Practical Machine Learning Week 4 Project

Younghoon Seo

5/5/2021

Overview

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, we will utilize 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data and Library Loading

```
library(ggplot2)
library(caret)

## Loading required package: lattice

library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(corrplot)

## corrplot 0.84 loaded

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

set.seed(12345)

train_url<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.
csv"
test_url<- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.c
sv"

if (!file.exists("./data")) {
        dir.create("./data")
}
if (!file.exists("./data/pml-training.csv")) {
        download.file(train_url, destfile="./data/pml-training.csv", method="
curl")
}
if (!file.exists("./data/pml-testing.csv")) {
        download.file(test_url, destfile="./data/pml-testing.csv", method="cu
rl")
}

traincsv <- read.csv("./data/pml-training.csv")
testcsv <- read.csv("./data/pml-testing.csv")
```

Cleaning and Processing Data

Missing values and metadata (first seven variables/columns) were eliminated from the data.

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

Afterward, near zero variance variables were also removed

```
nvz<-nearZeroVar(traincsv)
traincsv<-traincsv[,-nvz]
```

Upon eliminating unnecessary variables, traincsv data was split into training and validation data sets.

```
inTrain<-createDataPartition(y=traincsv$classe,p=.7,list=F)
train_data<-traincsv[inTrain,]
validation_data<-traincsv[-inTrain,]
```

The process of data cleaning was also applied to the test data set.

```
testcsv<-testcsv[,colSums(is.na(testcsv))==0]
testcsv <- testcsv[,-c(1:7)]
traincsv<-traincsv[,-nvz]
test_data<-testcsv
```

Prior to creating the models, a correlation matrix of variables in the training data set was created for visualizing the relationship between the variables.

```
cor_data<-cor(train_data[,-length(names(train_data))])
corrplot(cor_data,method="color")
```

Models

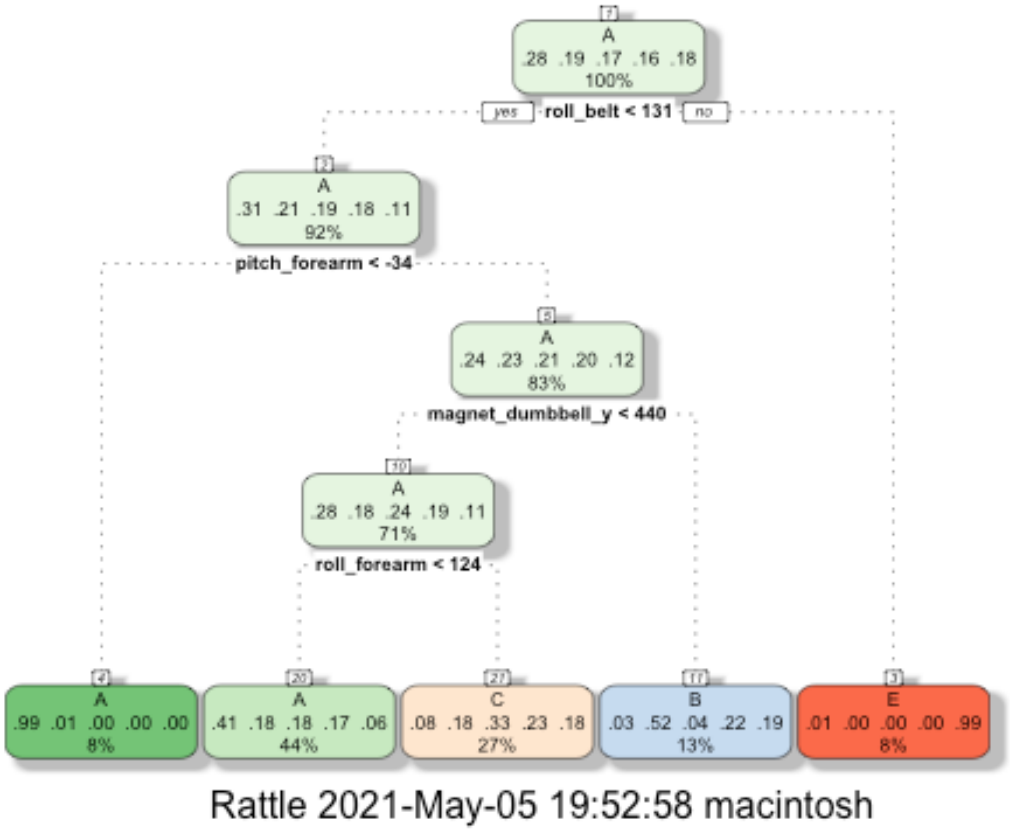
This investigation utilized the following models:

- Decision Tree
- Random Forest

- Gradient Boosted Trees

Decision Tree

```
model_tree<-train(classe~.,data=train_data,method="rpart",
                  trControl=trainControl(method="cv",number=4,verboseIter=F))
fancyRpartPlot(model_tree$finalModel)
```

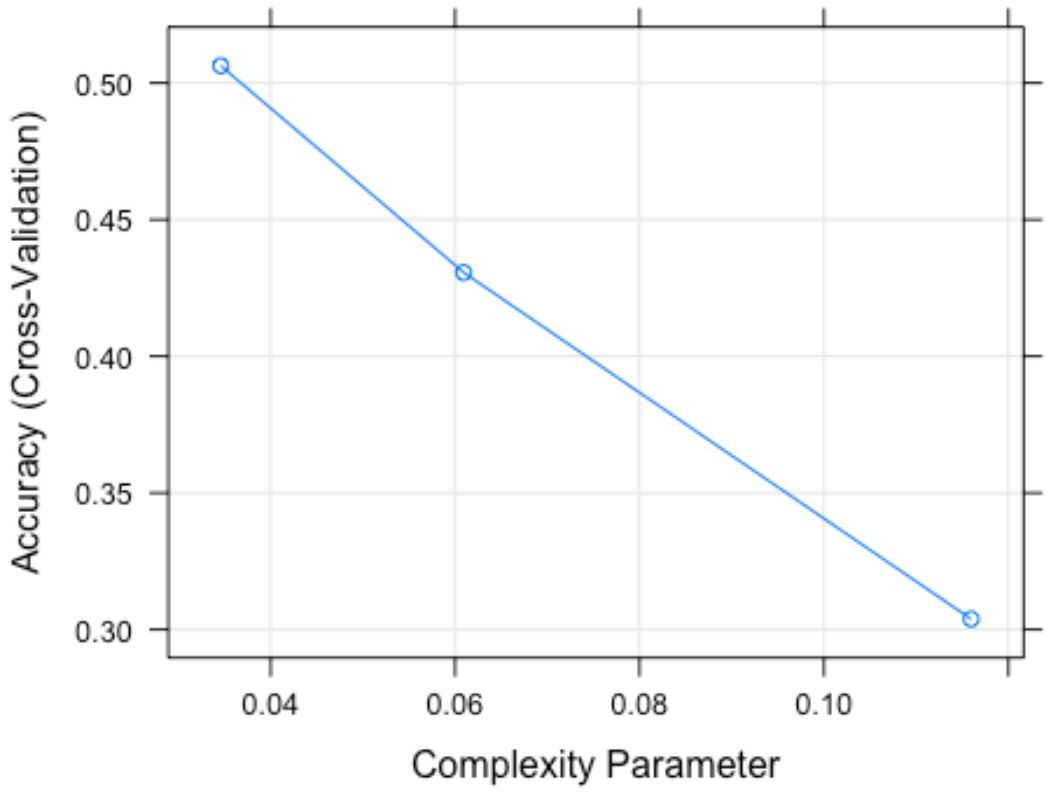



```
predict_tree<-predict(model_tree,validation_data)
confusionMatrix(predict_tree,factor(validation_data$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1525  484  499  423  153
##          B   29  385   37  187  159
##          C  116  270  490  354  289
##          D    0    0    0    0    0
##          E    4    0    0    0  481
##
## Overall Statistics
##
##                Accuracy : 0.4895
```

```
##                  95% CI : (0.4767, 0.5024)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3324
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9110  0.33802  0.47758   0.0000  0.44455
## Specificity            0.6298  0.91319  0.78823   1.0000  0.99917
## Pos Pred Value         0.4945  0.48306  0.32258      NaN  0.99175
## Neg Pred Value         0.9468  0.85181  0.87723   0.8362  0.88870
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2591  0.06542  0.08326   0.0000  0.08173
## Detection Prevalence   0.5240  0.13543  0.25811   0.0000  0.08241
## Balanced Accuracy      0.7704  0.62560  0.63291   0.5000  0.72186
```

```
plot(model_tree)
```



```
accuracy_tree<-postResample(predict_tree,factor(validation_data$classe))["Acc
uracy"]
oose_tree<-1-accuracy_tree
```

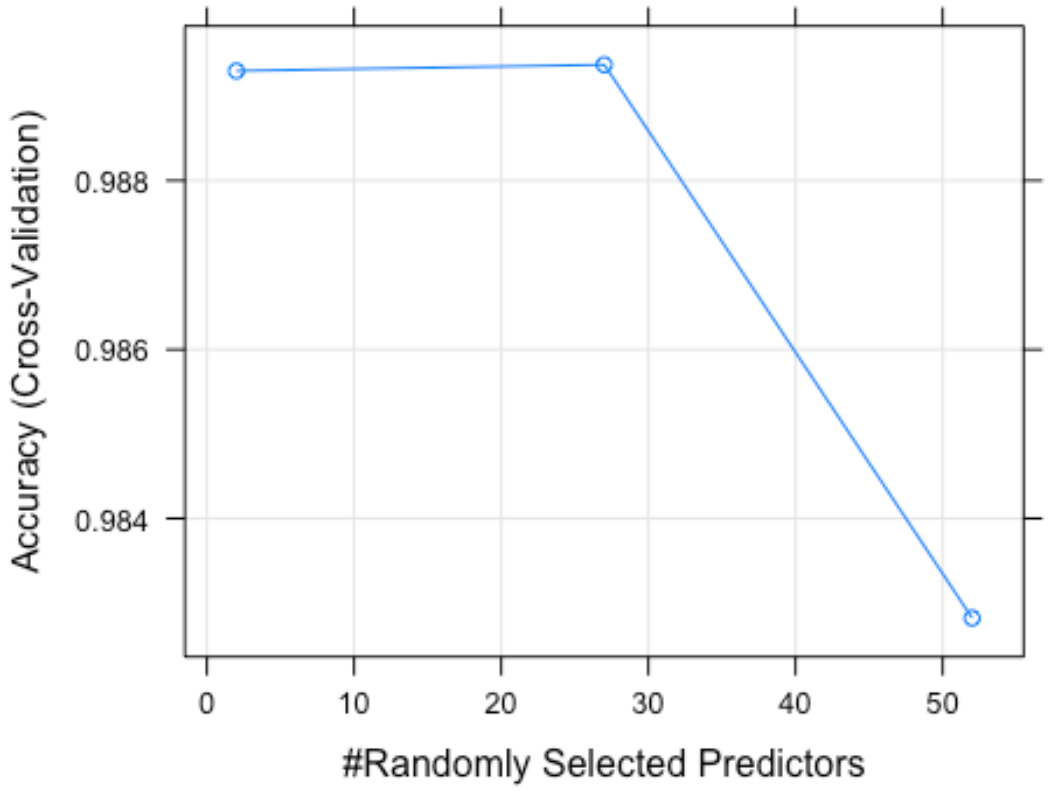
Random Forest

```
model_forest<-train(classe~.,data=train_data,method="rf",
                    trControl=trainControl(method="cv",number=4,verboseIter=F
))

predict_forest<-predict(model_forest,validation_data)
confusionMatrix(predict_forest,factor(validation_data$classe))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    6    0    0    0
##          B    1 1130    5    0    0
##          C    1    3 1018    7    2
##          D    0    0    3  956    1
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9921   0.9922   0.9917   0.9972
## Specificity            0.9986   0.9987   0.9973   0.9992   0.9998
## Pos Pred Value         0.9964   0.9947   0.9874   0.9958   0.9991
## Neg Pred Value         0.9995   0.9981   0.9984   0.9984   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1920   0.1730   0.1624   0.1833
## Detection Prevalence   0.2851   0.1930   0.1752   0.1631   0.1835
## Balanced Accuracy      0.9987   0.9954   0.9948   0.9954   0.9985

plot(model_forest)
```




```
accuracy_forest<-postResample(predict_forest,factor(validation_data$classe))[
"Accuracy"]
oose_forest<-1-accuracy_forest
```

Gradient Boosted Tree

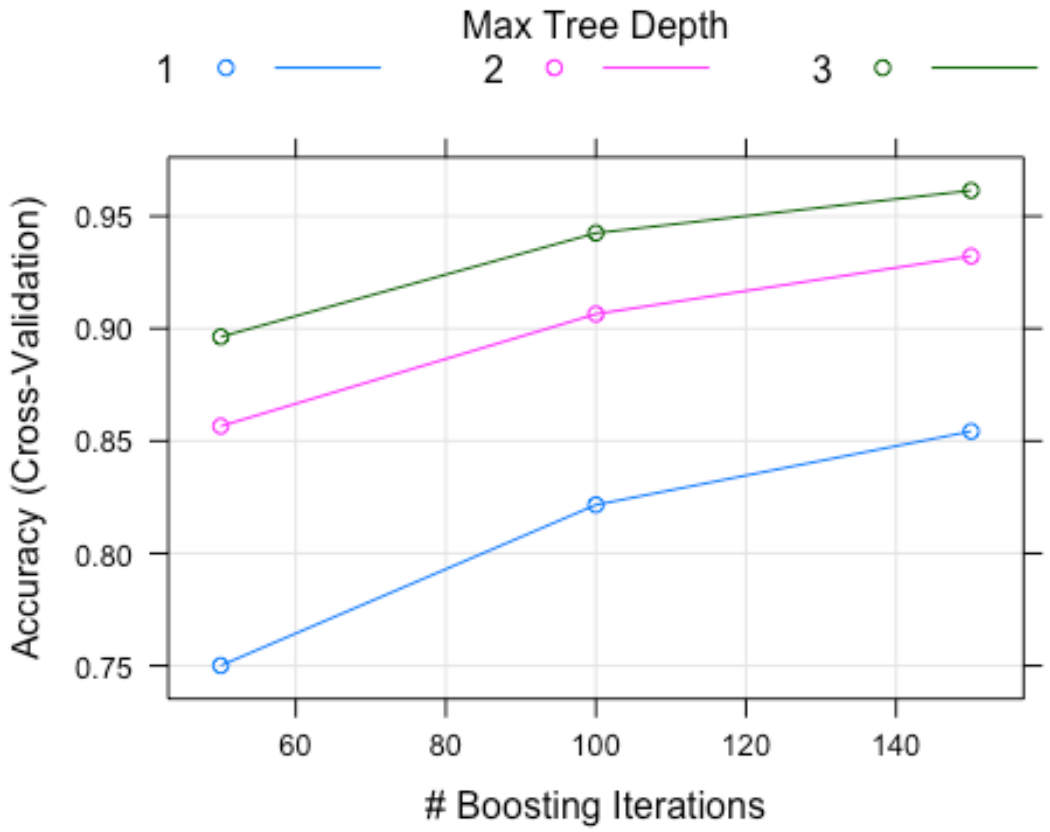
```
model_boosting<-train(classe~.,data=train_data,method="gbm",
                      trControl=trainControl(method="cv",number=4,verboseIter
=F),verbose=F)

predict_boosting<-predict(model_boosting,validation_data)
confusionMatrix(predict_boosting,factor(validation_data$classe))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1648   45    0    3    1
##          B   20 1060   35    2   15
##          C    3   31  978   35    8
##          D    3    1   12  920    8
##          E    0    2    1    4 1050
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9611
##                  95% CI : (0.9558, 0.9659)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9508
##
##  Mcnemar's Test P-Value : 7.007e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9845   0.9306   0.9532   0.9544   0.9704
## Specificity            0.9884   0.9848   0.9842   0.9951   0.9985
## Pos Pred Value         0.9711   0.9364   0.9270   0.9746   0.9934
## Neg Pred Value         0.9938   0.9834   0.9901   0.9911   0.9934
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2800   0.1801   0.1662   0.1563   0.1784
## Detection Prevalence   0.2884   0.1924   0.1793   0.1604   0.1796
## Balanced Accuracy      0.9864   0.9577   0.9687   0.9747   0.9845
```

```
plot(model_boosting)
```



```
accuracy_boosting<-postResample(predict_boosting,factor(validation_data$class
e))["Accuracy"]
oose_boosting<-1-accuracy_boosting
```

Error Comparison

```
error_tree<-data.frame(accuracy_tree,oose_tree)
error_forest<-data.frame(accuracy_forest,oose_forest)
error_boosting<-data.frame(accuracy_boosting,oose_boosting)
error_table<-data.frame(Accuracy=c(error_tree[[1]],error_forest[[1]],error_bo
osting[[1]]),
           Out_Of_Sample_Error=c(error_tree[[2]],error_forest[[2]],error_boos
ting[[2]]))
rownames(error_table)<-c("Decision Tree","Random Forest","Gradient Boosted Tr
ee")
error_table
```

```
##                          Accuracy Out_Of_Sample_Error
## Decision Tree          0.4895497         0.510450297
## Random Forest          0.9949023         0.005097706
## Gradient Boosted Tree 0.9610875         0.038912489
```

The table clearly delineated that the best model was Random Forest model, given the 0.9896 accuracy and 0.0104 out of sample error rate.

Prediction

The Random Forest model was used to predict the classe outcome for 20 cases in the test data set.

```
prediction<-predict(model_forest,test_data)
prediction
```

```
##  [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
```