,data=training,kernel="radial")  
svmPrediction <- predict(svmModel, testing)  
accuracy=sum(svmPrediction==testing$classe)/dim(testing)[[1]]  
print(accuracy)

## [1] 0.9492251

## Random forest trees

randomForestModel <- randomForest(classe ~. , data=training,na.action=na.exclude)  
randomForestPrediction <- predict(randomForestModel, testing, type = "class")  
confusionMatrix(randomForestPrediction, testing$classe)

## $positive  
## NULL  
##   
## $table  
## Reference  
## Prediction A B C D E  
## A 1395 0 0 0 0  
## B 0 949 0 0 0  
## C 0 0 855 1 0  
## D 0 0 0 803 0  
## E 0 0 0 0 901  
##   
## $overall  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.9997961 0.9997421 0.9988644 0.9999948 0.2844617   
## AccuracyPValue McnemarPValue   
## 0.0000000 NaN   
##   
## $byClass  
## Sensitivity Specificity Pos Pred Value Neg Pred Value Precision  
## Class: A 1.0000000 1.000000 1.0000000 1.0000000 1.0000000  
## Class: B 1.0000000 1.000000 1.0000000 1.0000000 1.0000000  
## Class: C 1.0000000 0.999753 0.9988318 1.0000000 0.9988318  
## Class: D 0.9987562 1.000000 1.0000000 0.9997562 1.0000000  
## Class: E 1.0000000 1.000000 1.0000000 1.0000000 1.0000000  
## Recall F1 Prevalence Detection Rate  
## Class: A 1.0000000 1.0000000 0.2844617 0.2844617  
## Class: B 1.0000000 1.0000000 0.1935155 0.1935155  
## Class: C 1.0000000 0.9994155 0.1743475 0.1743475  
## Class: D 0.9987562 0.9993777 0.1639478 0.1637439  
## Class: E 1.0000000 1.0000000 0.1837276 0.1837276  
## Detection Prevalence Balanced Accuracy  
## Class: A 0.2844617 1.0000000  
## Class: B 0.1935155 1.0000000  
## Class: C 0.1745514 0.9998765  
## Class: D 0.1637439 0.9993781  
## Class: E 0.1837276 1.0000000  
##   
## $mode  
## [1] "sens\_spec"  
##   
## $dots  
## list()  
##   
## attr(,"class")  
## [1] "confusionMatrix"

# Results

## Selected model

With 99% accuracy, random forest was selected as preffered regression.

## Running test set,predicting lables

randomForestTestPredict <- predict(randomForestModel, org\_testing, type = "class")  
print(randomForestTestPredict)

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