Practical Machine Learning Assignment

Kheng

21 Mar 2015

An Analysis of the Weight Lifting Exercises Dataset

==

Summary

The task of the project was to predict how well subjects performed weigh lifting excercises based on data collected form accelerometers attached to the person performing the exercises. The data set consists of data form six different people and the outcome is classified into five different categories. So, this is a supervised learning task, and the goal is to prduce a calssifier that orreclty classifies 20 samples provided as a testing set that needs to submitted for grading.

Random forrest algorithm usually performs rather well on a task like this, so that was chosen as the first algorithm to try. If the performance is not satisfacgtory, another algorithm will be tried

Data Source

The data is taken from the Human Activity Recognition programme at Groupware.The links for the training and test data are given below

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

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

Prepare the Setting

```
library(Hmisc)
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: survival
## Loading required package: splines
## Loading required package: Formula
## Loading required package: ggplot2
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(caret)
```

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

```
library(randomForest)
```

```
## 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
```

```
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators
## Loading required package: parallel
```

```
set.seed(8888)
options(warn=-1)
```

Process and Analyse Data

Data is loaded for the provided training and test data with replacing "#DIV/0!" with an NA.

```


training_data <- read.csv("/Users/Kheng/Documents/Data Sceince/Practical Machine Learning/pml-training.
testing_data <- read.csv("/Users/Kheng/Documents/Data Sceince/Practical Machine Learning/pml-testing.cs
```

Prepare datasets

Data Cleaning Data is to reformat to 8 columns and remove the non contributor data from the datasets.

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

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

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

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

Develop Prediction Model

First part of the model is to split training data to training and validation set.

```r
idx <- createDataPartition(y=model_data$classe, p=0.75, list=FALSE )
training <- model_data[idx,]
testing <- model_data[-idx,]
```

5 random forests algorithm with 150 trees each will be built and parallel processing will be used to build this model.

```r
registerDoParallel()
x <- training[-ncol(training)]
y <- training$classe

rf <- foreach(ntree=rep(150, 6), .combine=randomForest::combine, .packages='randomForest') %dopar% {
randomForest(x, y, ntree=ntree)
}
```

Validate the Model

The following matrix shows the training and testing accuracy using the model built.

```r
predictions1 <- predict(rf, newdata=training)
confusionMatrix(predictions1,training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843     
##     P-Value [Acc > NIR] : < 2.2e-16  
##                                      
##                   Kappa : 1          
##  Mcnemar's Test P-Value : NA         
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
predictions2 <- predict(rf, newdata=testing)
confusionMatrix(predictions2,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    3    0    0
##          C    0    0  852   11    0
##          D    0    0    0  792    3
##          E    0    0    0    1  898
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9963          
##                  95% CI : (0.9942, 0.9978)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9954          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9965   0.9851   0.9967
## Specificity            1.0000   0.9992   0.9973   0.9993   0.9998
## Pos Pred Value         1.0000   0.9968   0.9873   0.9962   0.9989
## Neg Pred Value         1.0000   1.0000   0.9993   0.9971   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1935   0.1737   0.1615   0.1831
## Detection Prevalence   0.2845   0.1941   0.1760   0.1621   0.1833
## Balanced Accuracy      1.0000   0.9996   0.9969   0.9922   0.9982
```

Conclusion

The model built using the randomForest algorithm is pretty accurate, with 99% testing accuracy.

Results

Using the generated model on the testing set 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)
  }
}


x <- testing_data
x <- x[feature_set[feature_set!='classe']]
answers <- predict(rf, newdata=x)

answers
```

```
##  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
```

```
pml_write_files(answers)
```