Practical Machine Learning - Course Project

Jayasree kulothungan

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.

Importing libraries

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
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:ggplot2':
##
##     margin
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

Reading the data

```
train <- read.csv("pml-training.csv")
test<- read.csv("pml-testing.csv")
dim(train)
```

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

```
dim(test)
```

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

Cleaning data

Remove all the data with missing values

```
sum(complete.cases(train))
```

```
## [1] 406
```

```
sum(complete.cases(test))
```

```
## [1] 0
```

```
trainData<- train[, colSums(is.na(train)) == 0]
testData <- test[, colSums(is.na(test)) == 0]
dim(trainData)
```

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

```
dim(testData)
```

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

Remove variables with less impact to the outcome

```
trainData <- trainData[, -c(1:7)]
testData <- testData[, -c(1:7)]
dim(trainData)
```

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

```
dim(testData)
```

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

removing variables with near zero variance

```r
NZV <- nearZeroVar(trainData)
trainData <- trainData[, -NZV]
testData  <- testData[, -NZV]
dim(trainData)
```

```
## [1] 19622    53
```

```r
dim(testData)
```

```
## [1] 20 29
```

Prepare the data for prediction

split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation data set to conduct cross validation in future steps.

```r
set.seed(1234)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData <- trainData[inTrain, ]
testData1 <- trainData[-inTrain, ]
dim(trainData)
```

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

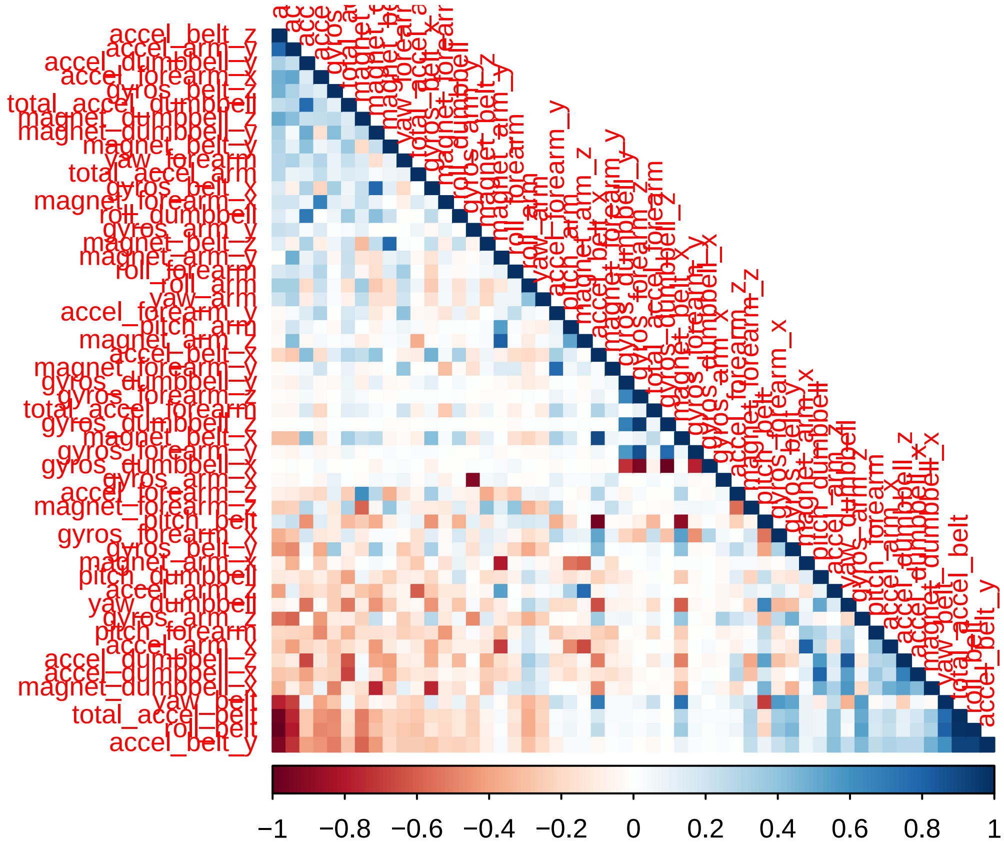
```r
dim(testData1)
```

```
## [1] 4123   53
```

Correlation Matrix Visualization

The following correlation plot uses the following parameters (source:CRAN Package 'corrplot') "FPC": the first principal component order. "AOE": the angular order tl.cex Numeric, for the size of text label (variable names) tl.col The color of text label.

```r
cor_mat <- cor(trainData[, -53])
corrplot(cor_mat, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8)
```



Data Modelling

For this project we will use two different algorithms - classification trees - random forests

Random Forest

fit a predictive model for activity recognition using **Random Forest** algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use **5-fold cross validation** when applying the algorithm.

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
modRF1 <- train(classe ~ ., data=trainData, method="rf", trControl=controlRF)
modRF1$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: 0.71%
## Confusion matrix:
```

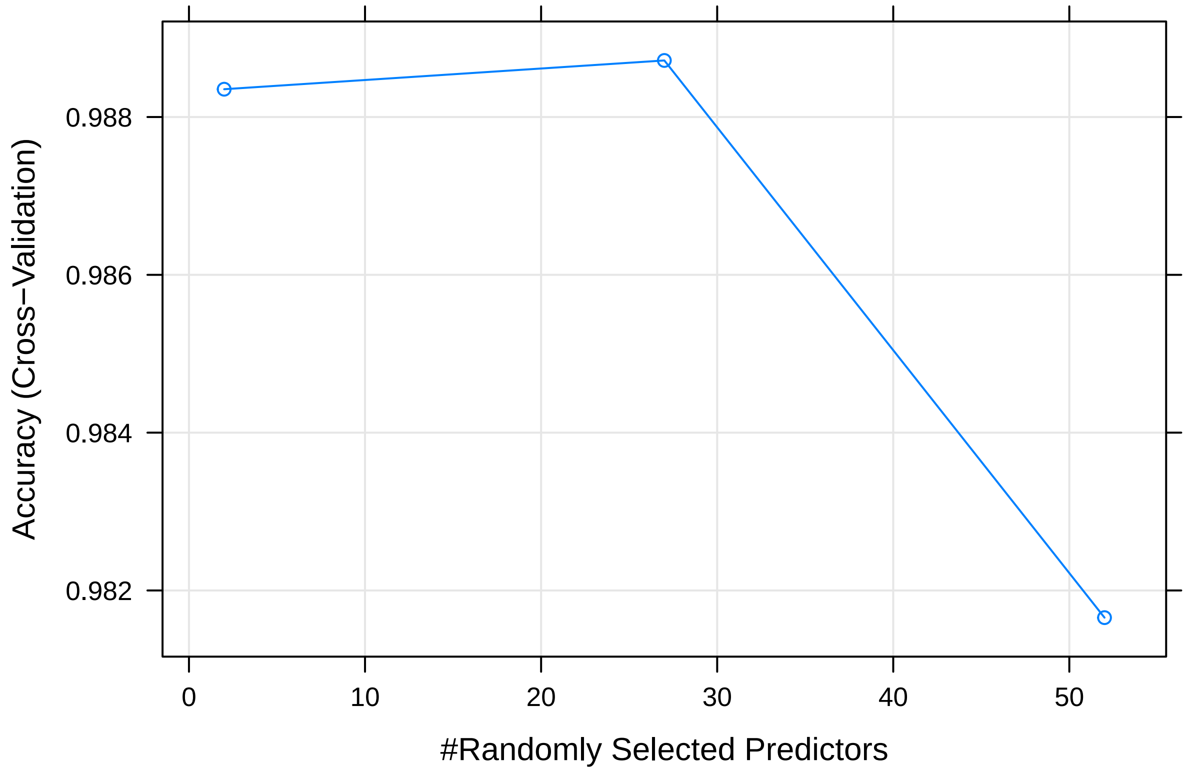
```
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   18 2633    7    0    0 0.009405568
## C    0   16 2371    9    0 0.010434057
## D    0    1   29 2222    0 0.013321492
## E    0    2    5    7 2511 0.005544554
```

estimate the performance of the model on the validation data set

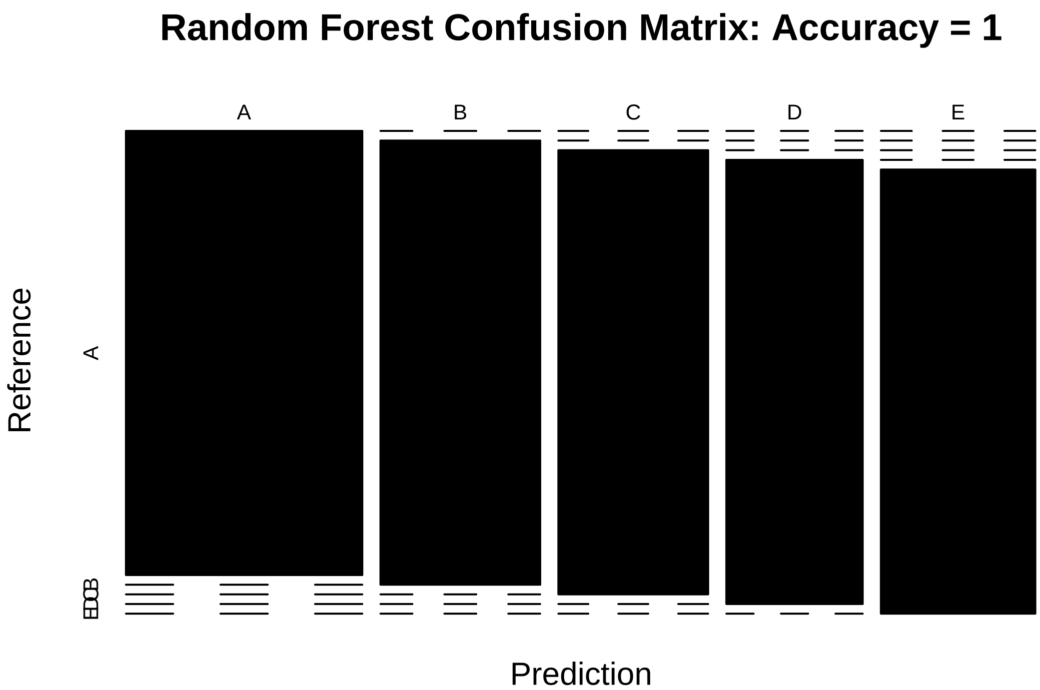
```r
predictRF1 <- predict(modRF1, newdata=testData1)
cmrf <- confusionMatrix(predictRF1,as.factor(testData1$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1165    0    0    0    0
##          B    0  787    0    0    0
##          C    0    0  738    0    0
##          D    0    0    0  672    0
##          E    0    0    0    0  761
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000    1.000    1.000   1.0000
## Specificity            1.0000   1.0000    1.000    1.000   1.0000
## Pos Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Neg Pred Value         1.0000   1.0000    1.000    1.000   1.0000
## Prevalence             0.2826   0.1909    0.179    0.163   0.1846
## Detection Rate         0.2826   0.1909    0.179    0.163   0.1846
## Detection Prevalence   0.2826   0.1909    0.179    0.163   0.1846
## Balanced Accuracy      1.0000   1.0000    1.000    1.000   1.0000
```

```r
plot(modRF1)
```




```r
plot(cmrf$table, col = cmrf$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =", round(cm
```

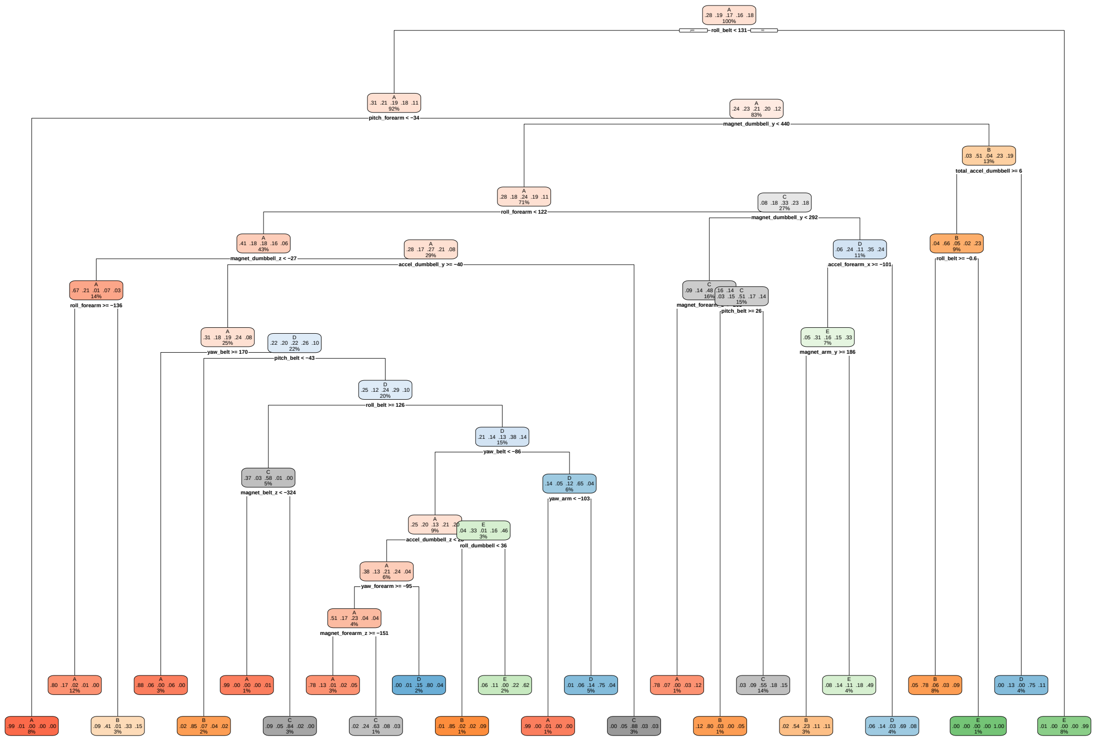


Classification Tree Visualization

We first obtail the model, and then we use the fancyRpartPlot() function to plot the classification tree as a dendogram.

```
set.seed(12345)
decisionTreeMod1 <- rpart(classe ~ ., data=trainData, method="class")
rpart.plot(decisionTreeMod1)
```

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



validate the model "decisionTreeModel" on the testData to find out how well it performs by looking at the accuracy variable

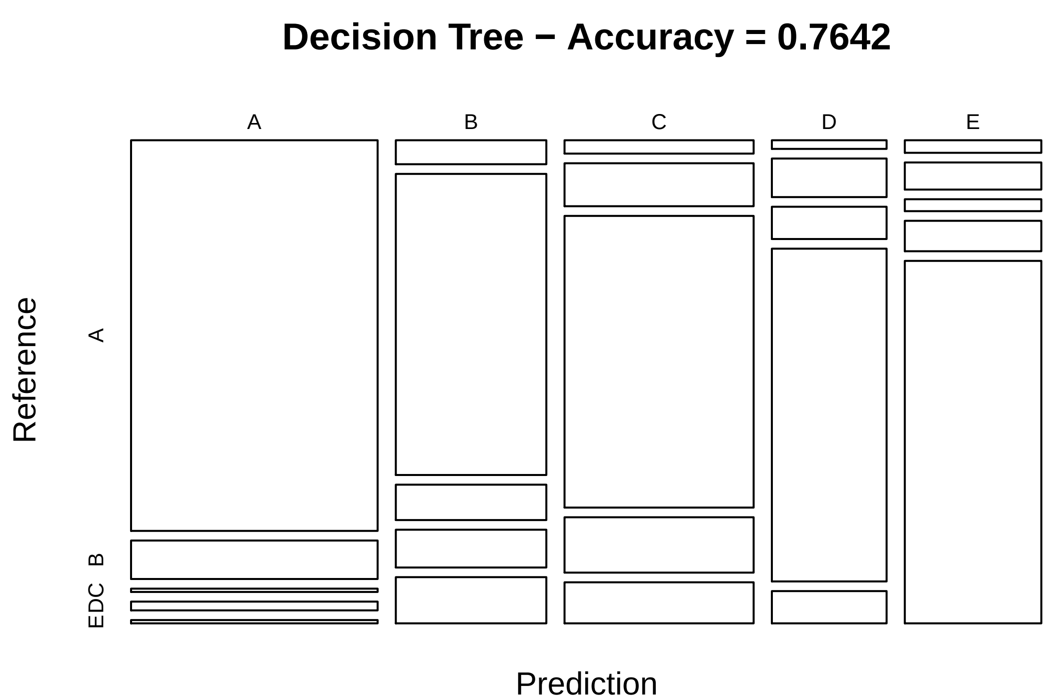
```r
predictTreeMod1 <- predict(decisionTreeMod1, testData1, type = "class")
cmtree <- confusionMatrix(predictTreeMod1,as.factor(testData1$classe))
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1067  105    9   24    9
##          B   40  502   59   63   77
##          C   28   90  611  116   86
##          D   11   49   41  423   41
##          E   19   41   18   46  548
##
## Overall Statistics
##
##                Accuracy : 0.7642
##                  95% CI : (0.751, 0.7771)
##     No Information Rate : 0.2826
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7015
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9159   0.6379   0.8279   0.6295   0.7201
## Specificity            0.9503   0.9284   0.9055   0.9589   0.9631
## Pos Pred Value         0.8789   0.6775   0.6563   0.7487   0.8155
## Neg Pred Value         0.9663   0.9157   0.9602   0.9300   0.9383
## Prevalence             0.2826   0.1909   0.1790   0.1630   0.1846
## Detection Rate         0.2588   0.1218   0.1482   0.1026   0.1329
## Detection Prevalence   0.2944   0.1797   0.2258   0.1370   0.1630
## Balanced Accuracy      0.9331   0.7831   0.8667   0.7942   0.8416
```

plot matrix results

```
plot(cmtree$table, col = cmtree$byClass,
     main = paste("Decision Tree - Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```



Result

Random Forest method is comparitively more accurate than Classification Tree