Practical Machine Learning Project

Shonda Kuiper

March 11, 2017

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. 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, our goal is 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. We describe how we built our model, how we used cross validation, what we think the expected out of sample error is, and why we made the particular choices.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Getting and cleaning data

After downloading the data, we load it locally

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

```
## Warning: package 'rpart.plot' was built under R version 3.3.3
```

```
library(randomForest);

training <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
testing <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

Remove columns of the training and testing files that contain any missing values and remove the first seven columns from each dataset We now have 53 instead of 160 columns in both datasets The training dataset has 19622 rows and the testing data has 20 rows

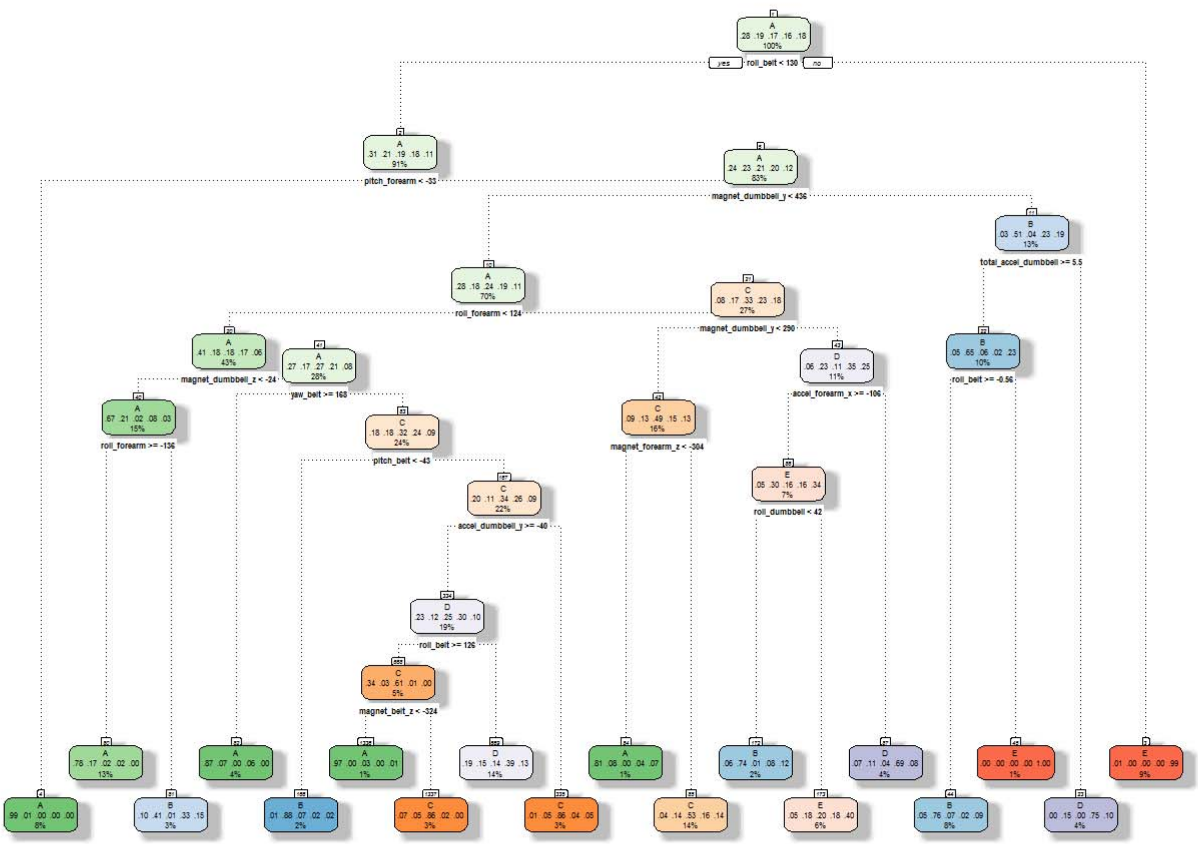
```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
```

Splitting “training” into both a training and testing dataset

```
set.seed(1234)
inTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[inTrain, ]
valid <- training[-inTrain, ]
```

Predicting with Classification Trees

```
CTModel <- rpart(classe ~ ., data=train, method="class")
fancyRpartPlot(CTModel)
```



Rattle 2017-Mar-11 13:42:30 KUIPERS

```
pred1 <- predict(CTModel, valid, type = "class")
confusionMatrix(pred1, valid$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1364  169   24   48   16
##          B   60  581   46   79   74
##          C   52  137  765  129  145
##          D  183  194  125  650  159
##          E   15   58   66   58  688
##
## Overall Statistics
##
##                Accuracy : 0.6879
##                  95% CI : (0.6758, 0.6997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6066
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8148  0.51010   0.7456   0.6743   0.6359
## Specificity            0.9390  0.94543   0.9047   0.8657   0.9590
## Pos Pred Value         0.8415  0.69167   0.6230   0.4958   0.7774
## Neg Pred Value         0.9273  0.88940   0.9440   0.9314   0.9212
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2318  0.09873   0.1300   0.1105   0.1169
## Detection Prevalence   0.2754  0.14274   0.2087   0.2228   0.1504
## Balanced Accuracy      0.8769  0.72776   0.8252   0.7700   0.7974
```

From the confusion matrix, the accuracy rate is 0.688 so the expected out-of-sample error is 100-68.8 = 31.2%.

Predicting with Random Forests

CTModel <- rpart(classe ~ ., data=train, method="class") fancyRpartPlot(CTModel)

```
set.seed(1234)
RFModel <- randomForest(classe ~ ., data=train)
pred2 <- predict(RFModel, valid, type = "class")
confusionMatrix(pred2, valid$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    8    0    0    0
##          B    0 1130    6    0    0
##          C    0    1 1020    4    0
##          D    0    0    0  959    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.9946, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9942   0.9948   0.9991
## Specificity            0.9981   0.9987   0.9990   0.9998   0.9998
## Pos Pred Value         0.9952   0.9947   0.9951   0.9990   0.9991
## Neg Pred Value         1.0000   0.9981   0.9988   0.9990   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1920   0.1733   0.1630   0.1837
## Detection Prevalence   0.2858   0.1930   0.1742   0.1631   0.1839
## Balanced Accuracy      0.9991   0.9954   0.9966   0.9973   0.9994
```

From the confusion matrix, the accuracy rate is 0.999 so the random forest method does predict our outcome of interest, classe, very well. The expected out-of-sample error is 100-99.9 = .01%.

Predicting Results using the pml-testing data

```
pred2.test <- predict(RFModel, testing, type = "class")
pred2.test
```

```
##  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
```

This provided the following output.

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