Practical Machine Learning - Prediction Assignment

Joe Florence

Contents

Practical Machine Learning - Prediction Assignment

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.

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).

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.

Loading Data

Here's the data used for this project. Note that I am using the command `read_csv` to load data. This command (not to be confused with `read.csv`) comes from Hadley Wickham's relatively new tidyverse package.

```r
data_whole <- read_csv("pml-training.csv")
testingSet <- read_csv("pml-testing.csv")
```

Data Cleaning and Dimensional Reduction

In this code chunk, I remove unneccessary or irrelevent columns and features (e.g., those with lots of missing data or not used in the analysis).

```r
    NA_Count = sapply(1:dim(data_whole)[2],function(x)sum(is.na(data_whole[,x])))
    NA_list = which(NA_Count>0)

    data_whole = data_whole[,-NA_list]
    data_whole = data_whole[,-c(1:7)]
    data_whole$classe = factor(data_whole$classe)

    NA_Count1 = sapply(1:dim(testingSet)[2],function(x)sum(is.na(testingSet[,x])))
    NA_list1 = which(NA_Count1>0)
    testingSet = testingSet[,-NA_list]
    testingSet = testingSet[,-c(1:7)]
```

Partitioning the Dataset

Next, I subdivide the training data set to estimate the out of sample prediction error. I partition the data into a training component (60% of the total cases) and a testing component (40% of the total cases). This is seperate from the data in the pml-testing.csv file.

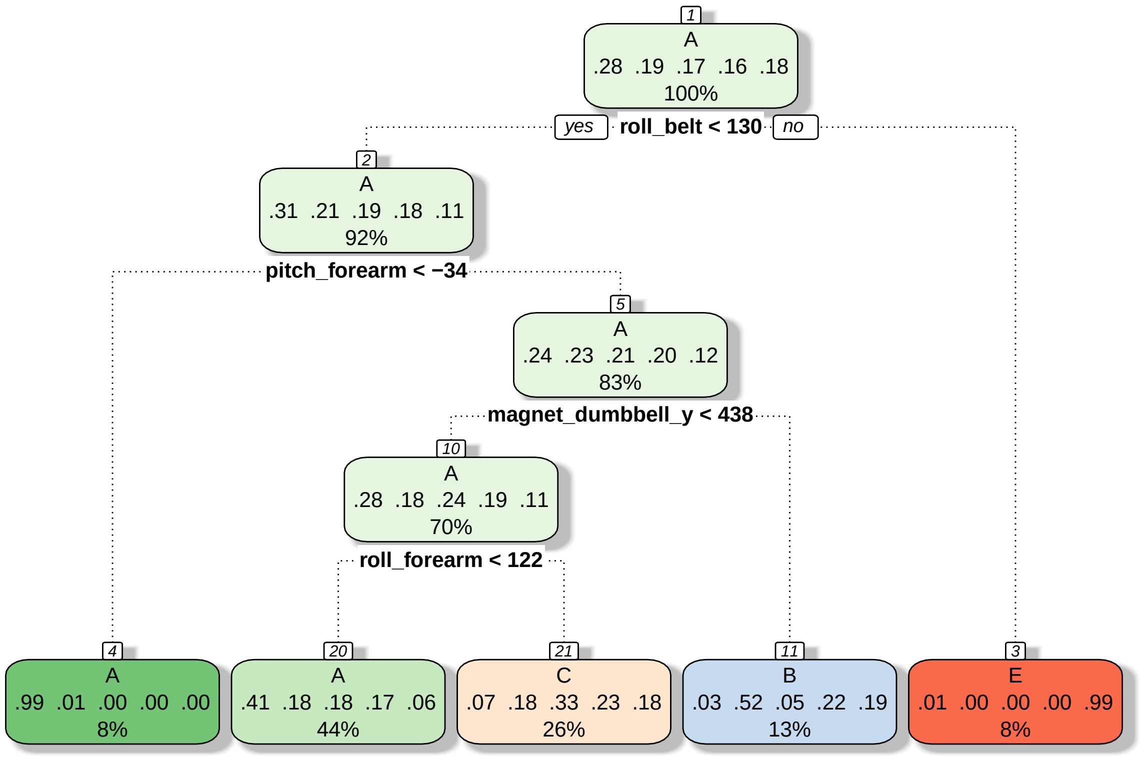
```r
set.seed(02143)

inTrain=createDataPartition(y=data_whole$classe, p=0.6, list=FALSE)
training <-data_whole[inTrain,]
testing <- data_whole[-inTrain,]
```

Predicting with a Decision Tree Model

Using a decision tree, accuracy of around 80% (i.e., not very accurate) is conventionally acceptable.

```r
modfit1 <- train(classe ~ .,method='rpart',data=training)
fancyRpartPlot(modfit1$finalModel)
```



Rattle 2017−Feb−26 15:08:12 Joe

Using this decision tree, here is a confusion matrix of predictions.

```
set.seed(02143)
pred=predict(modfit1,newdata=testing)
z1=confusionMatrix(pred,testing$classe)
z1$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2005  638  638  548  204
##          B   43  495   36  242  207
##          C  179  385  694  496  387
##          D    0    0    0    0    0
##          E    5    0    0    0  644
```

```
z1$overall[1]
```

```
##  Accuracy
## 0.4891665
```

The dismal accuracy for this model fit (~.5) suggests that the data are not and analysts should reject the model fit based on decision tree predictions.

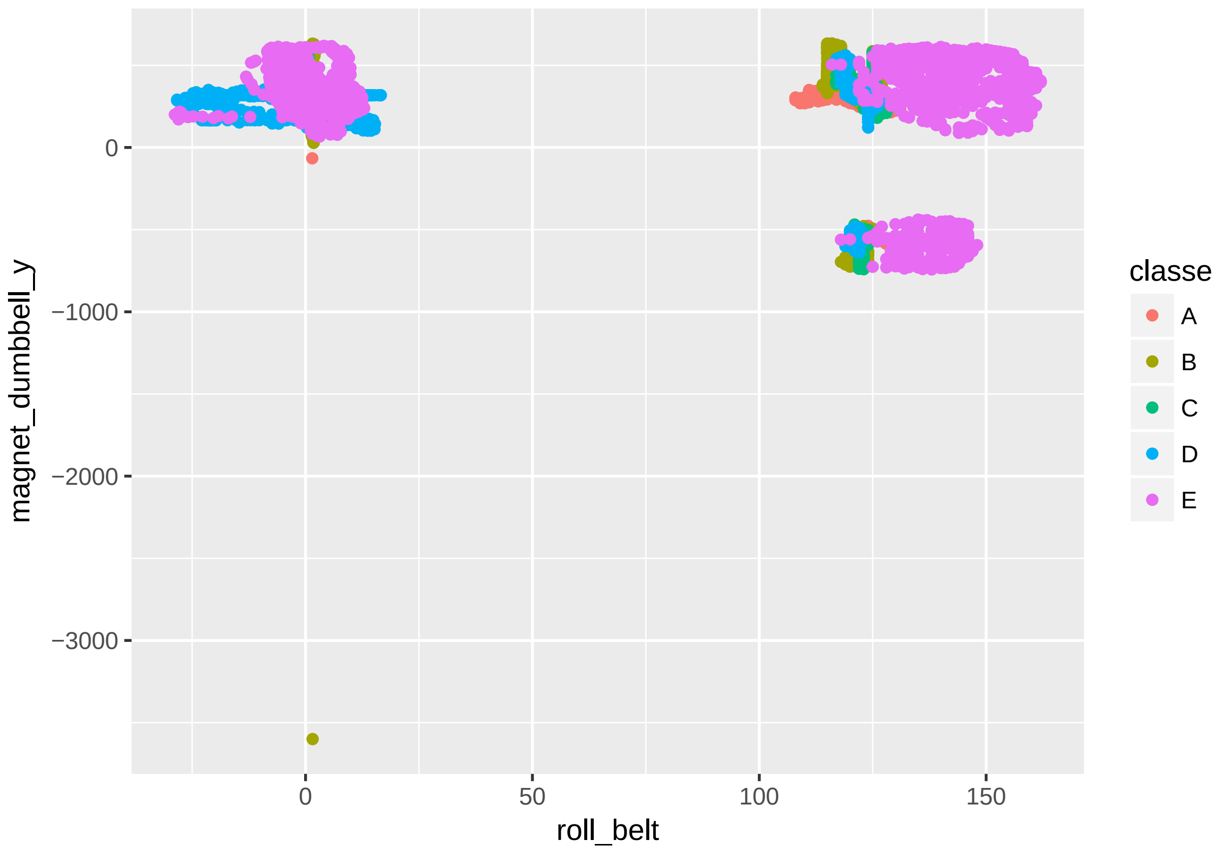
Predicting with a Random Forest Model

Using a random forest model should reduce the out of sample error significantly.

```
set.seed(02143)

modfit2=randomForest(classe~., data=training, method='class')
```

```r
pred2 = predict(modfit2,testing,type='class')
qplot(roll_belt, magnet_dumbbell_y, colour=classe, data=training)
```



Here is a confusion matrix of predictions from the random forest model.

```r
z2=confusionMatrix(pred2,testing$classe)
z2$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   12    0    1    0
##          B    3 1504   12    0    3
##          C    0    2 1351   12    3
##          D    0    0    5 1271    5
##          E    0    0    0    2 1431
```

The overall accuracy of this model is listed here:

```r
z2$overall[1]
```

```
##  Accuracy 
## 0.9923528
```

Conclusion

Based on the accuracy results from the confusion matrix, the random forest model is very good at predicting classification (with 99% accuracy). Because of this, we should expect that 99% of test cases to be correct once submitted.