Practical Machine Learning Final Project

Ross

July 12, 2018

Introduction

In this project, we use data collected from the motion detection devices on FitBit etc. to predict the motion being undertaken by the wearers of those devices. The training data includes both the measurements and a factor varible identify the motion being undertaken during the measurements. The Test data include measurements but no identification of the motion, which is to be predicted from the model. After cleaning and partitioning the training data, we develop a random forest model with 99% accuracy and then apply it to the Test data set.

Get the needed packages

```
library(caret)
library(randomForest)
```

Get the Data

First, download the data and bring it into R.

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "pml-t
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "pml-te
df_training <- read.csv("~/pml-training.csv", na.strings=c("NA",""))
df_testing <- read.csv("~/pml-testing.csv", na.strings=c("NA",""))
```

Clean the Data

A visual inspection shows that many of the rows have a lot of NAs or are factors. We clean both the training data and the test data by removing rows which have more the 50% NAs and the first 7 rows (which are labeling and are not used by the model).

```
df_training2 <- df_training[, -which(colMeans(is.na(df_training)) > 0.5)]
df_training3 <- df_training2[, -c(1:7)]
df_testing2 <- df_testing[, -which(colMeans(is.na(df_testing)) > 0.5)]
df_testing3 <- df_testing2[, -c(1:7)]
```

Partion training Data

Next, partition the training data into a training set and a cross validation set. The cross validation set will be used to guage the accuracy of the model before applying it to the test data upon which we want to make a prediction.

```
set.seed(688)
inTrain <- createDataPartition(y = df_training3$classe, p = 0.7, list = FALSE)
Train <- df_training3[inTrain, ]
CrossEval <- df_training3[-inTrain, ]
```

Fit a Random Forest Model

Now we'll fit a random forest model to the Train set.

```
trControl <- trainControl(method = "cv", number = 2)
Fit <- train(classe ~ ., data = Train, method="rf", prox = TRUE, trControl = trControl)
```

Let's use the CrossEval partition to see if the model predicts accurately. The Fit model is 1.2 Gb and took 30 minutes to run, so it better.

```
pred <- predict(Fit, newdata = CrossEval)
confusionMatrix(pred, reference = CrossEval$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   10    0    0    0
##          B    0 1125    7    1    1
##          C    0    4 1011   11    3
##          D    0    0    8  952    2
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9894, 0.9941)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9899
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9877   0.9854   0.9876   0.9945
## Specificity            0.9976   0.9981   0.9963   0.9980   1.0000
## Pos Pred Value         0.9941   0.9921   0.9825   0.9896   1.0000
## Neg Pred Value         1.0000   0.9971   0.9969   0.9976   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1912   0.1718   0.1618   0.1828
## Detection Prevalence   0.2862   0.1927   0.1749   0.1635   0.1828
## Balanced Accuracy      0.9988   0.9929   0.9908   0.9928   0.9972
```

The Confusion Matrix shows that the Fit mode has an accuracy of more than 99%, so we'll stop there and use the Fit model to predict the movements in the test data.

```
Results <- predict(Fit, newdata = df_testing3)
Results
```

```
##  [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
```

Conclusion

Using the random forest function we were able to develop a model with 99% accuracy.