Machine Learning Final

Tim Lambert

5/10/2018

Approach

Using exercise data, we attempt to predict how a particular subject performed an exercise. By downloading the data from the website, getting rid of blank or erroneous data, and separating it into a training and test set, training and cross-validation can be performed using the caret and RandomForest packages in R.

```
library(caret)
```

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

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

```
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
```

```
set.seed(10000)

training.data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                          na.strings=c("NA","#DIV/0!", ""))
testing.data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                         na.strings=c("NA","#DIV/0!", ""))

#Delete columns with zero
training.data <- training.data[,colSums(is.na(training.data)) == 0]

#Number of NA values
sum(is.na(training.data))
```

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

```
#Delete admin data
training.data <- training.data[,-c(1:7)]

#Create training and test set for cross validation
index <- createDataPartition(training.data$classe, p = .75, list = FALSE)

#Create training and test set
training.data.train <- training.data[index,]
training.data.test <- training.data[-index,]
rm(index)
```

Model One - Decision Tree

First, a decision tree model is applied to the training set and a cross validation is applied to test set. Then, a confusion matrix is used to see how well the model did.

```
#Decision Tree Model
modfit1 <- train(data = training.data.train, classe ~ ., method = "rpart")

#Decision Tree Confusion Matrix
predict.modfit1 <- predict(modfit1, training.data.test)
confusionMatrix(training.data.test$classe, predict.modfit1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1273   17  103    0    2
##          B  378  334  237    0    0
##          C  407   26  422    0    0
##          D  348  161  295    0    0
##          E  134  118  252    0  397
##
## Overall Statistics
##
##                Accuracy : 0.4947
##                  95% CI : (0.4806, 0.5088)
##     No Information Rate : 0.5179
##     P-Value [Acc > NIR] : 0.9995
##
##                   Kappa : 0.3397
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5012  0.50915  0.32238       NA  0.99499
## Specificity            0.9484  0.85523  0.87955   0.8361  0.88812
## Pos Pred Value         0.9125  0.35195  0.49357       NA  0.44062
## Neg Pred Value         0.6389  0.91858  0.78093       NA  0.99950
## Prevalence             0.5179  0.13377  0.26692   0.0000  0.08136
## Detection Rate         0.2596  0.06811  0.08605   0.0000  0.08095
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7248  0.68219  0.60097       NA  0.94156
```

The decision tree model's accuracy is just shy of 50% meaning the out of sample error is just above 50%.

Random Forest Model

Next a Random Forest is applied to the training data set. Just as with the decision tree model, cross-validation is applied to test set and a confusion matrix used to see how well the model performed.

```
#Random Forest Model
modfit2 <- randomForest(data = training.data.train, classe ~ .)

#Random Forest Confusion Matrix
```

```
predict.modfit2 <- predict(modfit2, training.data.test)
confusionMatrix(training.data.test$classe, predict.modfit2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    2  947    0    0    0
##          C    0    5  850    0    0
##          D    0    0    4  799    1
##          E    0    0    3    4  894
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9937   0.9918   0.9950   0.9989
## Specificity            0.9997   0.9995   0.9988   0.9988   0.9983
## Pos Pred Value         0.9993   0.9979   0.9942   0.9938   0.9922
## Neg Pred Value         0.9994   0.9985   0.9983   0.9990   0.9998
## Prevalence             0.2847   0.1943   0.1748   0.1637   0.1825
## Detection Rate         0.2843   0.1931   0.1733   0.1629   0.1823
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9991   0.9966   0.9953   0.9969   0.9986
```

The random forest did much better with an accuracy of 99.59% meaning the out of sample error is .41%.

20 observations

Use the random forest model to predict the 20 observations.

```
#Predict the 20 from test data set
predict(modfit2, testing.data)
```

```
##  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
```