Practical Machine Learning - Course Project

Melissa Sacevich

7/13/2020

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

Data Loading and Processing

Load the R libraries required for analysis.

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(corrplot)
```

Load the test and training dataset from the provided URL.

```
TrainData <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
TestData  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
train_data <- read.csv(url(TrainData))
test_data <- read.csv(url(TestData))
dim(train_data)
```

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

```
dim(test_data)
```

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

Data Cleaning

Clean the data to get rid of observations with missing values and columns that do not contribute much to the accelerometer measurements.

```
#remove variables with missing values
train_data<- train_data[, colSums(is.na(train_data)) == 0]
test_data <- test_data[, colSums(is.na(test_data)) == 0]
dim(train_data)
```

```
## [1] 19622    93
```

```
dim(test_data)
```

```
## [1] 20 60
```

```
#remove identification variables (columns 1-7)
train_data <- train_data[, -c(1:7)]
test_data <- test_data[, -c(1:7)]
dim(train_data)
```

```
## [1] 19622    86
```

```
dim(test_data)
```

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

Data Partitioning

Partition the Training dataset in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations.

```
set.seed(1234)
training <- createDataPartition(train_data$classe, p = 0.7, list = FALSE)
train_data1 <- train_data[training, ]
train_data2 <- train_data[-training, ]
dim(train_data1)
```

```
## [1] 13737    86
```

```
dim(train_data2)
```

```
## [1] 5885   86
```

```
#remove variables with near zero variance
NZV <- nearZeroVar(train_data1)
train_data1 <- train_data1[, -NZV]
train_data2  <- train_data2[, -NZV]
dim(train_data1)
```

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

```
dim(train_data2)
```

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

Data Modeling

Here we will test three methods to model the regressions: Random Forest, Decision Tree and Generalized Boosted Model.

Random Forest

```
#fitting the model with train data
set.seed(12345)
RandForest <- trainControl(method="cv", number=3, verboseIter=FALSE)
trainRF <- train(classe ~ ., data=train_data1, method="rf", trControl=RandForest)
trainRF$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: 2
## 
##         OOB estimate of  error rate: 0.68%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    3    0    0    0 0.0007680492
## B   14 2636    8    0    0 0.0082768999
## C    0   19 2375    2    0 0.0087646077
## D    0    0   40 2210    2 0.0186500888
## E    0    0    1    5 2519 0.0023762376
```

```
#prediction using partitioned test data
predictRF <- predict(trainRF, newdata=train_data2)
RF <- confusionMatrix(predictRF, train_data2$classe)
RF
```

```
## Confusion Matrix and Statistics
## 
##           Reference
```

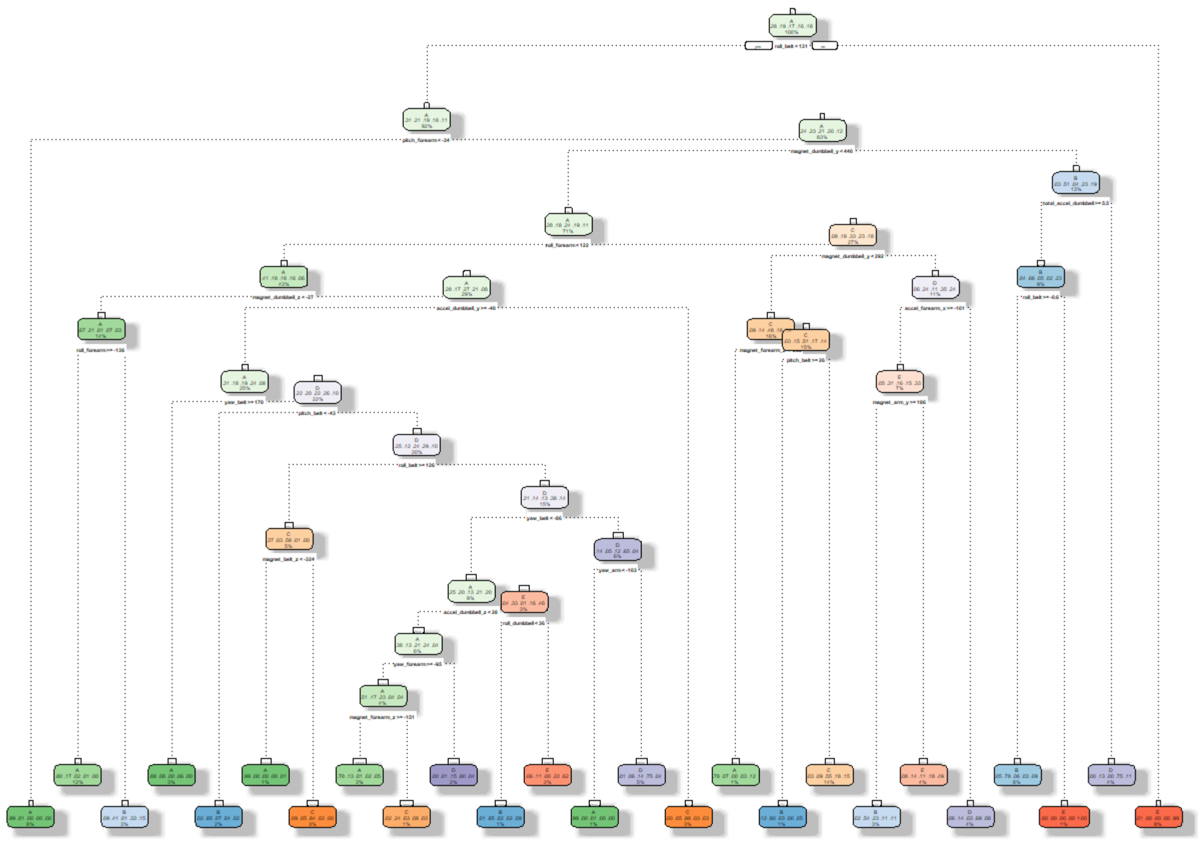
```
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1130   15    0    0
##          C    0    6 1010   14    0
##          D    0    0    1  949    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9844   0.9844   0.9991
## Specificity            0.9993   0.9968   0.9959   0.9996   0.9998
## Pos Pred Value         0.9982   0.9869   0.9806   0.9979   0.9991
## Neg Pred Value         1.0000   0.9981   0.9967   0.9970   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1920   0.1716   0.1613   0.1837
## Detection Prevalence   0.2850   0.1946   0.1750   0.1616   0.1839
## Balanced Accuracy      0.9996   0.9945   0.9901   0.9920   0.9994
```

The random forest accuracy is 0.993.

Decision Trees

```
#fitting the model with train data
set.seed(12345)
DecisionTree <- rpart(classe ~ ., data=train_data1, method="class")
fancyRpartPlot(DecisionTree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2020-Jul-15 19:45:18 msacevich

```
#prediction using partitioned test data
predictDT <- predict(DecisionTree, train_data2, type = "class")
CM <- confusionMatrix(predictDT, train_data2$classe)
CM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1522  167   12   49   13
##          B   58  706  100   79   96
##          C   47  109  819  148  139
##          D   25   94   67  609   52
##          E   22   63   28   79  782
##
## Overall Statistics
##
##                Accuracy : 0.7541
##                  95% CI : (0.7429, 0.7651)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6885
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9092   0.6198   0.7982   0.6317   0.7227
## Specificity            0.9428   0.9298   0.9088   0.9516   0.9600
## Pos Pred Value         0.8633   0.6795   0.6490   0.7190   0.8029
## Neg Pred Value         0.9631   0.9106   0.9552   0.9295   0.9389
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2586   0.1200   0.1392   0.1035   0.1329
## Detection Prevalence   0.2996   0.1766   0.2144   0.1439   0.1655
## Balanced Accuracy      0.9260   0.7748   0.8535   0.7917   0.8414
```

The decision tree accuracy is 0.7541.

Apply Selected Model to Test Data

The accuracy for the Random Forest model was greater than the Decision Tree Model. Therefore, the Random Forest model will be applied to predict the 20 quiz results using the testing dataset.

```
results <- predict(trainRF, newdata=test_data)
results
```

```
##  [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
```