Practical Machine Learning - Final Project

Luigi

17th June 2016

Prediction Assignment Writeupless

Background

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

Data

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

```r
# load the libraries
library(caret)
```

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

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

## Warning: package 'ggplot2' was built under R version 3.1.3
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.1

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

```r
# load the data
train<- read.csv("~/Downloads/pml-training.csv", header=TRUE)
test <- read.csv("~/Downloads/pml-testing.csv", header=TRUE)
```

Then we partition the train dataset.

```r
set.seed(7)
inTrain <- createDataPartition(train$classe, p = .75, list=FALSE)
myTraining <- train[inTrain, ]
myTesting <- train[-inTrain, ]
#check dimensions
dim(myTraining); dim(myTesting)
```

```
## [1] 14718   160
```

```
## [1] 4904  160
```

Note that the original “train” and “test” datasets differs only by the columns “classe” (present only in “train”) and “problem_id” (present only in “test”). Furthermore since I like to keep track of the changes, I put a number in the dataset names after every change.

Note also that I work only on the dataset “myTraining” and just remove the same columns in the other one.

Next step we can remove the near zero variance variables.

```r
#calculate near zero variance variables
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
#remove relative variables in all datasets
myTraining1 <- myTraining[,nzv$nzv==FALSE]
myTesting1 <- myTesting[,nzv$nzv==FALSE]
myTest1<-test[,nzv$nzv==FALSE]
```

This datasets contain lots of NA so we remove the column where almost all the rows has NA as value.

```r
#sum of na by columns
NAind<-apply(myTraining1,2,function(x) {sum(is.na(x))})
length(which(NAind==0)) #variables with no NAs: 59
```

```
## [1] 59
```

```r
myTraining2 <- myTraining1[,which(NAind == 0)]
myTesting2<-myTesting1[,which(NAind==0)]
myTest2<-myTest1[,which(NAind==0)]

#I decided to remove also the first 5 columns since they have no useful data(e.g. name)
myTraining3<-myTraining2[,-c(1:5)]
myTesting3<-myTesting2[,-c(1:5)]
myTest3<-myTest2[,-c(1:5)]
```

Now we train the model, after some tests I decided to use random forest.

```r
fit <- randomForest(classe ~. , data=myTraining3)
#predict on myTesting
predictionsB1 <- predict(fit, myTesting3, type = "class")
#check the results
confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  948    0    0    0
##          C    0    1  855    4    0
##          D    0    0    0  800    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9973, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   1.0000   0.9950   0.9989
## Specificity            1.0000   1.0000   0.9988   0.9998   1.0000
## Pos Pred Value         1.0000   1.0000   0.9942   0.9988   1.0000
## Neg Pred Value         1.0000   0.9997   1.0000   0.9990   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1743   0.1631   0.1835
## Detection Prevalence   0.2845   0.1933   0.1754   0.1633   0.1835
## Balanced Accuracy      1.0000   0.9995   0.9994   0.9974   0.9994
```

From the confusion matrix we have a good result with 99% accuracy, thus we can finally predict on the test dataset.

```
mytest<-test[,c(setdiff(names(myTraining2),"classe"))]
pred<-predict(fit,myTest3)
```

The prediction are

```
pred
```

```
##  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
```