Practical Machine Learning - Course Project

Pieter van der Want

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.

Accelerometers were placed 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).

My goal is to correctly predict the way in which the excersize was performed. Exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E)

Load Data and Libraries

```
### load libraries and set seed
library(caret)
```

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

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

```
library(rpart)
set.seed(5252)

### download files
train_file <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_file <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if (!file.exists("pml-training.csv")) {
      download.file(train_file)
}
if (!file.exists("pml-testing.csv")) {
      download.file(test_file)
}

### load data
training <- read.csv("pml-training.csv", na.strings=c("NA","","#DIV/0!"))
testing <- read.csv("pml-testing.csv", na.strings=c("NA","","#DIV/0!"))
training$classe <- as.factor(training$classe)

### split training set
part <- createDataPartition(training$classe, p=0.7, list = FALSE)
```

```
int_train <- training[part, ]
int_test <- training[-part, ]
```

cleaning the data

When looking at the data we see a lot of NA's as well as NZV's(near zero variance).

```
### Remove variables with near zero variance
NZV <- nearZeroVar(int_train)
int_train <- int_train[, -NZV]
int_test <- int_test[, -NZV]

### Remove variables that are mostly NA
na_train <- sapply(int_train, function(x) mean(is.na(x))) > 0.95
int_train <- int_train[, na_train == FALSE]
int_test <- int_test[, na_train == FALSE]

### Remove identification variables
int_train <- int_train[, -(1:5)]
int_test <- int_test[, -(1:5)]
```

With the cleaning process above we reduced the variables from 160 to 54.

Prediction model building

Three methods will be applied to model the regressions (in the int_train dataset) and the best one (with higher accuracy when applied to the int_test dataset) will be used for the quiz predictions. The methods are: Random Forests, Decision Tree and Generalized Boosted Model, as described below. A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

1. Random forrest

Generate model fit

```
modfit_rf <- train(classe ~ ., method = "rf", data = int_train, trControl = trainControl(method = 'cv')
```

Predict on the test set

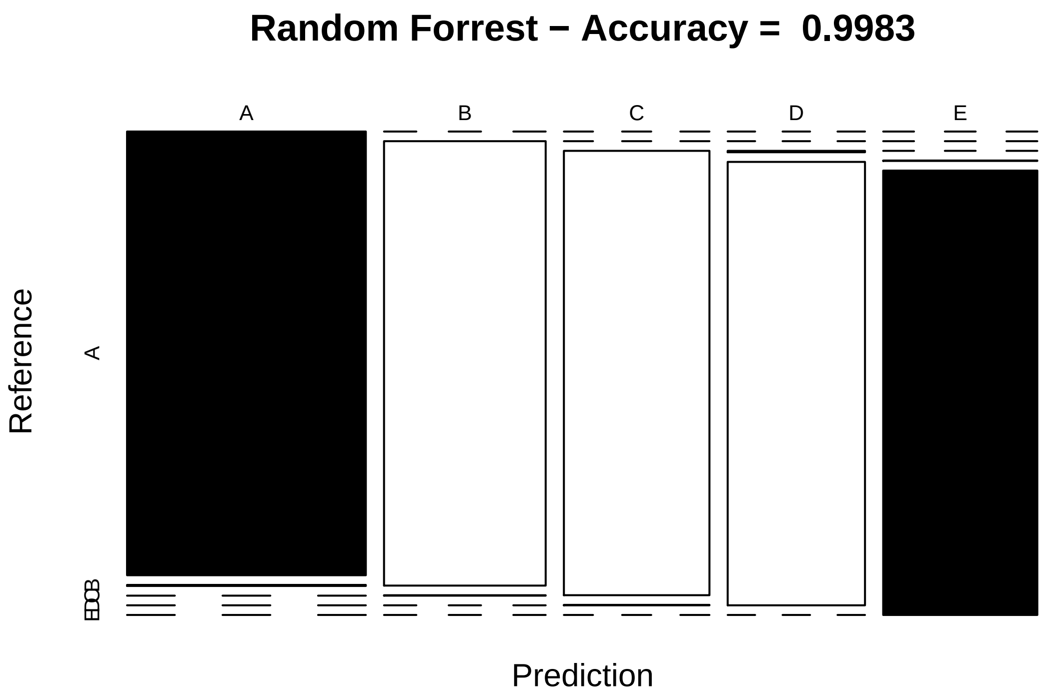
```
predict_rf <- predict(modfit_rf, newdata = int_test)
conf_rf <- confusionMatrix(predict_rf, int_test$classe)
conf_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1135    1    0    0
##          C    0    0 1022    1    0
##          D    0    0    3  962    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9969, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9961   0.9979   1.0000
## Specificity            0.9991   0.9998   0.9998   0.9994   0.9998
## Pos Pred Value         0.9976   0.9991   0.9990   0.9969   0.9991
## Neg Pred Value         1.0000   0.9992   0.9992   0.9996   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1737   0.1635   0.1839
## Detection Prevalence   0.2851   0.1930   0.1738   0.1640   0.1840
## Balanced Accuracy      0.9995   0.9981   0.9979   0.9987   0.9999
```

Plot results

```
plot(conf_rf$table, col = conf_rf$byClass, main = paste("Random Forrest - Accuracy = ", round(conf_rf$o
```



2. Descision tree Generate model fit

```
modfit_dt <- rpart(classe ~ ., data = int_train, method = "class")
```

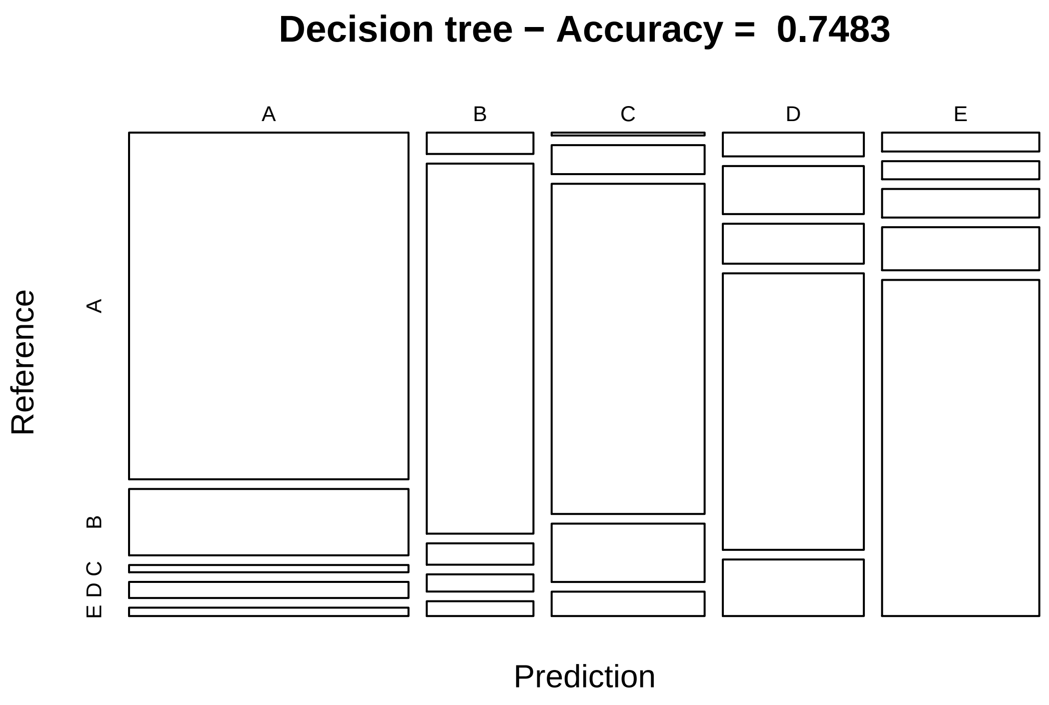
Predict on the test set

```r
predict_dt <- predict(modfit_dt, newdata = int_test, type = "class")
conf_dt <- confusionMatrix(predict_dt, int_test$classe)
conf_dt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1531  293   32   71   37
##          B   36  624   36   29   25
##          C    7   70  798  141   59
##          D   53  107   89  616  126
##          E   47   45   71  107  835
##
## Overall Statistics
##
##                Accuracy : 0.7483
##                  95% CI : (0.737, 0.7594)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.68
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9146   0.5478   0.7778   0.6390   0.7717
## Specificity            0.8972   0.9735   0.9430   0.9238   0.9438
## Pos Pred Value         0.7795   0.8320   0.7423   0.6216   0.7557
## Neg Pred Value         0.9635   0.8997   0.9526   0.9289   0.9483
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2602   0.1060   0.1356   0.1047   0.1419
## Detection Prevalence   0.3337   0.1274   0.1827   0.1684   0.1878
## Balanced Accuracy      0.9059   0.7607   0.8604   0.7814   0.8578
```

Plot results

```r
plot(conf_dt$table, col = conf_dt$byClass, main = paste("Decision tree - Accuracy = ", round(conf_dt$ove
```



3. Generalized boosted model Generate model fit

```
modfit_gbm <- train(classe ~ ., method = "gbm", data = int_train, trControl = trainControl(method = "re
```

Predict on the test set

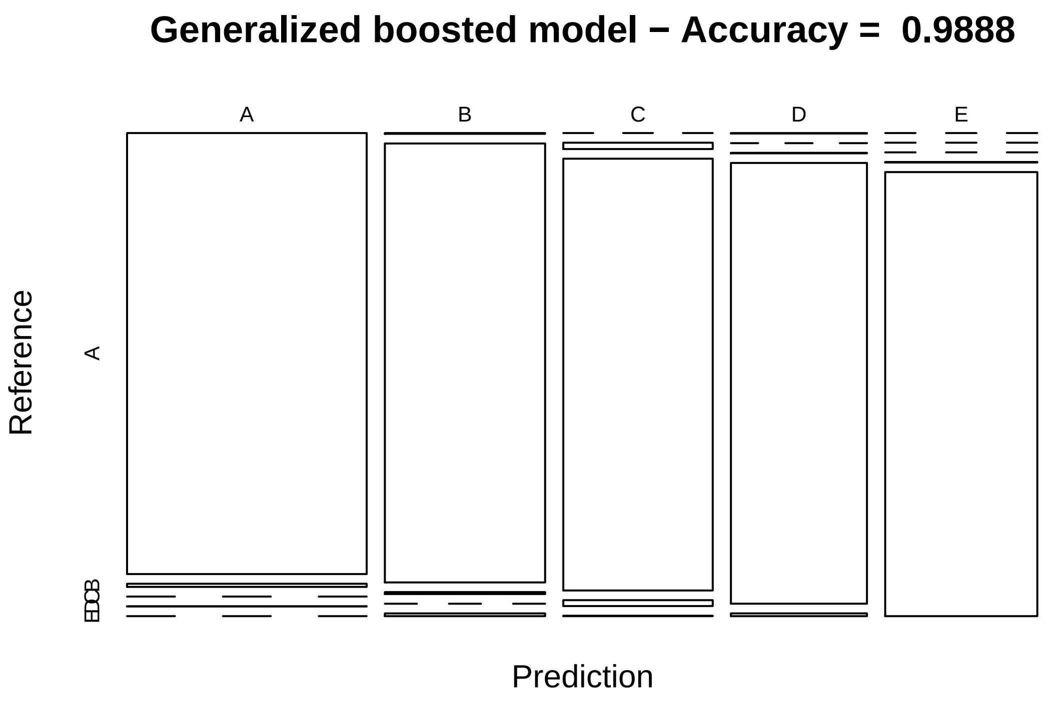
```
predict_gbm <- predict(modfit_gbm, newdata = int_test)
conf_gbm <- confusionMatrix(predict_gbm, int_test$classe)
conf_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671   12    0    1    0
##          B    2 1112    5    0    7
##          C    0   15 1020   14    1
##          D    1    0    1  948    6
##          E    0    0    0    1 1068
##
## Overall Statistics
##
##                Accuracy : 0.9888
##                  95% CI : (0.9858, 0.9913)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.9858
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9763   0.9942   0.9834   0.9871
## Specificity            0.9969   0.9971   0.9938   0.9984   0.9998
## Pos Pred Value         0.9923   0.9876   0.9714   0.9916   0.9991
## Neg Pred Value         0.9993   0.9943   0.9988   0.9968   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1890   0.1733   0.1611   0.1815
## Detection Prevalence   0.2862   0.1913   0.1784   0.1624   0.1816
## Balanced Accuracy      0.9976   0.9867   0.9940   0.9909   0.9934
```

Plot results

```
plot(conf_gbm$table, col = conf_gbm$byClass, main = paste("Generalized boosted model - Accuracy = ", rou
```



Select and apply model to test data

The accuracy of the three model are
1. Random forest : 0.9985
2. Decision tree : 0.7483
3. Generalized boosted model : 0.989

Because of the high accuracy of the Random forrest model, it will be applied to the test data.

```r
pred_test <- predict(modfit_rf, newdata = testing)
pred_test
```

```
##  [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
```