Prediction Assigment Write Up

Diego Tangassi

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 Loading and Exploratory Analysis

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 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.

Data

We need te following R-packages to complete the analysis, as well as the downloaded data (with NA's removed).

```
library(knitr); library(randomForest); library(corrplot)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rpart.plot); library(rattle);library(caret); library(rpart)
```

```
## Loading required package: rpart
```

```
## Rattle: A free graphical interface for data mining with R.
## Versión 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

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

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

```
Train <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Test  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
set.seed(10)

training <- read.csv(url(Train)); testing  <- read.csv(url(Test))
inTrain  <- createDataPartition(training$classe, p = 0.7, list = FALSE)
TrainSet <- training[inTrain, ]; TestSet  <- training[-inTrain, ]

nas <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -nas]; TestSet  <- TestSet[, -nas]

AllNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, AllNA == FALSE]; TestSet  <- TestSet[, AllNA == FALSE]

TrainSet <- TrainSet[, -(1:5)]; TestSet  <- TestSet[, -(1:5)]
```

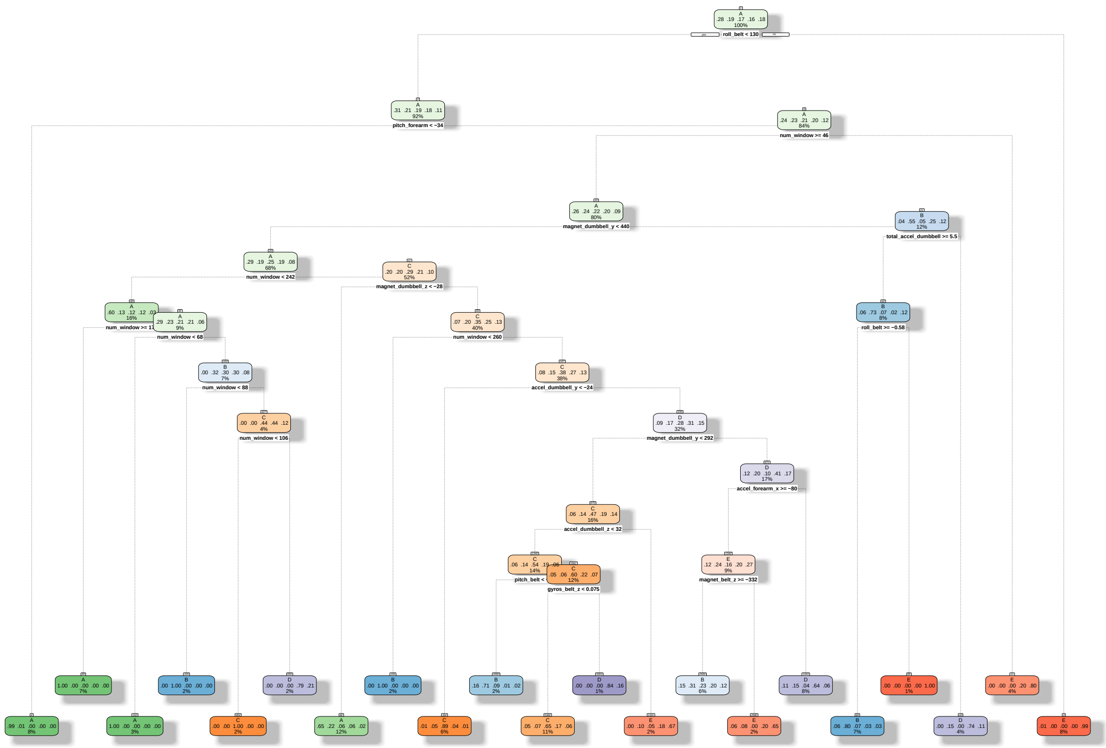
Prediction Models

We are using three methods (Decision Tree, Generalized Boosted Model and Random Forests) to know which one fit better the data(higher accuracy), and use it for the prediction.

Decision Trees

```
set.seed(10)
TreeModel <- rpart(classe ~ ., data = TrainSet, method = "class")
fancyRpartPlot(TreeModel)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2016−oct−16 18:30:52 diegot

```r
DecisionTree <- predict(TreeModel, newdata = TestSet, type = "class")
DecisionTreeCMatrix <- confusionMatrix(DecisionTree, TestSet$classe)
DecisionTreeCMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1489  140   47   57   12
##          B   87  808  119   76   72
##          C   23   56  840  132   45
##          D   63  110   17  607   72
##          E   12   25    3   92  881
##
## Overall Statistics
##
##                Accuracy : 0.7859
##                  95% CI : (0.7752, 0.7963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7287
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8895   0.7094   0.8187   0.6297   0.8142
## Specificity            0.9392   0.9254   0.9473   0.9468   0.9725
## Pos Pred Value         0.8533   0.6954   0.7664   0.6985   0.8697
## Neg Pred Value         0.9553   0.9299   0.9612   0.9288   0.9587
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2530   0.1373   0.1427   0.1031   0.1497
## Detection Prevalence   0.2965   0.1975   0.1862   0.1477   0.1721
## Balanced Accuracy      0.9143   0.8174   0.8830   0.7882   0.8934
```

Generalized Boosted Model

```
set.seed(10)
TrainGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
GBMModel  <- train(classe ~ ., data = TrainSet, method = "gbm",
                    trControl = TrainGBM, verbose = FALSE)
```

```
## Loading required package: gbm
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

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

```
## Loading required package: splines
```

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

```
## Loaded gbm 2.1.1
```

```
## Loading required package: plyr
```

```
GBMModel$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 44 had non-zero influence.
```

```
GBM <- predict(GBMModel, newdata = TestSet)
GBMCMatrix <- confusionMatrix(GBM, TestSet$classe)
GBMCMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1668    7    0    0    0
##          B    6 1115    8    1    2
##          C    0   17 1010    9    1
##          D    0    0    7  954    9
##          E    0    0    1    0 1070
##
## Overall Statistics
##
##                Accuracy : 0.9884
##                  95% CI : (0.9854, 0.991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9854
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9789   0.9844   0.9896   0.9889
## Specificity            0.9983   0.9964   0.9944   0.9967   0.9998
## Pos Pred Value         0.9958   0.9850   0.9740   0.9835   0.9991
## Neg Pred Value         0.9986   0.9950   0.9967   0.9980   0.9975
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1895   0.1716   0.1621   0.1818
## Detection Prevalence   0.2846   0.1924   0.1762   0.1648   0.1820
## Balanced Accuracy      0.9974   0.9877   0.9894   0.9932   0.9944
```

Random Forest

```r
set.seed(10)
TrainRF <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
RFModel <- train(classe ~ ., data = TrainSet, method = "rf",
                          trControl = TrainRF)
RFModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2648    4    0    0 0.0037622272
## C    0    7 2389    0    0 0.0029215359
## D    0    0    8 2243    1 0.0039964476
## E    0    0    0    4 2521 0.0015841584
```

```r
RF <- predict(RFModel, newdata = TestSet)
RFCMatrix <- confusionMatrix(RF, TestSet$classe)
RFCMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1134    3    0    0
##          C    0    1 1023    2    0
##          D    0    1    0  962    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.998
##                  95% CI : (0.9964, 0.9989)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9974
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9956   0.9971   0.9979   0.9982
## Specificity            0.9993   0.9994   0.9994   0.9994   1.0000
## Pos Pred Value         0.9982   0.9974   0.9971   0.9969   1.0000
## Neg Pred Value         1.0000   0.9989   0.9994   0.9996   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1927   0.1738   0.1635   0.1835
## Detection Prevalence   0.2850   0.1932   0.1743   0.1640   0.1835
## Balanced Accuracy      0.9996   0.9975   0.9982   0.9987   0.9991
```

Applying the Selected Model to the Test Data The accuracy of the 3 regression modeling methods above are:

```
## [1] "Random Forest :  0.997960917587086"
```

```
## [1] "Decision Tree :  0.78589634664401"
```

```
## [1] "GBM :  0.988445199660153"
```

The Random Forest model has the highest Accuracy, so we will be appling such model to predict the testing dataset as shown.

```r
predictTEST <- predict(RFModel, newdata = testing)
predictTEST
```

```
##  [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
```