Peer-graded Assignment: Prediction Assignment Writeup

Fadi Bartik

1) Setting libraries

```
library(knitr)
library(caret)
```

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

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

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(corrplot)
set.seed(1337)
```

2) Setting and cleaning Input Data

```
# Download and define the Input datasets
UrlTraining <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
"
UrlTesting  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(UrlTraining))
testing  <- read.csv(url(UrlTesting))

inTraining  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainingSet <- training[inTraining, ]
TestingSet  <- training[-inTraining, ]

# remove variables with Nearly Zero Variance
temp <- nearZeroVar(TrainingSet)
TrainingSet <- TrainingSet[, -temp]
TestingSet  <- TestingSet[, -temp]

# remove variables that are mostly NA
temp2     <- sapply(TrainingSet, function(x) mean(is.na(x))) > 0.95
TrainingSet <- TrainingSet[, temp2==FALSE]
TestingSet  <- TestingSet[, temp2==FALSE]

# remove identification variables (columns 1 to 5)
TrainingSet <- TrainingSet[, -(1:5)]
TestingSet  <- TestingSet[, -(1:5)]
```

3) Prediction Models

We will compare the accuracy of the Random Forest and Decision Tree. The model with the best accuracy will be used as prediction on the testing dataset.

3.1) Random Forest

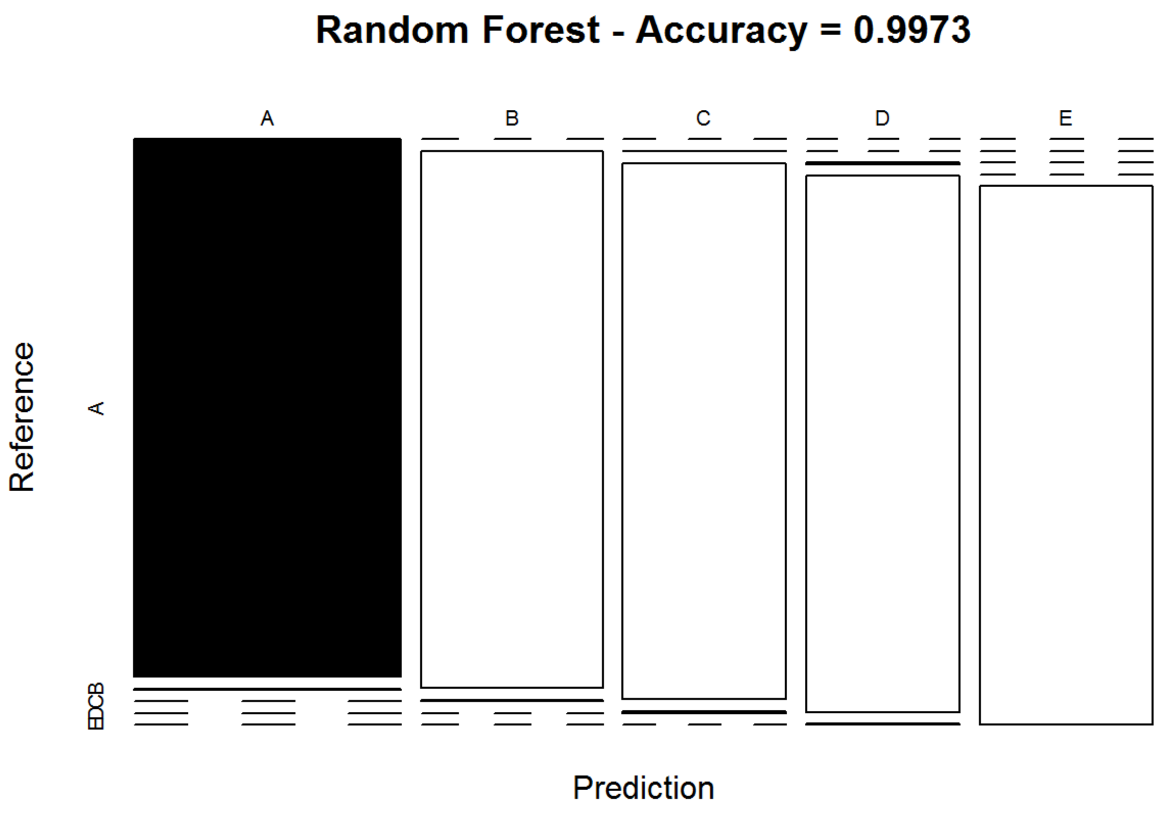
```
# Fitting
control <- trainControl(method="cv", number=3, verboseIter=FALSE)
RFfit <- train(classe ~ ., data=TrainingSet, method="rf", trControl=control)
RFfit$finalModel0
```

```
## NULL
```

```
# prediction on Test dataset
RF <- predict(RFfit, newdata=TestingSet)
RFmatrix <- confusionMatrix(RF, TestingSet$classe)
RFmatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1135    3    0    0
##          C    0    1 1020    4    0
##          D    0    0    3  960    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9956, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9965   0.9942   0.9959   0.9982
## Specificity            0.9993   0.9994   0.9990   0.9990   1.0000
## Pos Pred Value         0.9982   0.9974   0.9951   0.9948   1.0000
## Neg Pred Value         1.0000   0.9992   0.9988   0.9992   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1929   0.1733   0.1631   0.1835
## Detection Prevalence   0.2850   0.1934   0.1742   0.1640   0.1835
## Balanced Accuracy      0.9996   0.9979   0.9966   0.9974   0.9991
```

```
# Confusion matrix and accuracy
plot(RFmatrix$table, col = RFmatrix$byClass, main = paste("Random Forest - Accuracy
=", round(RFmatrix$overall['Accuracy'], 4)))
```



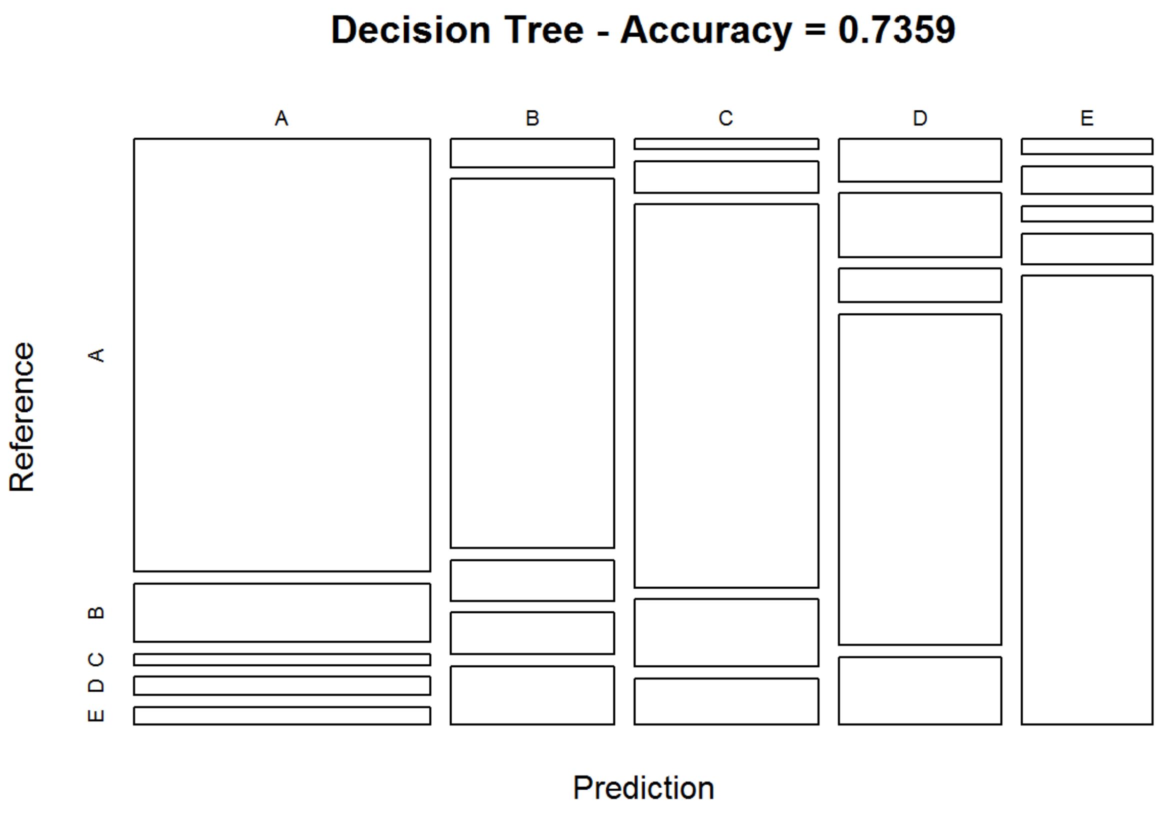
3.2) Prediction Model Building Method: Decision Trees

```
# fitting
Treefit <- rpart(classe ~ ., data=TrainingSet, method="class")

# prediction on Test dataset
Tree <- predict(Treefit, newdata=TestingSet, type="class")
Treematrix <- confusionMatrix(Tree, TestingSet$classe)
Treematrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1495  202   38   65   61
##          B   54  705   78   80  112
##          C   22   68  822  144  100
##          D   80  121   64  629  129
##          E   23   43   24   46  680
##
## Overall Statistics
##
##                Accuracy : 0.7359
##                  95% CI : (0.7245, 0.7472)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6649
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8931   0.6190   0.8012   0.6525   0.6285
## Specificity            0.9131   0.9317   0.9313   0.9199   0.9717
## Pos Pred Value         0.8033   0.6851   0.7111   0.6149   0.8333
## Neg Pred Value         0.9555   0.9106   0.9569   0.9311   0.9207
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2540   0.1198   0.1397   0.1069   0.1155
## Detection Prevalence   0.3162   0.1749   0.1964   0.1738   0.1387
## Balanced Accuracy      0.9031   0.7753   0.8662   0.7862   0.8001
```

```
# onfusion matrix and accuracy
plot(Treematrix$table, col = Treematrix$byClass,
     main = paste("Decision Tree - Accuracy =", round(Treematrix$overall['Accuracy'
], 4)))
```



4) Test Data Prediction data

The Random Forest model accuracy (0.9973) is higher than the Decision Trees model accuracy (0.7359).

For that reason we will choose the Random Forest model to apply on the testing data.

Below are the results of the predictions

```
final <- predict(RFfit, newdata=testing)
final
```

```
##  [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
```

Loading [Contrib]/a11y/accessibility-menu.js