Practical Machine Learning Course Project

Frederica Janga

1 februari 2017

Background and Introduction

Nowadays it is possible for several apps like Nike, Fitbit and Runkeeper, to track a lot of data about personal activities. These apps are a part of a group of investigators who take measurements about themselves regularly to improve their health or to find patterns in their behavior. One thing that people measure most of the time is how much time they are doing a particular exercise. But they rarely quantify how well the exercise is done.

In this paper, data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants is analyzed. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The goal of this paper is to predict the manner in which the participants did the exercise. The “classe” variable in the training set is used to measure this. In the first part, the model is built and used with cross validation. Then the expected out of sample error is calculated. In the end, the prediction model is used to predict 20 different test cases.

Exploratory data analysis

Read in the data

Read in the needed packages.

```
#Read in the data and clean up the data.
training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
dim(training); dim(testing)
```

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

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

```
#change NA values into zero
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]

#Delete the first 7 columns
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
dim(training); dim(testing)
```

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

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

As you can see after cleaning up the data, the training set consists of 19622 rows and 53 columns. And the testing set consists of 20 rows and 53 columns.

Data splitting

Split the training data into a training and a testing set. (60% trainset, 40% testset)

```
set.seed(1234)
#set the training set 60% and the testing set 40%.
inTrain <- createDataPartition(y=training$classe , p=0.6, list=FALSE)
trainset <- training[inTrain,]
testset <- training[-inTrain,]
```

Model building and out of sample error

R part method and decision tree

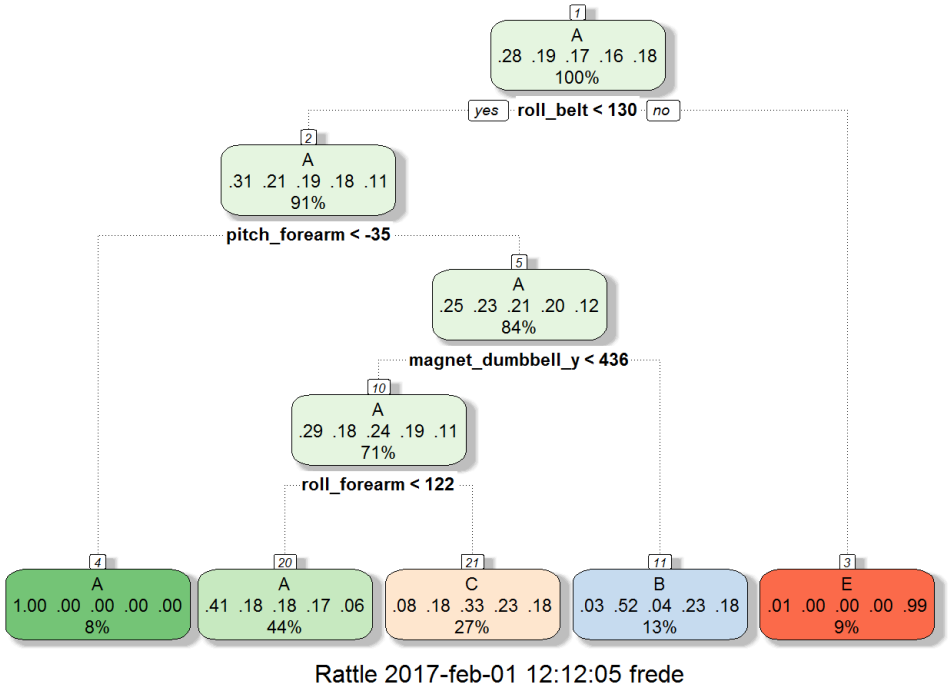
```
#train the data with the rpart method.
```

```
modrpart <- train(classe ~., method="rpart", data=trainset)

print(modrpart$finalModel)
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 11776 8428 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 10774 7436 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.55 919    2 A (1 0.0022 0 0 0) *
##      5) pitch_forearm>=-34.55 9855 7434 A (0.25 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 436.5 8314 5944 A (0.29 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 122.5 5137 3022 A (0.41 0.18 0.18 0.17 0.061) *
##         21) roll_forearm>=122.5 3177 2124 C (0.08 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=436.5 1541  743 B (0.033 0.52 0.039 0.23 0.18) *
##    3) roll_belt>=130.5 1002   10 E (0.01 0 0 0 0.99) *
```

```
#Make a classification tree plot with the rpart method.
fancyRpartPlot(modrpart$finalModel)
```




```
#Calculate the predictions
predictionsrpart <- predict(modrpart, newdata=testset)

#Set up the confusion matrix to check the accuracy.
confusionMatrix(predictionsrpart, testset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2029  638  644  567  209
##          B   44  505   49  232  211
##          C  155  375  675  487  383
##          D    0    0    0    0    0
##          E    4    0    0    0  639
##
## Overall Statistics
##
##                Accuracy : 0.4904
##                  95% CI : (0.4793, 0.5016)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3339
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9091  0.33267  0.49342   0.0000  0.44313
## Specificity            0.6334  0.91530  0.78388   1.0000  0.99938
## Pos Pred Value         0.4965  0.48511  0.32530      NaN  0.99378
## Neg Pred Value         0.9460  0.85114  0.87992   0.8361  0.88852
## Prevalence             0.2845  0.19347  0.17436   0.1639  0.18379
## Detection Rate         0.2586  0.06436  0.08603   0.0000  0.08144
## Detection Prevalence   0.5209  0.13268  0.26447   0.0000  0.08195
## Balanced Accuracy      0.7712  0.62399  0.63865   0.5000  0.72125
```

The accuracy is 0.49. This means the out of sample error is 0.51. From this error we can conclude that the classification tree is not the best choice to predict the classe. Let's take a look at the Random Forest Method.

Random forests

Now use the random forests method to train the data. We expect the random forest method will be a better predicter and will give a higher accuracy.

```
#Use the randomForest method
modforest <- randomForest(classe ~.,data = trainset)

print(modforest)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainset)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.64%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    2    0    0    2 0.001194743
## B   15 2260    4    0    0 0.008336990
## C    0   16 2036    2    0 0.008763389
## D    0    0   25 1903    2 0.013989637
## E    0    0    2    5 2158 0.003233256
```

```
#Calculate the predictions from the random forest method.
predictionsforest <- predict(modforest, newdata=testset)

#Set up the confusion matrix
confusionMatrix(predictionsforest, testset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232   13    0    0    0
##          B    0 1499   11    0    0
##          C    0    6 1354   20    2
##          D    0    0    3 1264    2
##          E    0    0    0    2 1438
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9903, 0.9943)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9875   0.9898   0.9829   0.9972
```

```
## Specificity            0.9977   0.9983   0.9957   0.9992   0.9997
## Pos Pred Value         0.9942   0.9927   0.9797   0.9961   0.9986
## Neg Pred Value         1.0000   0.9970   0.9978   0.9967   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1911   0.1726   0.1611   0.1833
## Detection Prevalence   0.2861   0.1925   0.1761   0.1617   0.1835
## Balanced Accuracy      0.9988   0.9929   0.9927   0.9911   0.9985
```

The accuracy is 0.9913. This means the out of sample error is 0.0087. From this error we can conclude that the random forest method is way better a good predictor then the rpart method.

Prediction case

Let's use the random forest method to predict 20 different test cases.

```
##  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
```

Conclusion

After using the rpart method and the random forest method, the random forest has by far the highest accuracy and is the best model to use for prediction.