Human Activiry Recognition

Himank Jain

07/08/2019

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. We 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 The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Synopsis

In this analysis We use har dataset to predict if the excercise done by participants was right or not. The outcome variable classe is a factor with 5 levels : A B C D E aPrticipants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions: exactly according to the specification (Class A) throwing the elbows to the front (Class B) lifting the dumbbell only halfway (Class C) lowering the dumbbell only halfway (Class D) throwing the hips to the front (Class E)?

We Use Random forest and Decision trees two train our Models, cross validate the results and predict on a test set of 20 observations.

Cross Validation:

Cross Validation is done by subsampling our trainSet data set into training(75%) and validation(25%) datasets. The model is trained on training dataset and validated on validation dataset. The test set is used to predict the outcome for 20 observations.

```
set.seed(123)
trainset<-read.csv("C:/Users/himan/Desktop/HAR-Analysis/pml-training.csv")
testing<-read.csv("C:/Users/himan/Desktop/HAR-Analysis/pml-testing.csv")
dim(trainset)
```

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

```
table(trainset$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Loading Required Libraries:

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

Partitioning the TrainingSet:

```
partition<-createDataPartition(y=trainset$classe,p=0.75,list=F)
training<-trainset[partition,]
validation<-trainset[-partition,]
```

Cleaning And Preprocessing And Feature Selection:

Removing near-zero variance variables:

```
training<-training[,-nearZeroVar(training)]
```

Removing Descriptive Variables:

```
Cl<-c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
  "cvtd_timestamp", "new_window", "num_window")

training <- training[,!names(training) %in% Cl]
```

Removing Variables with more than 60% NA values:

```
training[training==""] <- NA
naproportion<-sapply(training,function(x) sum(is.na(x))/nrow(training))
training<-training[!(naproportion>0.60)]
dim(training)
```

```
## [1] 14718    53
```

Training Model: Decision Tree

Traning With Decision Tree:

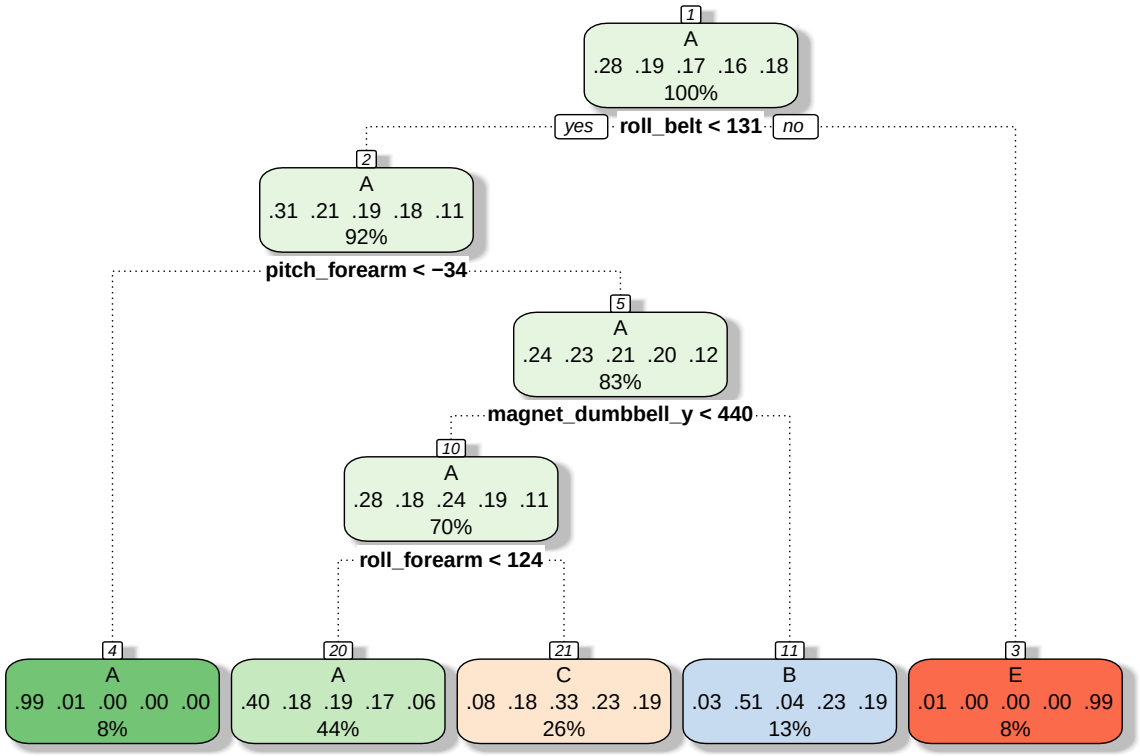
```
modelrp<-train(classe~.,data=training,method='rpart')
```

Prediction on training Model and accuracy on traning Set:

```
predictrp<-predict(modelrp,training)
print(confusionMatrix(predictrp,training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3811 1202 1207 1100  397
##          B   58  966   85  433  370
##          C  303  680 1275  879  715
##          D    0    0    0    0    0
##          E   13    0    0    0 1224
##
## Overall Statistics
##
##                Accuracy : 0.4944
##                  95% CI : (0.4863, 0.5025)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3388
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9106  0.33919  0.49669   0.0000  0.45233
## Specificity            0.6292  0.92030  0.78792   1.0000  0.99892
## Pos Pred Value         0.4938  0.50523  0.33100      NaN  0.98949
## Neg Pred Value         0.9466  0.85304  0.88110   0.8361  0.89007
## Prevalence             0.2843  0.19350  0.17441   0.1639  0.18386
## Detection Rate         0.2589  0.06563  0.08663   0.0000  0.08316
## Detection Prevalence   0.5243  0.12991  0.26172   0.0000  0.08405
## Balanced Accuracy      0.7699  0.62974  0.64230   0.5000  0.72562
```

```
fancyRpartPlot(modelrp$finalModel)
```



Accuracy on Validation set:

```
print(confusionMatrix(predict(modelrp,validation),validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1271  384  380  349  127
##          B   23  320   23  135  116
##          C  100  245  452  320  251
##          D    0    0    0    0    0
##          E    1    0    0    0  407
##
## Overall Statistics
##
##                Accuracy : 0.4996
##                  95% CI : (0.4855, 0.5137)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3468
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9111  0.33720  0.52865   0.0000  0.45172
## Specificity            0.6466  0.92491  0.77377   1.0000  0.99975
## Pos Pred Value         0.5062  0.51864  0.33041      NaN  0.99755
## Neg Pred Value         0.9482  0.85328  0.88603   0.8361  0.89012
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2592  0.06525  0.09217   0.0000  0.08299
## Detection Prevalence   0.5120  0.12582  0.27896   0.0000  0.08320
## Balanced Accuracy      0.7789  0.63105  0.65121   0.5000  0.72574
```

Decision Trees had an accuracy of 0.4944 on training set and 0.4996 on validation Set.

Training Model: Random Forest

Training with Random Forest:

```
modelrf<-train(classe~.,data=training,method='rf',ntree=10)
```

Prediction on training Model and Accuracy on training Set:

```
predictrf<-predict(modelrf,training)
print(confusionMatrix(predictrf,training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    5    0    0    0
##          B    0 2843    1    0    0
##          C    0    0 2566    3    0
##          D    0    0    0 2409    1
##          E    0    0    0    0 2705
##
## Overall Statistics
##
##                Accuracy : 0.9993
##                  95% CI : (0.9988, 0.9997)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            1.0000   0.9982   0.9996   0.9988   0.9996
## Specificity            0.9995   0.9999   0.9998   0.9999   1.0000
## Pos Pred Value         0.9988   0.9996   0.9988   0.9996   1.0000
## Neg Pred Value         1.0000   0.9996   0.9999   0.9998   0.9999
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1932   0.1743   0.1637   0.1838
## Detection Prevalence   0.2847   0.1932   0.1745   0.1637   0.1838
## Balanced Accuracy      0.9998   0.9991   0.9997   0.9993   0.9998
```

Accuracy on Validation Set:

```
print(confusionMatrix(predict(modelrf,validation),validation$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    6    0    0    0
##          B    3  941   10    1    0
##          C    0    2  840   14    3
##          D    0    0    5  788    4
##          E    0    0    0    1  894
##
## Overall Statistics
##
##                Accuracy : 0.99
##                  95% CI : (0.9868, 0.9926)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9874
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9916   0.9825   0.9801   0.9922
## Specificity            0.9983   0.9965   0.9953   0.9978   0.9998
## Pos Pred Value         0.9957   0.9853   0.9779   0.9887   0.9989
## Neg Pred Value         0.9991   0.9980   0.9963   0.9961   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1919   0.1713   0.1607   0.1823
## Detection Prevalence   0.2851   0.1947   0.1752   0.1625   0.1825
## Balanced Accuracy      0.9981   0.9940   0.9889   0.9890   0.9960
```

Random Forest had an accuracy of 0.9992 on training set and 0.9904 on validation Set.

Conclusion and Prediction:

The validation set accuracy (out-of-sample) for random forest 0.9904 was far greater than out-of-sample accuracy for decision tree 0.4996.

Prediction on test set:

```
predictions<-predict(modelrf,testing)
predictions
```

```
##  [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
```