Practical Machine Learning

Steve Morin

November 19, 2016

Synopsis

This project will predict the manner in which exercise was done from a test data set. The model will be trained using labeled data and an evaluation will be done of the most accurate modelling method.The model that is generated will be used to predict the ‘classe’ in the test data set.Prediction outcomes will be evaluated by calculating the confusion matrix for the training/test data sets from the provided training data.

Load Libraries

```
## Include required libraries
library(caret)
```

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

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

```
library(RColorBrewer)
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
```

```r
library(rpart)
library(rpart.plot)
options(warn=-1)
```

Load & Prepare Data

```r
# Set the seed

set.seed(11111)
```

```r
# Read testing data

testingData<-read.csv("D:\\Coursera\\Data Science Specialization\\Practical Machine Learning\\Ass
ignment\\pml-testing.csv")
```

```r
# Read training data
trainingData<-read.csv("D:\\Coursera\\Data Science Specialization\\Practical Machine Learning\\As
signment\\pml-training.csv")
```

```r
# Partition the training data
trainIndex<-createDataPartition(trainingData$classe,p=0.6,list=FALSE)
```

```r
# Subset the training data

trainDataPart<-trainingData[trainIndex, ]
```

```r
# Subset the testing data

testDataPart<-trainingData[-trainIndex, ]
```

```r
# Find columns with near zero variance

varNearZero <- nearZeroVar(trainDataPart)
```

```r
# Remove columns identified as near zero from the training and testing data sets

trainDataPart<-trainDataPart[,-varNearZero]
testDataPart<-testDataPart[,-varNearZero]
```

```r
# Find columns with greater than 75% NAs

columnsWithNA <- sapply(trainDataPart, function(x) mean(is.na(x))) > 0.75
```

```
# Remove columns with greater than 75% NAs

trainDataPart<-trainDataPart[,!columnsWithNA]
testDataPart<-testDataPart[,!columnsWithNA]
```

```
# Remove columns 1 - 5 from the train/test datasets as they are cannot be used in the analysis

trainDataPart<-trainDataPart[,-(1:5)]
testDataPart<-testDataPart[,-(1:5)]
```

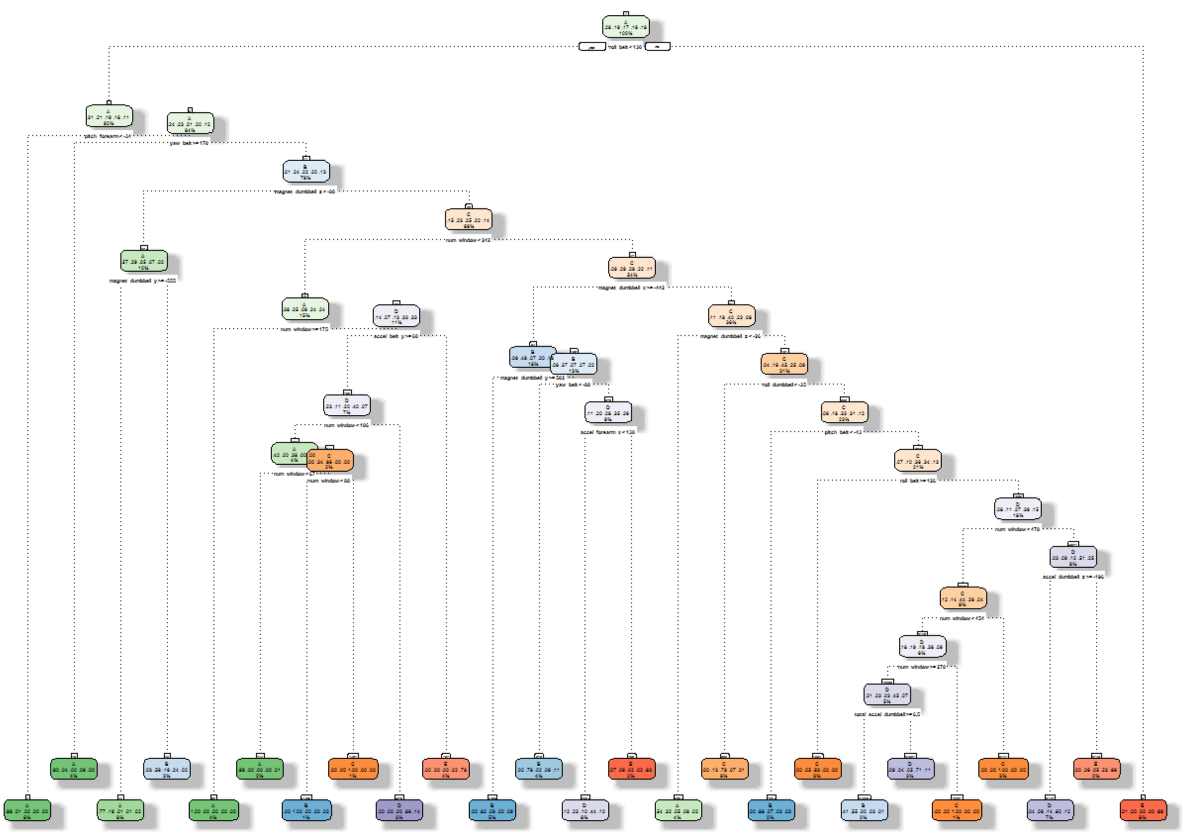
Decision Tree Method

```
# Create the Decision Tree

rpartModel <- rpart(classe ~ .,data=trainDataPart,method="class")

# Plot the Decision Tree

fancyRpartPlot(rpartModel)
```



Rattle 2016-Nov-19 16:16:13 morins

```
# Apply model to test data to obtain predicted classe

predictTestDataDT <- predict(rpartModel, newdata=testDataPart, type="class")
```

```
# Calculate the confusion matrix

confMatrixDT <- confusionMatrix(predictTestDataDT, testDataPart$classe)
confMatrixDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2006  264   21   67   25
##          B   82  940   85   97   90
##          C    1   82 1099   49    7
##          D  129  219  150  959  210
##          E   14   13   13  114 1110
##
## Overall Statistics
##
##                Accuracy : 0.7793
##                  95% CI : (0.7699, 0.7884)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7205
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8987   0.6192   0.8034   0.7457   0.7698
## Specificity            0.9328   0.9441   0.9785   0.8921   0.9760
## Pos Pred Value         0.8418   0.7264   0.8877   0.5753   0.8782
## Neg Pred Value         0.9586   0.9118   0.9593   0.9471   0.9496
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2557   0.1198   0.1401   0.1222   0.1415
## Detection Prevalence   0.3037   0.1649   0.1578   0.2125   0.1611
## Balanced Accuracy      0.9158   0.7816   0.8910   0.8189   0.8729
```

Random Forest Method

The second method tried was Random Forest using cross-validation for 3 iterations. Cross-validation is used to avoid overfitting the model and helps to make the model more generalizable to other data sets.

```
# Random Forest Method

RFModel <- train(classe ~ ., data=trainDataPart, method="rf",trControl=trainControl(method="cv",
number=3, verboseIter=FALSE))

RFModel$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: 0.34%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    1    0    0    1 0.0005973716
## B   10 2265    3    1    0 0.0061430452
## C    0    4 2048    2    0 0.0029211295
## D    0    0    8 1921    1 0.0046632124
## E    0    2    0    7 2156 0.0041570439
```

```
# Prediction based on Test dataset with labels

predictTestDataRF <- predict(RFModel, newdata=testDataPart)
confMatrixRF <- confusionMatrix(predictTestDataRF, testDataPart$classe)
confMatrixRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    9    0    0    0
##          B    0 1504    1    0    0
##          C    0    5 1367    8    0
##          D    0    0    0 1277    3
##          E    0    0    0    1 1439
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.995, 0.9977)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9908   0.9993   0.9930   0.9979
## Specificity            0.9984   0.9998   0.9980   0.9995   0.9998
## Pos Pred Value         0.9960   0.9993   0.9906   0.9977   0.9993
## Neg Pred Value         1.0000   0.9978   0.9998   0.9986   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1917   0.1742   0.1628   0.1834
## Detection Prevalence   0.2856   0.1918   0.1759   0.1631   0.1835
## Balanced Accuracy      0.9992   0.9953   0.9986   0.9963   0.9989
```

```
# Perform final prediction based on Test dataset without labels

predictFinalTestRF <- predict(RFModel, newdata=testingData)
predictFinalTestRF
```

```
##  [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
```

Summary

The Random Forest method is more accurate than the Decision Tree method in this case acheiving greater than 99% accuracy. As a result we used it to do the final prediction on the 20 original test cases.

——————————————— End of Assignment ————————————————-

```
```