Practical Machine Learning Week 4 Assignment

Lee Saxton

10/03/2021

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).

Introduction

The goal of this assignment is to us the accelerometer data from the belt, forearm, arm and dumbbell of 6 participants to identify the class of exercise being performed. This is described as:

Class A = perform barbell lift correctly
Class B = throwing elbows to the front
Class C = lifting dumbell only half way
Class D = lowering the dumbbell only half way
Class E = throwing the hips to the front

Data Loading and Cleaning

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

Download the datasets:

```
setwd("/users/lee_saxton/Documents/Data Science Specialisation/08 Practical Machine Lea
rning/Week 4 Assignment/Week 4 Assignment - R Studio/")
if (!file.exists("data")) {
  dir.create("data")
}
TrainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
TestUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(TrainUrl, destfile="./data/pml-training.csv", method="curl")
download.file(TestUrl, destfile="./data/pml-testing.csv", method="curl")
training <- read.csv("./data/pml-training.csv", na.strings=c("NA", ""))
testing <- read.csv("./data/pml-testing.csv", na.strings=c("NA",""))
```

Now remove the NA and blank entries from the data, essentially remove columns where the number of NA values in the column is not less than 95% of the number of rows in the dataset

```
training_noNA <- training[,colSums(is.na(training))<0.95*nrow(training)]
testing_noNA <- testing[,colSums(is.na(testing))<nrow(testing)]
```

Now remove the unnecessary columns (1-7) that don’t include measurement data

```
training_noNA <- training_noNA[,-(1:7)]
testing_noNA <- testing_noNA[,-(1:7)]
```

Validation

We will use the training_noNA dataset to build and validate models. The final testing will be made using the testing_noNA data. So we need to split the training_noNA data into 2 groups, one for modeling and one for testing.

```
library(caret)
set.seed(12345)
InTest <- createDataPartition(y=training_noNA$classe, p=0.7, list=FALSE)
subtraining <- training_noNA[InTest,]
subtesting <- training_noNA[-InTest,]
dim(subtesting)
```

```
## [1] 5885   53
```

```
dim(subtraining)
```

```
## [1] 13737    53
```

Modeling

I will create models using rf and gbm methods (random forest and generalised boosting). The resulting models will be tested using the subtesting data and I will then select the most accurate model for the final test on the testing_noNA data.

Random Forest Model

Run random forest model. Use of training control required to limit runtime (tip found online).

```
set.seed(12345)
library(caret)
controlrf <- trainControl(method="cv", number=3)
rffit <- train(classe ~ ., data=subtraining, method="rf", trControl=controlrf)
rffit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    5    0    0    2 0.001792115
## B   19 2630    9    0    0 0.010534236
## C    0   15 2373    8    0 0.009599332
## D    0    1   21 2227    3 0.011101243
## E    0    3    4    3 2515 0.003960396
```

Now run prediction test to obtain confusion matrix and an estimate of the model accuracy.

```
rfpredict <- predict(rffit, newdata=subtesting)
rfcm <- confusionMatrix(rfpredict, factor(subtesting$classe))
rfcm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    7    0    0    0
##          B    1 1129    4    0    0
##          C    1    3 1019    7    1
##          D    0    0    3  956    1
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9929, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9912   0.9932   0.9917   0.9982
## Specificity            0.9983   0.9989   0.9975   0.9992   0.9998
## Pos Pred Value         0.9958   0.9956   0.9884   0.9958   0.9991
## Neg Pred Value         0.9995   0.9979   0.9986   0.9984   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1918   0.1732   0.1624   0.1835
## Detection Prevalence   0.2853   0.1927   0.1752   0.1631   0.1837
## Balanced Accuracy      0.9986   0.9951   0.9954   0.9954   0.9990
```

Generalised Boosting Model

Run the gbm model. Again, use of trainControl applied to limit runtime.

```
set.seed(12345)
library(caret)
controlgb <- trainControl(method="repeatedcv", number=5, repeats=1)
gbfit <- train(classe ~ ., data=subtraining, method="gbm", trControl=controlgb, verbose
=FALSE)
gbfit$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 51 had non-zero influence.
```

Now run prediction test to obtain confusion matrix and an estimate of the model accuracy.

```
gbpredict <- predict(gbfit, newdata=subtesting)
gbcm <- confusionMatrix(gbpredict, factor(subtesting$classe))
gbcm
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   38    0    1    1
##          B   21 1066   37    2   15
##          C    1   33  981   35    7
##          D    3    2    7  918    7
##          E    0    0    1    8 1052
## 
## Overall Statistics
## 
##                Accuracy : 0.9628
##                  95% CI : (0.9576, 0.9675)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9529
## 
##  Mcnemar's Test P-Value : 1.234e-06
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9851   0.9359   0.9561   0.9523   0.9723
## Specificity            0.9905   0.9842   0.9844   0.9961   0.9981
## Pos Pred Value         0.9763   0.9343   0.9281   0.9797   0.9915
## Neg Pred Value         0.9940   0.9846   0.9907   0.9907   0.9938
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2802   0.1811   0.1667   0.1560   0.1788
## Detection Prevalence   0.2870   0.1939   0.1796   0.1592   0.1803
## Balanced Accuracy      0.9878   0.9601   0.9702   0.9742   0.9852
```

Results

The random forest model gave an accuracy of 99.51%. The generalized boosting model gave an accuracy of 96.28%. I will therefore use the random forest model on the validation data.

Validation

I will now apply the random forest predictor to the validation data. The resulting predictions will be entered in to the course prediction quiz.

```
finalprediction <- predict(rffit, testing_noNA)
finalprediction
```

```
##  [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
```