practical machine learning - project

radc

December 21, 2018

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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset). Data 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) 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 goal of this project is to predict the manner in which they did the exercise. I this project I will create a report describing how I built my model, how I used cross validation, what the expected out of sample error is, and why I made the choices I did.

First is to load the required r packages and the data. The code below will work only if the file (see link above) is already in the working directory. The data was also cleaned using the code below. I removed varaibles containing a lot of "NA" and the first 5 columns as it not related to the prediction.

```
library(caret)
```

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

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

```
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

```
## The following object is masked from 'package:rattle':
##
##     importance
```

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

```
library(knitr)
library(gbm)
```

```
## Loaded gbm 2.1.4
```

```
library(rmarkdown)

testing<-read.csv("pml-testing.csv")
training<-read.csv("pml-training.csv")
set.seed(2221)


nearzeroind <- nearZeroVar(training, saveMetrics=TRUE)
training <- training[,nearzeroind$nzv==FALSE]
training <- training[, colSums(is.na(training)) == 0]
training <- training[, -c(1:5)]
dim(training)
```

```
## [1] 19622    54
```

Partition was next created. The data was divided as 70% for the training and 30% for testing.

```
inTrain <- createDataPartition(training$classe, p=0.70, list = FALSE)
pmltraining <- training[inTrain,]
pmltesting<- training[-inTrain,]
dim(pmltraining)
```

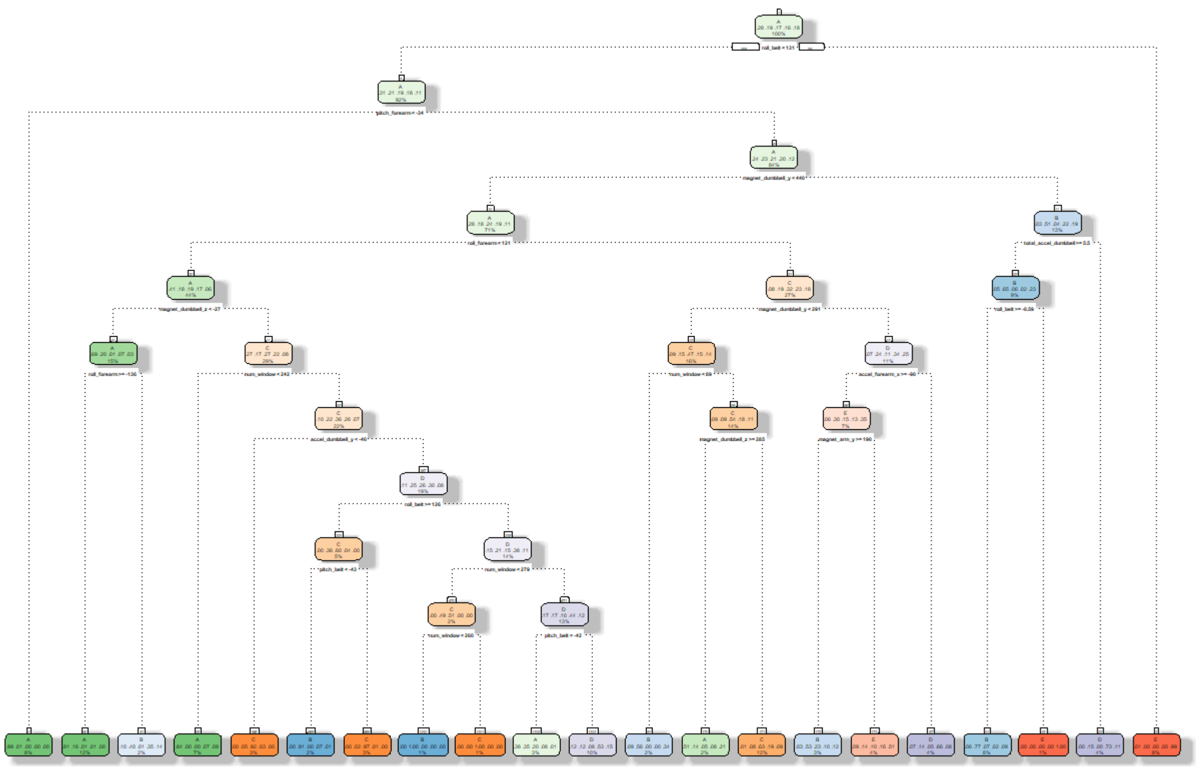
```
## [1] 13737    54
```

```
dim(pmltesting)
```

```
## [1] 5885   54
```

Training using classification tree

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



Rattle 2018-Dec-21 10:23:15 Ena Agustin

```
classificationtree_validation <- predict(classificationtree_model, pmltesting, type = "class")
cm_classficationtree <- confusionMatrix(classificationtree_validation, pmltesting$classe)
cm_classficationtree
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1493  207   42   62   91
##          B   57  704   69   76  117
##          C   12   57  822  146   70
##          D   96  136   58  634  115
##          E   16   35   35   46  689
## 
## Overall Statistics
##                                           
##                Accuracy : 0.7378          
##                  95% CI : (0.7264, 0.749)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.667           
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8919   0.6181   0.8012   0.6577   0.6368
## Specificity            0.9045   0.9328   0.9413   0.9177   0.9725
## Pos Pred Value         0.7879   0.6882   0.7425   0.6102   0.8392
## Neg Pred Value         0.9546   0.9105   0.9573   0.9319   0.9224
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2537   0.1196   0.1397   0.1077   0.1171
## Detection Prevalence   0.3220   0.1738   0.1881   0.1766   0.1395
## Balanced Accuracy      0.8982   0.7754   0.8713   0.7877   0.8047
```

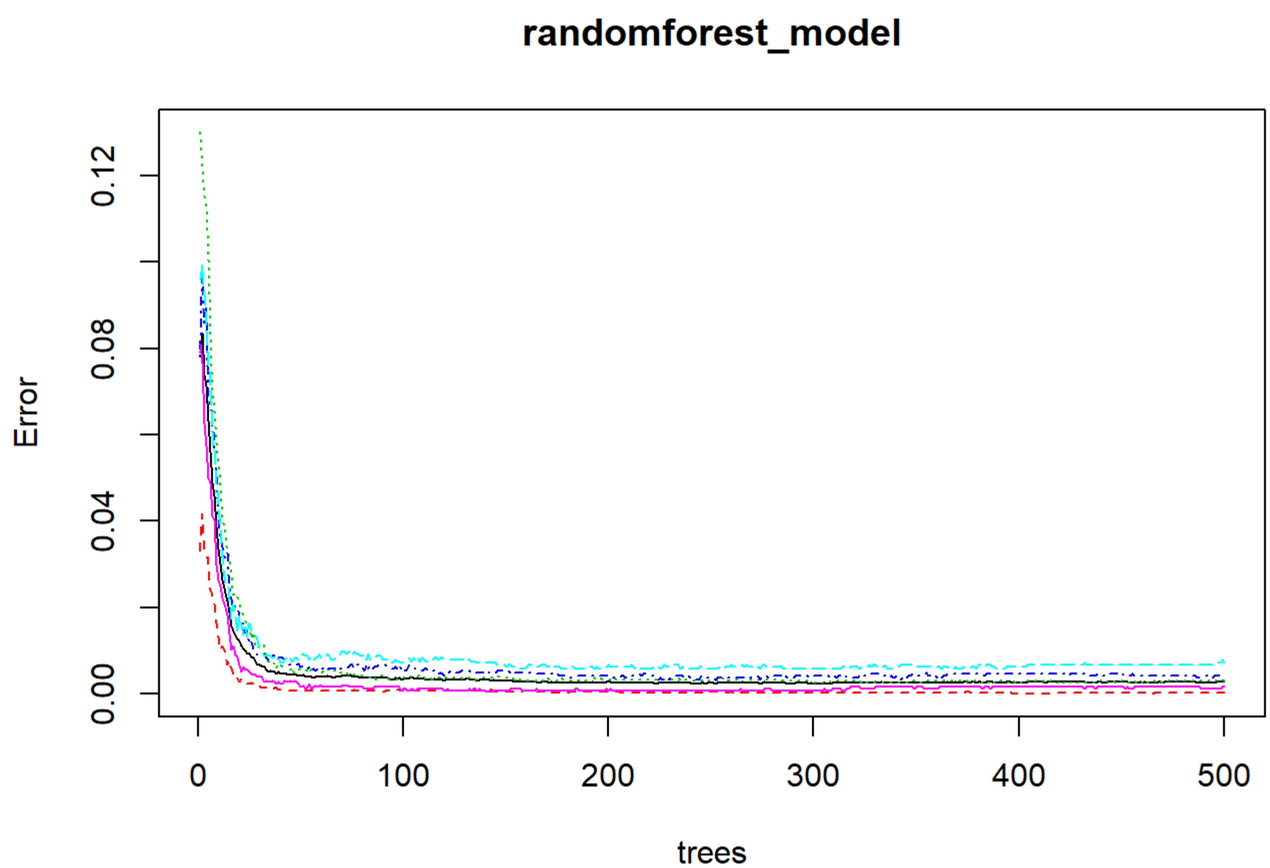
Using the classification tree for prediction, I came up with 73.78% accuracy.

Training using random forest

```
randomforest_model <- randomForest(classe~.,data=pmltraining)

randomforest_validation <- predict(randomforest_model, pmltesting, type="class")
cm_randomforest <- confusionMatrix(pmltesting$classe, randomforest_validation)
cm_randomforest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    1 1137    1    0    0
##          C    0    1 1024    1    0
##          D    0    0    2  962    0
##          E    0    0    0    8 1074
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9991   0.9971   0.9907   1.0000
## Specificity            1.0000   0.9996   0.9996   0.9996   0.9983
## Pos Pred Value         1.0000   0.9982   0.9981   0.9979   0.9926
## Neg Pred Value         0.9998   0.9998   0.9994   0.9982   1.0000
## Prevalence             0.2846   0.1934   0.1745   0.1650   0.1825
## Detection Rate         0.2845   0.1932   0.1740   0.1635   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9997   0.9993   0.9983   0.9952   0.9992
```



Using the random forest for prediction, I came up with 99.76% accuracy, which is really high.

Training using generalized boosted regression

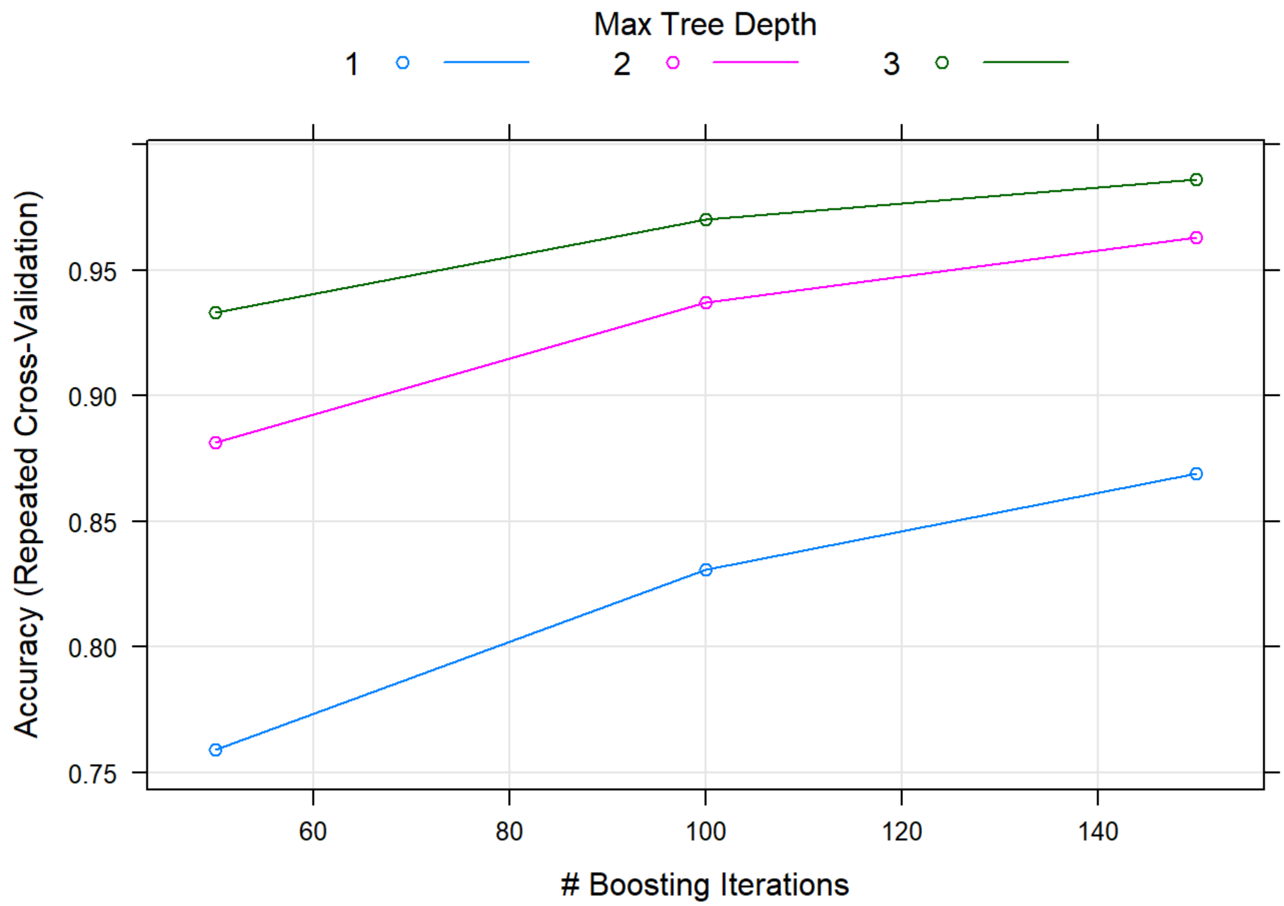
```
control_gbr <- trainControl(method = "repeatedcv", number = 5, repeats = 1)

gbm_model <- train(classe ~ ., data=pmltraining, method = "gbm", trControl = control_gbr, verbos
e = FALSE)
gbm_model
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10990, 10990, 10989, 10990, 10989
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7590451  0.6943771
##   1                  100      0.8306768  0.7856510
##   1                  150      0.8691856  0.8344480
##   2                   50      0.8814156  0.8498704
##   2                  100      0.9371042  0.9204131
##   2                  150      0.9631647  0.9533977
##   3                   50      0.9333920  0.9156878
##   3                  100      0.9703719  0.9625181
##   3                  150      0.9860231  0.9823182
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
gbm_validation <- predict(gbm_model, newdata=pmltesting)
cm_gbm <- confusionMatrix(gbm_validation, pmltesting$classe)
cm_gbm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1665    4    0    1    0
##          B    9 1121   11    0    4
##          C    0   12 1011    9    2
##          D    0    2    4  954   15
##          E    0    0    0    0 1061
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9876          
##                  95% CI : (0.9844, 0.9903)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9843          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9842   0.9854   0.9896   0.9806
## Specificity            0.9988   0.9949   0.9953   0.9957   1.0000
## Pos Pred Value         0.9970   0.9790   0.9778   0.9785   1.0000
## Neg Pred Value         0.9979   0.9962   0.9969   0.9980   0.9956
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2829   0.1905   0.1718   0.1621   0.1803
## Detection Prevalence   0.2838   0.1946   0.1757   0.1657   0.1803
## Balanced Accuracy      0.9967   0.9896   0.9903   0.9927   0.9903
```



Using the generalized boosted regression for prediction, I came up with 98.76% accuracy

The random forest gave the hihghest accuracy, thus this model will be used for the testing data. The out-of-sample error is 100% - 99.76% = 0.24% only.

Using the best preciction model on the testing data

```
prediction_testing <- predict(randomforest_model, testing)
prediction_testing
```

```
##  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
```