Practical Machine Learning Prediction Assignment

Kent Lanclos

June 29, 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://web.archive.org/web/20161224072740/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

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

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

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## 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)
```

```
## Warning: package 'knitr' was built under R version 3.4.4
```

Load the training and test datasets

Upon initially loading the two datasets, it was apparent there were missing values, division by zero and other data integrity issues. The "na.strings" function was used to identify and resolve those issues.

```
set.seed(1111)

Train_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Test_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

Train_data <- read.csv(url(Train_Url), na.strings=c("NA","#DIV/0!",""))
Test_data <- read.csv(url(Test_Url), na.strings=c("NA","#DIV/0!",""))
```

Partition the data

The training dataset was partitioned into training and test subsets, with 60% of the observations allocated to the training subset and the remaining 40% to the test subset.

```
Train_part <- createDataPartition(Train_data$classe, p=0.6, list=FALSE)
Train_set <- Train_data[Train_part, ]
Test_set <- Train_data[-Train_part, ]
dim(Train_set)
```

```
## [1] 11776   160
```

```
dim(Test_set)
```

```
## [1] 7846  160
```

Clean data subsets

The train and test subsets were cleaned by removing variables with minimal variance. The "nearZeroVar" function was used to identify such variables which were stored in the "nzv_t..." dataset which was used to

remove the corresponding variables from the orginal train and test subsets.

```
nzv_Train <- nearZeroVar(Train_set, saveMetrics=TRUE)
Train_set <- Train_set[,nzv_Train$nzv==FALSE]
nzv_Test <- nearZeroVar(Test_set,saveMetrics=TRUE)
Test_set <- Test_set[,nzv_Test$nzv==FALSE]
```

Remove ID variable

ID in first column removed as of no use in ML algos

```
Train_set <- Train_set[c(-1)]
```

Clean variables

Variables with more than 70% NAs are removed

```
Train_tmp <- Train_set
for(i in 1:length(Train_set)) {
    if( sum( is.na( Train_set[, i] ) ) /nrow(Train_set) >= .7) {
        for(j in 1:length(Train_tmp)) {
            if( length( grep(names(Train_set[i]), names(Train_tmp)[j]) ) == 1)  {
                Train_tmp <- Train_tmp[ , -j]
            }
        }
    }
}
Train_set <- Train_tmp
rm(Train_tmp)
```

```
allow1 <- colnames(Train_set)
allow2 <- colnames(Train_set[, -58])
Test_set <- Test_set[allow1]
Test_data <- Test_data[allow2]
dim(Test_set)
```

```
## [1] 7846   58
```

```
dim(Test_data)
```

```
## [1] 20 57
```

Coerce test dataset into same type as train dataset to ensure decision tree and random forest models work appropriately

```
for (i in 1:length(Test_data) ) {
    for(j in 1:length(Train_set)) {
        if( length( grep(names(Train_set[i]), names(Test_data)[j]) ) == 1)  {
            class(Test_data[j]) <- class(Train_set[i])
        }
    }
}
Test_data <- rbind(Train_set[2, -58] , Test_data)
Test_data <- Test_data[-1,]
```

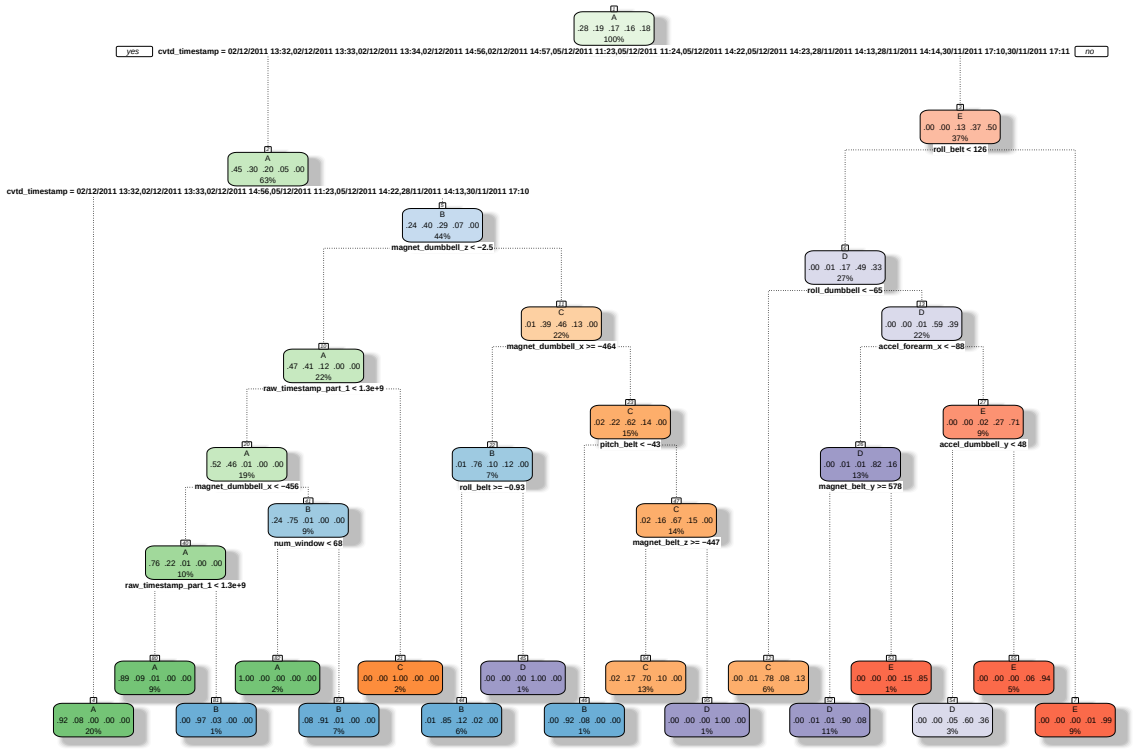
Approaches

Two ML modelling approaches will be evaluated: (1) Decision Tree, and (2) Random Forest. All predictor variables will be included in developing the models. Prediction accuracy will be the metric used to identify which method performs better.

Prediction with Decision Tree

The Decision Tree approach generates an overall accuracy of not quite 87%. The 95% confident interval is fairly tight and ranges from .8582 to .8734.

```
set.seed(1111)
model_dt <- rpart(classe ~ ., data=Train_set, method="class")
fancyRpartPlot(model_dt)
```



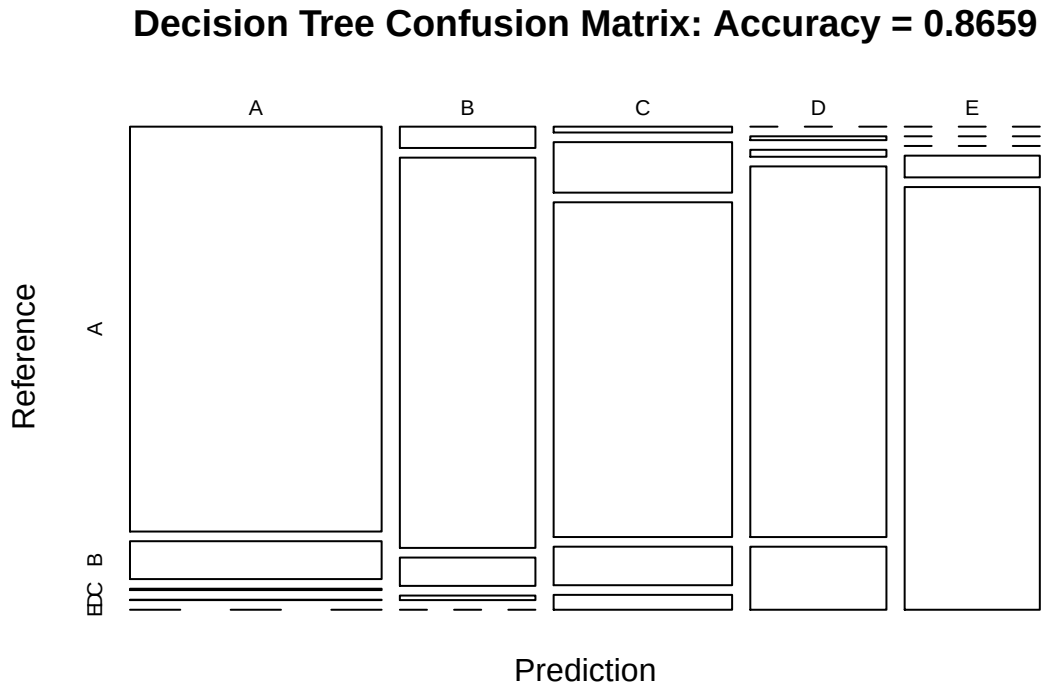
Rattle 2018−Jul−06 04:32:39 kentl

```
predict_dt <- predict(model_dt, Test_set, type = "class")
confuse_dt <- confusionMatrix(predict_dt, Test_set$classe)
confuse_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2149  201    7    2    0
##          B   61 1116   81   13    0
##          C   22  190 1260  145   56
```

```
##          D    0   11   20 1064  181
##          E    0    0    0   62 1205
##
## Overall Statistics
##
##                Accuracy : 0.8659
##                  95% CI : (0.8582, 0.8734)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8302
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9628   0.7352   0.9211   0.8274   0.8356
## Specificity            0.9626   0.9755   0.9362   0.9677   0.9903
## Pos Pred Value         0.9110   0.8780   0.7531   0.8339   0.9511
## Neg Pred Value         0.9849   0.9389   0.9825   0.9662   0.9640
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2739   0.1422   0.1606   0.1356   0.1536
## Detection Prevalence   0.3007   0.1620   0.2132   0.1626   0.1615
## Balanced Accuracy      0.9627   0.8553   0.9286   0.8975   0.9130
```

```
plot(confuse_dt$table, col = confuse_dt$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy
```



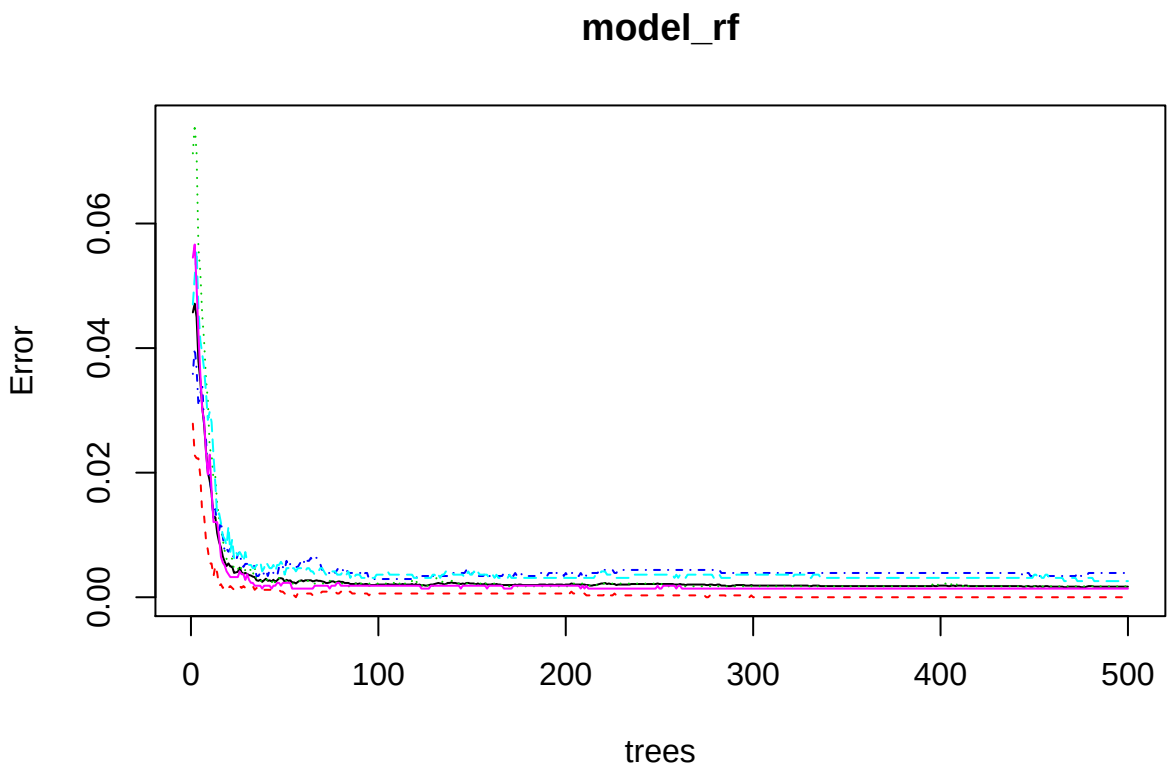
Prediction with Random Forests

The Random Forest approach generates an overall accuracy of over 99%. Further, the 95% confidence ranges from only .9977 to .9994, so we can be confident that the calculated accuracy is a fair assessment of the model's predictive ability.

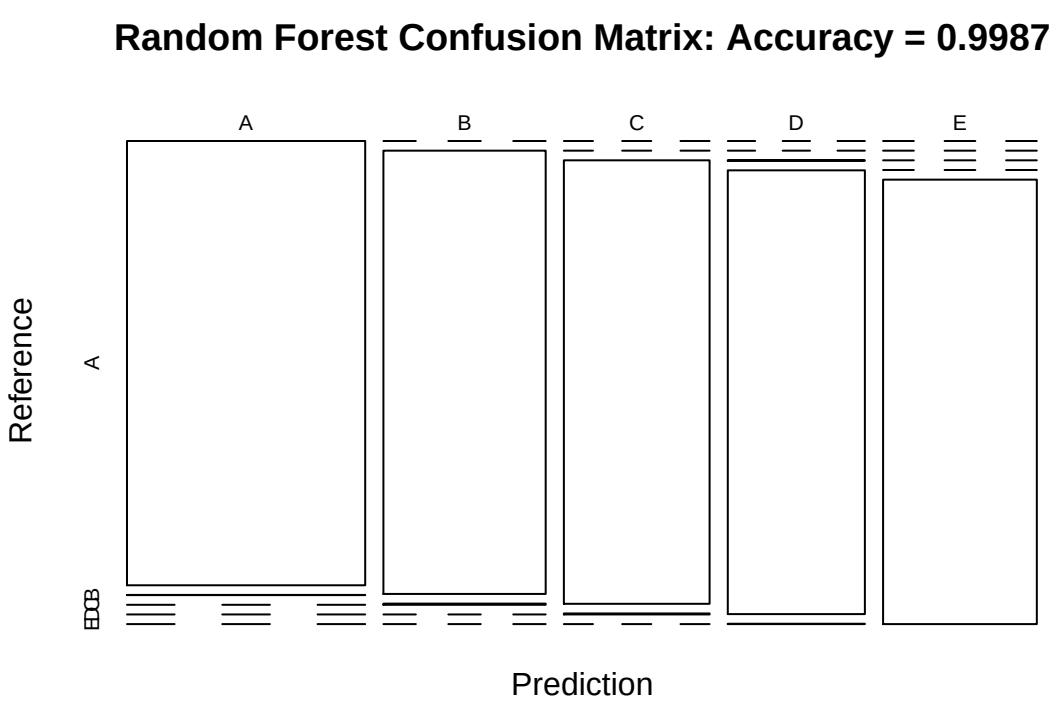
```
set.seed(1111)
model_rf <- randomForest(classe ~ ., data=Train_set)
predict_rf <- predict(model_rf, Test_set, type = "class")
confuse_rf <- confusionMatrix(predict_rf, Test_set$classe)
confuse_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    4    0    0
##          C    0    0 1363    3    0
##          D    0    0    1 1283    1
##          E    0    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.9987
##                  95% CI : (0.9977, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9984
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9963   0.9977   0.9993
## Specificity            0.9998   0.9994   0.9995   0.9997   1.0000
## Pos Pred Value         0.9996   0.9974   0.9978   0.9984   1.0000
## Neg Pred Value         1.0000   0.9998   0.9992   0.9995   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1737   0.1635   0.1837
## Detection Prevalence   0.2846   0.1939   0.1741   0.1638   0.1837
## Balanced Accuracy      0.9999   0.9994   0.9979   0.9987   0.9997
```

```
plot(model_rf)
```




```
plot(confuse_rf$table, col = confuse_dt$byClass, main = paste("Random Forest Confusion Matrix: Accuracy
```



Predicting Results on the Test Data

The prediction accuracy of Random Forest exceeds 99%, far superior to the prediction accuracy of the Decision Tree approach. As such, the Random Forest approach is preferred to predict the manner in which the participants exercised.

Calculated out-of-sample error

The expected out-of-sample error is 100-99.87 = 0.13%.

Generating Answers for Quiz

The Random Forest model is used to generate the answers for the 20 questions of the subsequent quiz. The answer to each of the questions is provided below.

```
prediction_rf <- predict(model_rf, Test_data, type = "class")
prediction_rf
```

```
##  1  2  3 41  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
```