Practical Machine Learning Course Project

Davis Balaba

Sunday, May 24, 2015

Step 1 : Load and inspect data

```
library(caret)
```

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

```
library(ggplot2)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
train <- read.csv("C:/Users/davis.balaba/Desktop/Coursera/Practical Machine Learning/pml-training.cs
v",na.strings=c("NA","#DIV/0!", ""), header=TRUE)

test <- read.csv("C:/Users/davis.balaba/Desktop/Coursera/Practical Machine Learning/pml-testing.cs
v",na.strings=c("NA","#DIV/0!", ""), header=TRUE)

#colnames(train)
```

We observe that columns 1 to 7 should be eliminated as they are not pertinent to the prediction process.

Step 2: Clean data

```
train <- train[,8:length(colnames(train))]
dim(train)
```

```
## [1] 19622   153
```

```
test <- test[,8:length(colnames(test))]

# Delete columns with all missing values
training2<-train[,colSums(is.na(train)) == 0]
testing2 <-test[,colSums(is.na(test)) == 0]
dim(training2)
```

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

```
dim(testing2)
```

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

Step 3: Split training sample 60/40 to create in-sample test set

```
## [1] 11776    53
```

```
## [1] 7846   53
```

Step 4a: Modeling using decision tree We examine the output variable and observe that it is a 5-level categorical variable. So first we attempt a decision tree model.

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2028  301   18  125   50
##          B   62  938  170   63  155
##          C   62  117 1068  199  183
##          D   50  108   82  836  125
##          E   30   54   30   63  929
##
## Overall Statistics
##
##                Accuracy : 0.7391
##                  95% CI : (0.7292, 0.7488)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6685
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9086   0.6179   0.7807   0.6501   0.6442
## Specificity            0.9120   0.9289   0.9134   0.9444   0.9724
## Pos Pred Value         0.8041   0.6758   0.6556   0.6961   0.8400
## Neg Pred Value         0.9617   0.9102   0.9517   0.9323   0.9239
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2585   0.1196   0.1361   0.1066   0.1184
## Detection Prevalence   0.3214   0.1769   0.2076   0.1531   0.1410
## Balanced Accuracy      0.9103   0.7734   0.8471   0.7972   0.8083
```

This approach yields ~73% accuracy.

Step 4b: Modeling using random forest

```
set.seed(90)
#Create model
model2 <- randomForest(classe ~ ., data=train_tr, method="class")
# Predict
prediction2 <- predict(model2, train_test, type = "class")
#Evaluate model
# Test results on our subTesting data set:
confusionMatrix(prediction2, train_test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    3    0    0    0
##          B    0 1511   10    0    0
##          C    0    4 1358   21    0
##          D    0    0    0 1265    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9934, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9939
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9954   0.9927   0.9837   1.0000
## Specificity            0.9995   0.9984   0.9961   1.0000   1.0000
## Pos Pred Value         0.9987   0.9934   0.9819   1.0000   1.0000
## Neg Pred Value         1.0000   0.9989   0.9985   0.9968   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1926   0.1731   0.1612   0.1838
## Detection Prevalence   0.2849   0.1939   0.1763   0.1612   0.1838
## Balanced Accuracy      0.9997   0.9969   0.9944   0.9918   1.0000
```

The random forest yields about 99% accuracy and will therefore be used for the final predictions.

Step 5: Final Prediction

```
prediction3 <- predict(model2, testing2, type="class")
prediction3
```

```
##  1  2  3  4  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
```