Prediction Project - Human Activiy Recognition

Aixa Rodriguez Salan

15/July/2017

Introduction

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 Source

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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

```
setwd("D:/Data Science/8-Practical Machine Learning/Week4/Project")
if(!dir.exists("./data")){dir.create("./data")}
if(!file.exists("./data/pml-training.csv")){
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                "./data/pml-training.csv")
  }
if(!file.exists("./data/pml-testing.csv")){
  download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                "./data/pml-testing.csv")
  }

training_data<-read.csv("./data/pml-training.csv")
testing_data<-read.csv("./data/pml-testing.csv")

dim(training_data)
```

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

```
dim(testing_data)
```

```
## [1]  20 160
```

Firt set all the data with NULL values to NA and then remove all columns containing NA's and last remove some columns that are not related to dependant variables

```
training_data[training_data==""]<-"NA"
testing_data[testing_data==""]<-"NA"

training_data<-training_data[,colSums(is.na(training_data))==0]
testing_data<-testing_data[,colSums(is.na(testing_data))==0]

col_del<-c('X','user_name','raw_timestamp_part_1','raw_timestamp_part_2',
           'cvtd_timestamp','new_window','num_window')
training_data<-training_data[,-which(names(training_data) %in% col_del)]
testing_data<-testing_data[,-which(names(testing_data) %in% col_del)]

dim(training_data)
```

```
## [1] 19622    53
```

```
dim(testing_data)
```

```
## [1] 20 53
```

And last split training_data into a 60% training and 40% testing data set used to make the cross validation.

```
set.seed(9910)
inTrain<-createDataPartition(training_data$classe,p=0.6,list=FALSE)
dt_training<-training_data[inTrain,]
dt_testing<-training_data[-inTrain,]

dim(dt_training)
```

```
## [1] 11776    53
```

```
dim(dt_testing)
```

```
## [1] 7846   53
```

Prediction Algorithms

Now we are trying to predict the model

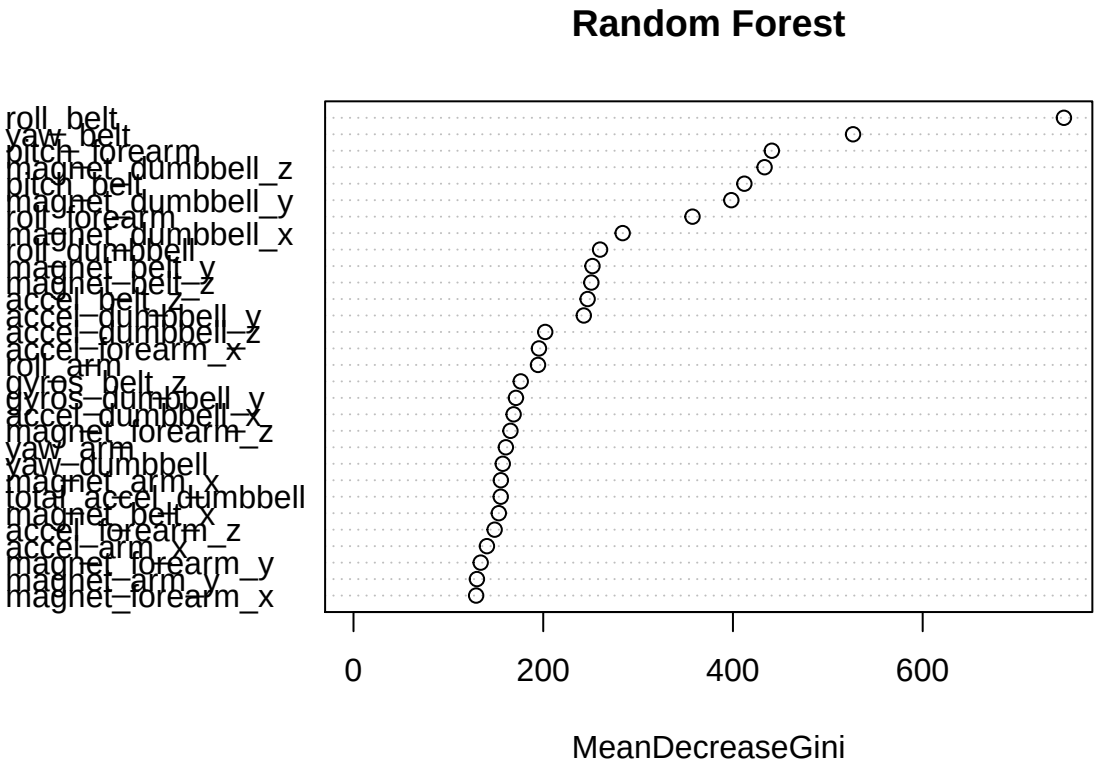
Random Forest (rf)

```
mod_rf<-randomForest(classe~.,data=dt_training,ntree=500)
print(mod_rf)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = dt_training, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.62%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    3    1    0    0 0.001194743
```

```
## B   17 2260    2    0    0 0.008336990
## C    0   14 2038    2    0 0.007789679
## D    0    0   23 1905    2 0.012953368
## E    0    0    3    6 2156 0.004157044
```

```
varImpPlot(mod_rf,main ="Random Forest")
```




```
pred_rf<-predict(mod_rf,newdata=dt_testing)
confusionMatrix(pred_rf,dt_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2227    7    0    0    0
##          B    2 1502    1    0    0
##          C    3    9 1365   20    0
##          D    0    0    2 1262    4
##          E    0    0    0    4 1438
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.9913, 0.995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

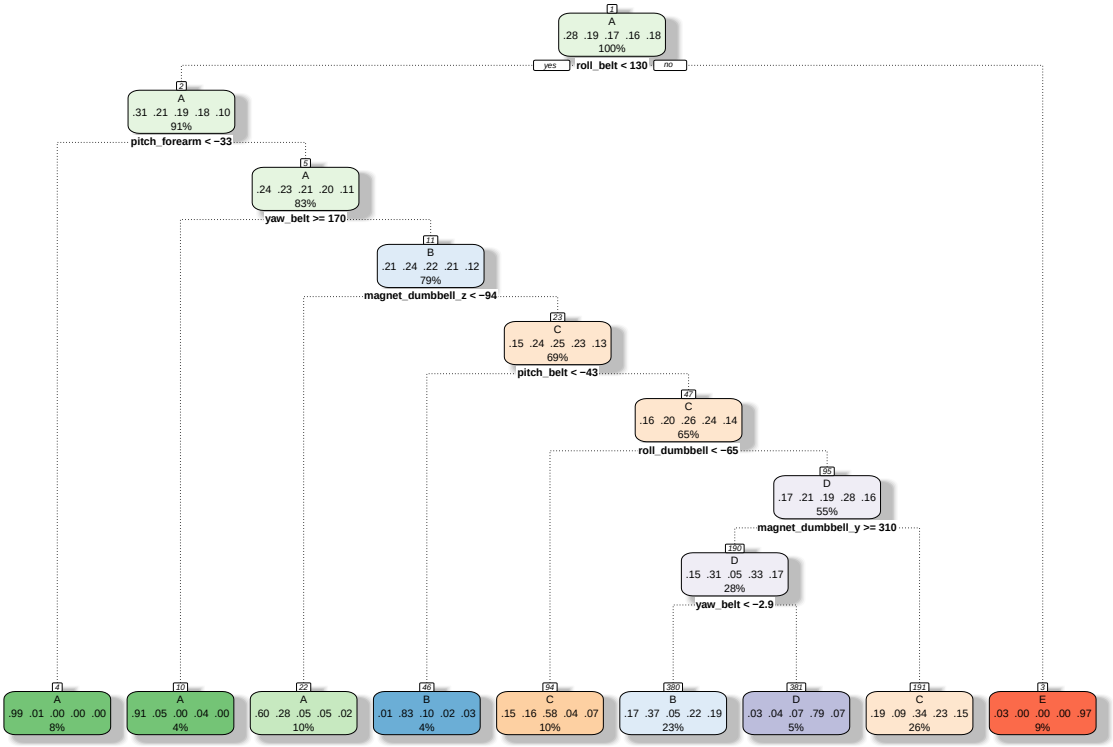
```
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9895   0.9978   0.9813   0.9972
## Specificity            0.9988   0.9995   0.9951   0.9991   0.9994
## Pos Pred Value         0.9969   0.9980   0.9771   0.9953   0.9972
## Neg Pred Value         0.9991   0.9975   0.9995   0.9964   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1914   0.1740   0.1608   0.1833
## Detection Prevalence   0.2847   0.1918   0.1781   0.1616   0.1838
## Balanced Accuracy      0.9983   0.9945   0.9964   0.9902   0.9983
```

CART (rpart)

```
mod_rpt<-train(classe~.,data=dt_training,method="rpart")
print(mod_rpt)
```

```
## CART
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02414570  0.5800590  0.46311052
##   0.03913147  0.4935699  0.33978014
##   0.11948268  0.3179167  0.05125749
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0241457.
```

```
fancyRpartPlot(mod_rpt$finalModel)
```



Rattle 2017−jul−16 09:51:54 Icha

```
pred_rpt<-predict(mod_rpt,newdata=dt_testing)
confusionMatrix(pred_rpt,dt_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1352  234   36   71   22
##          B  310  946  127  410  374
##          C  516  323 1186  482  369
##          D   17   15   19  323   22
##          E   37    0    0    0  655
##
## Overall Statistics
##
##                Accuracy : 0.5687
##                  95% CI : (0.5577, 0.5797)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4582
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.6057   0.6232   0.8670  0.25117  0.45423
## Specificity            0.9353   0.8070   0.7391  0.98887  0.99422
## Pos Pred Value         0.7883   0.4365   0.4124  0.81566  0.94653
## Neg Pred Value         0.8565   0.8993   0.9634  0.87074  0.88999
## Prevalence             0.2845   0.1935   0.1744  0.16391  0.18379
## Detection Rate         0.1723   0.1206   0.1512  0.04117  0.08348
## Detection Prevalence   0.2186   0.2762   0.3666  0.05047  0.08820
## Balanced Accuracy      0.7705   0.7151   0.8030  0.62002  0.72423
```

Linear Discriminant (lda)

```
mod_lda<-train(classe~.,data=dt_training,method="lda",verbose=FALSE)
```

```
## Loading required package: MASS
```

```
print(mod_lda)
```

```
## Linear Discriminant Analysis
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.6997    0.6201197
```

```
pred_lda<-predict(mod_lda,newdata=dt_testing)
confusionMatrix(pred_lda,dt_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1834  230  131   85   50
##          B   47  973  115   46  272
##          C  177  188  913  152  157
##          D  165   61  174  954  135
##          E    9   66   35   49  828
##
## Overall Statistics
##
##                Accuracy : 0.7012
##                  95% CI : (0.691, 0.7114)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6219
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8217   0.6410   0.6674   0.7418   0.5742
## Specificity            0.9116   0.9241   0.8960   0.9184   0.9752
## Pos Pred Value         0.7871   0.6696   0.5753   0.6407   0.8389
## Neg Pred Value         0.9278   0.9148   0.9273   0.9478   0.9105
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2337   0.1240   0.1164   0.1216   0.1055
## Detection Prevalence   0.2970   0.1852   0.2023   0.1898   0.1258
## Balanced Accuracy      0.8667   0.7826   0.7817   0.8301   0.7747
```

Support Vector Machines (svm)

```
mod_svm<-svm(classe~.,data=dt_training)
print(mod_svm)
```

```
##
## Call:
## svm(formula = classe ~ ., data = dt_training)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.01923077
##
## Number of Support Vectors:  5754
```

```
pred_svm<-predict(mod_svm,newdata=dt_testing)
confusionMatrix(pred_svm,dt_testing$classe)$overall[4]
```

```
## AccuracyUpper
##     0.9490314
```

As we can see using differents algorithms to predict and comparing the accuracy of each one Random Forest it's the most accuracy with a 99%.

	Accuracy	Kappa	AccuracyLower	AccuracyUpper	OutSampleError
Random Forest	0.9933724	0.9916168	0.9913178	0.9950464	0.0066276
Cran RPart	0.5686974	0.4581549	0.5576512	0.5796925	0.4313026
LDA	0.7012490	0.6219291	0.6909841	0.7113643	0.2987510
SVM	0.9440479	0.9291198	0.9387315	0.9490314	0.0559521

Predicts on Testing Set

Now using Random Forest to predict the outcome of the testing set

```
pred_testing<-predict(mod_rf,testing_data,type="class")
print(pred_testing)
```

```
##  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
```

```
pred_df<-data.frame(pred_testing)
write.csv(pred_df,file="./data/pml-predicts.csv",row.names=FALSE)
```