Practical Machine Learning: Prediction on How Well People Perform Barbell Lifts

HL

12/23/2020

Synopsis

Three machine learning models have been trained in predicting which way people perform barbell lifts correctly or incorrectly in 5 different ways. Based on the existing data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, recursive partitioning classification, gradient boosting and random forest algorithms are used with cross validation. The random forest model gives the best accuracy in prediction.

Introduction

By 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. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

This project' goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants for predicting how well people do barbell lifts correctly or incorrectly in 5 different ways. People regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

Data import and manipulation

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har).

Based on the WLE dataset (http://web.archive.org/web/20161224072740/http://groupware.les.inf.puc-rio.br/static/WLE/WearableComputing_weight_lifting_exercises_biceps_curl_variations.csv), only 53 variables are used for training models and 52 for predicting test data. The observations with any missing value are omitted.

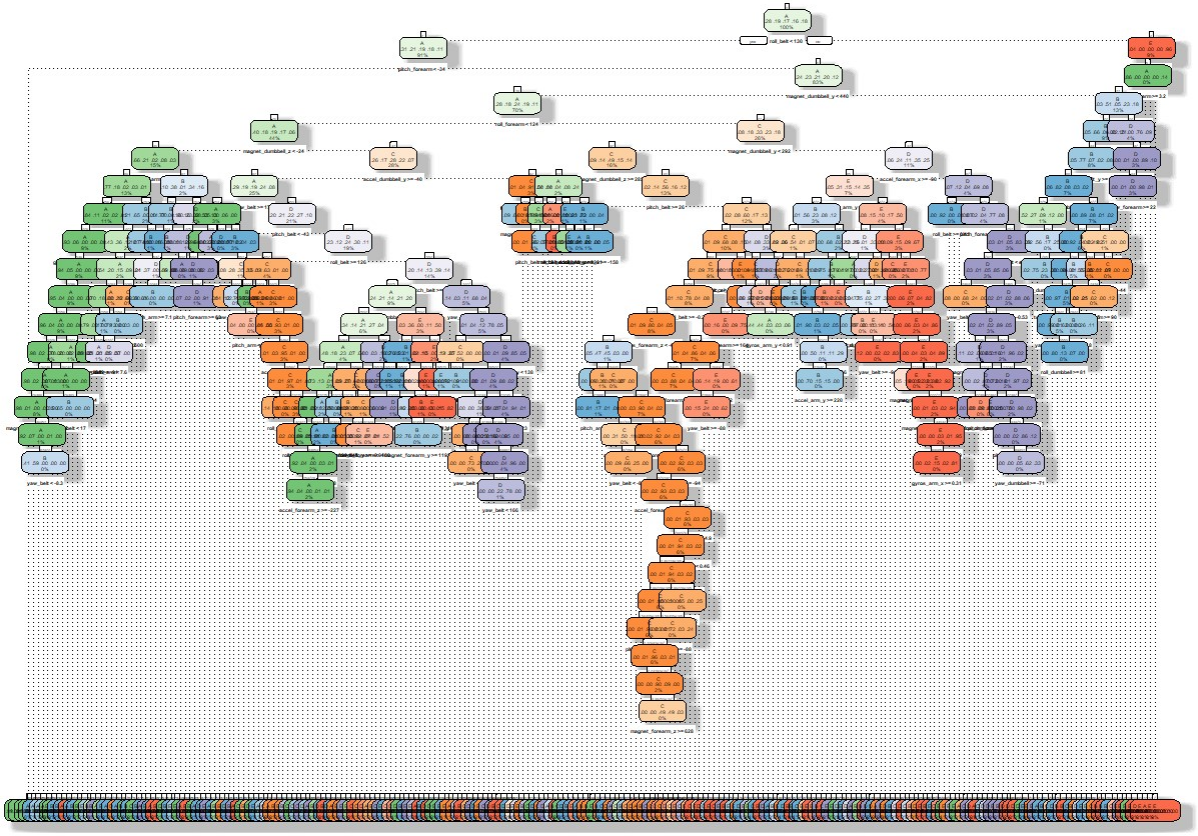
Build three machine learning models

Original train dataset is split to two datasets for training and testing.

The target of prediction is a categorical variable “classe” with value A, B, C, D, or E. The number of predicators is fifty two. Therefore, supervised learning classification method is appropriate for constructing the model. In this project, recursive partitioning classification, gradient boosting and random forest algorithms are selected.

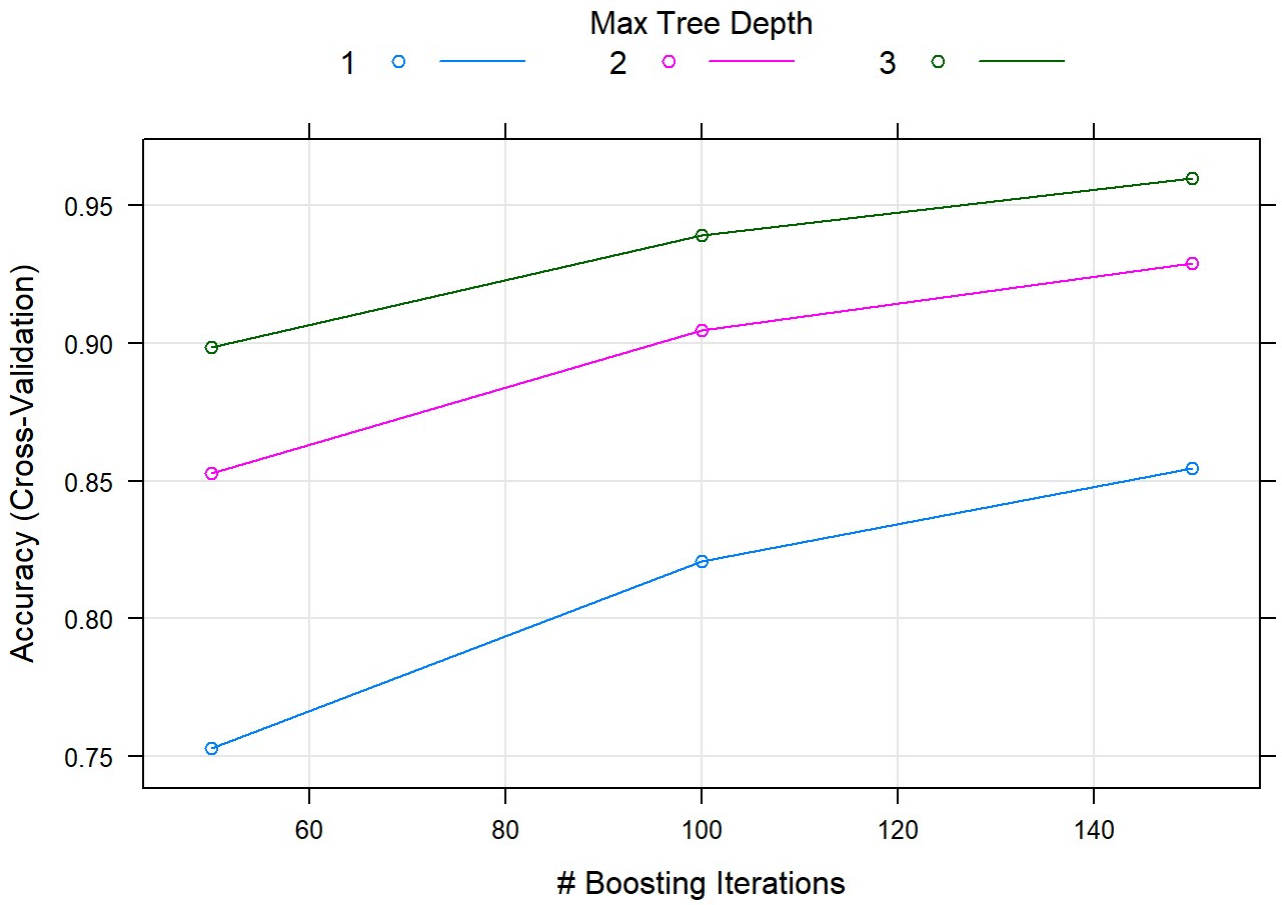
Three fold cross validation is used in training all three models.

```
# train recursive partitioning model
fitRP = rpart(classe ~ ., data = trainData, method = "class", control = rpart.control(cp = 0.0001), xval=3)
fancyRpartPlot(fitRP, sub="Recursive Partitioning Tree")
```

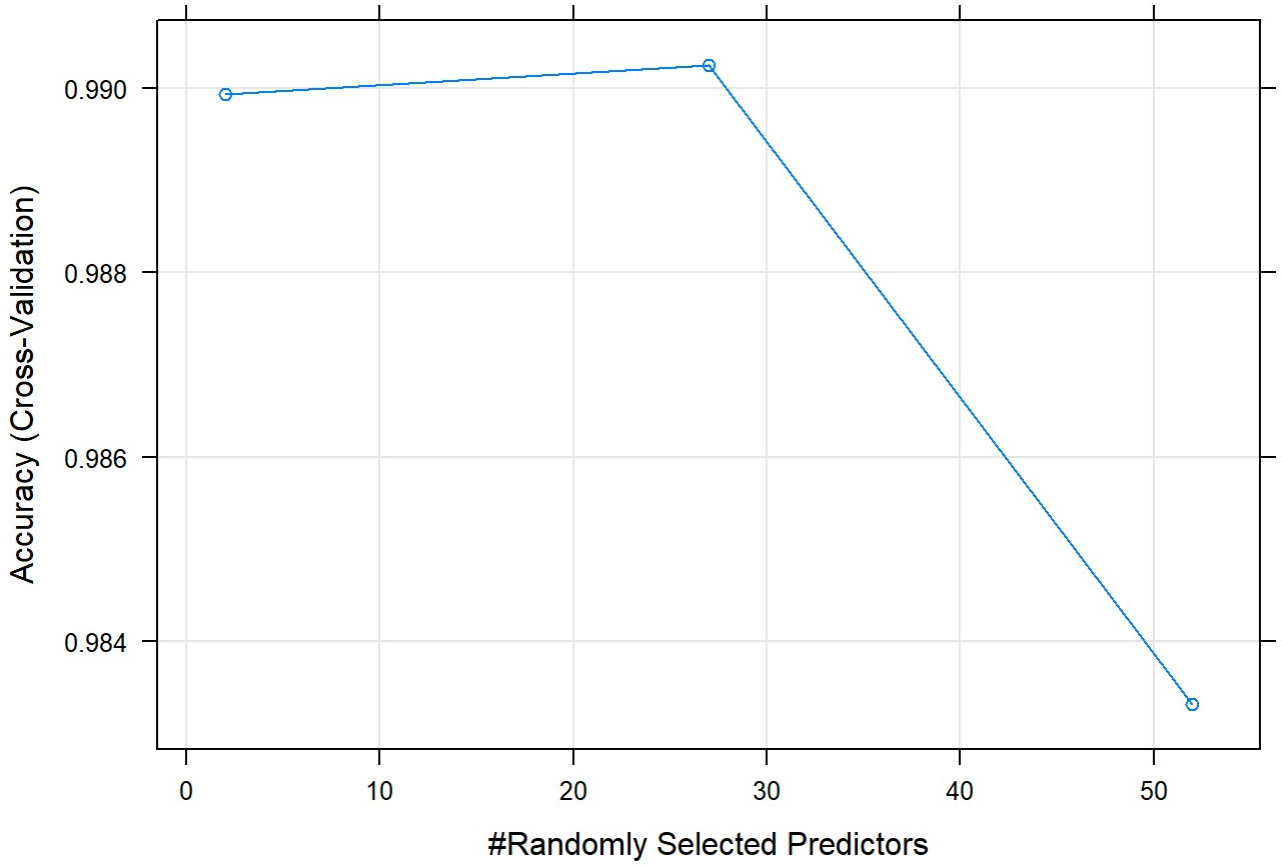


Recursive Partitioning Tree

```
# train gradient boosting model
trainCtrl = caret::trainControl(method = "cv", number=3)
fitGBM = caret::train(classe ~ ., data = trainData, trControl = trainCtrl, method = "gbm", verbose = FALSE)
plot(fitGBM)
```




```
# train random forest model
trainCtrl = caret::trainControl(method = "cv", number=3)
fitRF = caret::train(classe ~ ., data = trainData, trControl = trainCtrl, method = "rf", verbose = FALSE)
plot(fitRF)
```



Evaluate three machine learning models

Confusion matrix of test data is used for evaluating these three models. Other evaluation methods are not addressed here although data can be collected for evaluation.

```
# validate recursive partitioning model
pred = predict(fitRP, testData, type = "class")
confusionMatrix(testData$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1081   18    8    6    3
##          B   23  690   25    8   13
##          C    7   26  632   10    9
##          D    6    6   24  593   14
##          E    0   13    9   10  689
##
## Overall Statistics
##
##                Accuracy : 0.9393
##                  95% CI : (0.9314, 0.9466)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9233
##
##  Mcnemar's Test P-Value : 0.405
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9678   0.9163   0.9054   0.9458   0.9464
## Specificity            0.9875   0.9782   0.9839   0.9848   0.9900
## Pos Pred Value         0.9686   0.9091   0.9240   0.9222   0.9556
## Neg Pred Value         0.9872   0.9801   0.9796   0.9896   0.9878
## Prevalence             0.2847   0.1919   0.1779   0.1598   0.1856
## Detection Rate         0.2756   0.1759   0.1611   0.1512   0.1756
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9776   0.9473   0.9447   0.9653   0.9682
```

```
# validate gradient boosting model
pred = predict(fitGBM, testData)
confusionMatrix(testData$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1088   17    9    1    1
##          B   21  713   25    0    0
##          C    0   24  655    5    0
##          D    0    3   23  609    8
##          E    1    5    6   11  698
##
## Overall Statistics
##
##                Accuracy : 0.9592
##                  95% CI : (0.9525, 0.9652)
##     No Information Rate : 0.2829
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9484
##
##  Mcnemar's Test P-Value : 6.941e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9802   0.9357   0.9123   0.9728   0.9873
## Specificity            0.9900   0.9854   0.9910   0.9897   0.9928
## Pos Pred Value         0.9749   0.9394   0.9576   0.9471   0.9681
## Neg Pred Value         0.9922   0.9845   0.9805   0.9948   0.9972
## Prevalence             0.2829   0.1942   0.1830   0.1596   0.1802
## Detection Rate         0.2773   0.1817   0.1670   0.1552   0.1779
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9851   0.9606   0.9516   0.9813   0.9901
```

```
# validate random forest model
pred = predict(fitRF, testData)
confusionMatrix(testData$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    8  750    1    0    0
##          C    0    0  684    0    0
##          D    0    0    8  634    1
##          E    0    0    3    2  716
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9912, 0.9963)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9926
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9929   1.0000   0.9828   0.9969   0.9986
## Specificity            1.0000   0.9972   1.0000   0.9973   0.9984
## Pos Pred Value         1.0000   0.9881   1.0000   0.9860   0.9931
## Neg Pred Value         0.9971   1.0000   0.9963   0.9994   0.9997
## Prevalence             0.2865   0.1912   0.1774   0.1621   0.1828
## Detection Rate         0.2845   0.1912   0.1744   0.1616   0.1825
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9964   0.9986   0.9914   0.9971   0.9985
```

The accuracy of three models are all greater than 0.9. Under current conditions, the random forest model has the best accuracy greater than 0.99.

The random forest model gives the prediction from the raw test data.

```
pred = predict(fitRF, newdata=dataToPred)
pred
```

```
##  [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
```

Conclusion

Three machine learning models, recursive partitioning classification, gradient boosting and random forest, have been trained with existing data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. Three fold cross validation is used in all three models. The random forest model gives the best accuracy in prediction.

By using the random forest model, the classe values of raw test cases is predicted as follows:

```
##  [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
```

Reference

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.