Practical Machine Learning: Prediction Assignment Writeup

Sagi Greenstine

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, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants.
Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five 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) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.
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).
The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. It is possible to use any of the other variables to predict with. It is necessary to create a report describing how was built the model, how was used cross validation, what the expected out of sample error is, and why were made the choices that were made. The prediction model also should be used to predict 20 different test cases.

Data Processing

Loading and viewing the data

The "Weight Lifting Exercises Dataset" **Source:** Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13). Stuttgart, Germany: ACM SIGCHI, 2013.

Loading the dataset:

```
if(!exists("training")){
        training <- read.csv("./pml-training.csv", row.names = NULL, check.names=F, stringsAsFactors=F,
}
if(!exists("testing")){
        testing <- read.csv("./pml-testing.csv", row.names = NULL, check.names=F, stringsAsFactors=F, he
}
```

The structure of the data:
- **training** dataset has 19622 observations of 160 variables;
- **testing** dataset has 20 observations of 160 variables.
I'll try to predict the outcome of the variable **classe** in the *training* dataset.

Cleaning the data

Firstly, let's remove the columns that filled with the NA values.

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

Now, **training** and **testing** datasets have 19622 and 20 observations (appropriately) of 60 variables.

Also, there are the first seven variables related to the time-series or are not numeric and they are unnecessary for predicting.
Let's remove these columns.

```
trainData <- training[, -c(1:7)]
testData <- testing[, -c(1:7)]
```

Now, **trainData** and **testData** datasets have 19622 and 20 observations (appropriately) of 53 variables, the first 52 variables are the same, and the 53-th variable is *classe* and *problem_id* appropriatelly.

Slicing the data

Now we can split the cleaned training set into a training set (train, 70%) and a validation set (test, 30%). The validation set will be used to conduct cross validation.

```
library(caret); library(lattice); library(ggplot2)
```

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

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

```
set.seed(77777)
inTrain <- createDataPartition(trainData$classe, p=0.70, list=FALSE)
train <- trainData[inTrain, ]
test <- trainData[-inTrain, ]
```

Predictive Models

We fit a predictive model for activity recognition using Classification Trees and Random Forest algorithms.

Classification Trees

We will use 5-fold cross validation when applying the algorithm.

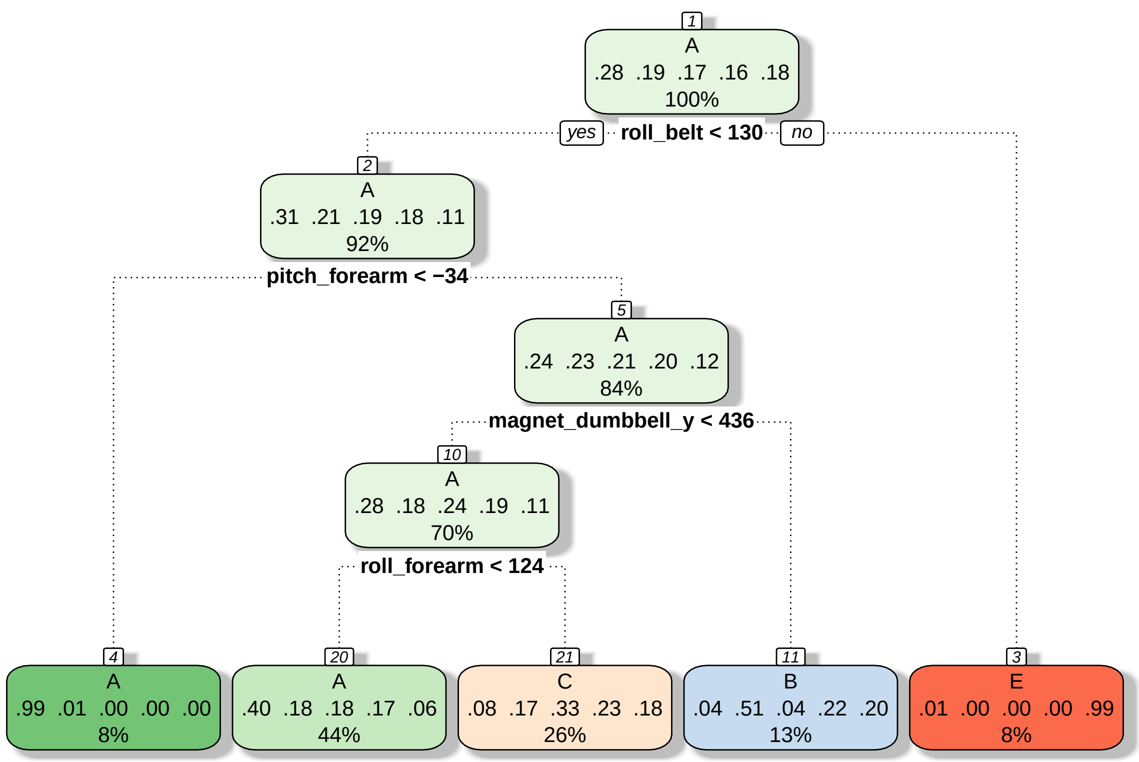
```
library(rpart)
library(rpart.plot)
controlrf <- trainControl(method = "cv", number = 5)
modelrf <- train(classe ~ ., data = train, method = "rpart", trControl = controlrf)
modelrf
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10990, 10990
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa       Accuracy SD  Kappa SD
##   0.03234666  0.5460410  0.41471790  0.04834644   0.07251407
##   0.05984471  0.3907733  0.16626911  0.05556085   0.09330210
##   0.11565456  0.3324551  0.07357513  0.04400472   0.06725617
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03234666.
```

```r
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.0.5 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
fancyRpartPlot(modelrf$finalModel)
```



Rattle 2016−Jan−30 21:53:51 Serg

Predicting outcomes with the validation set and viewing the prediction result:

```
predictrp <- predict(modelrf, test)
confrp <- confusionMatrix(test$classe, predictrp)
confrp
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1521   32  114    0    7
##          B  450  388  301    0    0
##          C  469   32  525    0    0
##          D  408  188  368    0    0
##          E  152  144  299    0  487
##
## Overall Statistics
##
##                Accuracy : 0.4963
##                  95% CI : (0.4835, 0.5092)
##     No Information Rate : 0.5098
##     P-Value [Acc > NIR] : 0.9809
##
##                   Kappa : 0.3426
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5070  0.49490  0.32670       NA  0.98583
## Specificity            0.9470  0.85277  0.88289   0.8362  0.88963
## Pos Pred Value         0.9086  0.34065  0.51170       NA  0.45009
## Neg Pred Value         0.6488  0.91656  0.77732       NA  0.99854
## Prevalence             0.5098  0.13322  0.27307   0.0000  0.08394
## Detection Rate         0.2585  0.06593  0.08921   0.0000  0.08275
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7270  0.67384  0.60479       NA  0.93773
```

```
accurp <- confrp$overall[1]
accurp
```

```
##  Accuracy
## 0.4963466
```

The accuracy rate is 0.5 approximately, thus the out-of-sample error rate is about 0.5. Therefore, using the Classification Tree doesn't predict the outcome *classe* very well.

Random Forests

Now, let's try the Random Forests algorithm.

```r
library(randomForest)
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rfit <- train(classe ~ ., data = train, method = "rf",
              trControl = controlrf)
rfit
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10990, 10989, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9904639  0.9879355  0.001440687  0.001824023
##   27    0.9895176  0.9867381  0.002203689  0.002789806
##   52    0.9849312  0.9809363  0.002754791  0.003486407
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Predicting outcomes with the validation set and viewing the prediction result:

```r
predictrf <- predict(rfit, test)
confrf <- confusionMatrix(test$classe, predictrf)
confrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    0    3    0    0
##          B   10 1124    5    0    0
##          C    0    7 1018    1    0
##          D    0    0    7  954    3
##          E    0    0    0    0 1082
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9939
##                  95% CI : (0.9915, 0.9957)
##     No Information Rate : 0.2856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9923
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9941   0.9938   0.9855   0.9990   0.9972
## Specificity            0.9993   0.9968   0.9984   0.9980   1.0000
## Pos Pred Value         0.9982   0.9868   0.9922   0.9896   1.0000
## Neg Pred Value         0.9976   0.9985   0.9969   0.9998   0.9994
## Prevalence             0.2856   0.1922   0.1755   0.1623   0.1844
## Detection Rate         0.2839   0.1910   0.1730   0.1621   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9967   0.9953   0.9919   0.9985   0.9986
```

```
accurf <- confrf$overall[1]
accurf
```

```
##  Accuracy
## 0.9938828
```

Here, the Random Forest algorithm is better than Classification Tree algorithm. The accuracy rate is 0.994, thus,the out-of-sample error rate is 0.006.
Therefore, we will use the Random Forests model to predict the outcome variable *classe* for the testing set.

Predicting for Test Data Set

```
predict(rfit, testing)
```

```
##  [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
```