Practical Machine Learning Project

You

12/13/2019

Practical Machine Learning Project

Summary

This project strives to apply the techniques of machine learning to the supplied dataset in order to develop the ability to predict if a measured activity is being performed correctly, specificall dumbbell curls. The dataset includes data capturing the movement of the body and the dumbbell when this activity is being performed both correctly and incorrectly. The model is built using 52 predictors which actually record the movement of different parts of the body and the dumbbell, using the random forest method in the "ranger" option. Applied to a validation dataset, it produced an accuracy of 99%.

Building the model

First, it is necessary to load the datasets into R. We will take the dataset provided for training and split it to provide a training set and a validation set to test our model, but first we need to decide which predictors to use. Although the dataset provides a large number of variables (160), only 52 capture a large amount of information about the movement. Some other variables included were potentially considered, but had an enormous number of NA values. There was insufficient documentation to suggest why those variables had so many NA values, and the sheer number made an approach such as K nearest neighbors seem unwise, due to there being insufficient data to reliably extrapolate from.

```
library(caret)
```

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

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

```
#load data
testing <- read.csv("pml-testing.csv", header = TRUE)
training_og <- read.csv("pml-training.csv", header = TRUE)

#select predictors with both lots of data and direct relevance to activity

preds <- c("roll_belt", "pitch_belt", "yaw_belt", "total_accel_belt", "gyros_belt_x",
           "gyros_belt_y", "gyros_belt_z", "accel_belt_x", "accel_belt_y", "accel_belt_z",
           "magnet_belt_x", "magnet_belt_y", "magnet_belt_z", "roll_arm", "pitch_arm",
           "yaw_arm", "total_accel_arm", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z",
           "accel_arm_x", "accel_arm_y", "accel_arm_z", "magnet_arm_x", "magnet_arm_y",
           "magnet_arm_z", "roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell", "total_accel_dumbbell",
           "gyros_dumbbell_x", "gyros_dumbbell_y", "gyros_dumbbell_z", "accel_dumbbell_x",
           "accel_dumbbell_y", "accel_dumbbell_z", "magnet_dumbbell_x", "magnet_dumbbell_y",
           "magnet_dumbbell_z", "roll_forearm", "pitch_forearm", "yaw_forearm", "total_accel_forearm",
           "gyros_forearm_x", "gyros_forearm_y", "gyros_forearm_z", "accel_forearm_x",
           "accel_forearm_y", "accel_forearm_z", "magnet_forearm_x", "magnet_forearm_y",
```

```
           "magnet_forearm_z")

#separate training set into training and validation sets - create index
inTrain <- createDataPartition(training_og$classe, p = 0.75)[[1]]

#actually separate
training <- training_og[inTrain, c(preds, "classe")]
validation <- training_og[-inTrain, c(preds, "classe")]
```

Now that we have loaded and partitioned the datasets, we can now apply our machine learning algorithm. The model is built using the random forest technique captured in the "ranger" option.

```
#set a seed value
set.seed(62433)

#create a random forest model
mdl <- train(classe ~ ., data = training, method = "ranger")

mdl
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##    2    gini        0.9887229  0.9857323
##    2    extratrees  0.9863388  0.9827153
##   27    gini        0.9899080  0.9872332
##   27    extratrees  0.9921045  0.9900120
##   52    gini        0.9829614  0.9784428
##   52    extratrees  0.9928223  0.9909201
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 52, splitrule =
##  extratrees and min.node.size = 1.
```

```
#test against validation
pred <- predict(mdl, newdata = validation)

#check accuracy
confusionMatrix(pred, validation$classe)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  948    3    0    0
##          C    0    1  852    7    0
##          D    0    0    0  793    1
##          E    0    0    0    4  900
##
## Overall Statistics
##
##                Accuracy : 0.9967
##                  95% CI : (0.9947, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9965   0.9863   0.9989
## Specificity            1.0000   0.9992   0.9980   0.9998   0.9990
## Pos Pred Value         1.0000   0.9968   0.9907   0.9987   0.9956
## Neg Pred Value         1.0000   0.9997   0.9993   0.9973   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1737   0.1617   0.1835
## Detection Prevalence   0.2845   0.1939   0.1754   0.1619   0.1843
## Balanced Accuracy      1.0000   0.9991   0.9973   0.9930   0.9989
```

Using the confusion matrix function, we achieve a reported 99% accuracy rating, indicating excellent model performance. We can now apply this to the testing dataset and achieve the following results:

```
test_pred <- predict(mdl, newdata = testing[,preds])

summary(test_pred)
```

```
## A B C D E
## 7 8 1 1 3
```