Execrcise Analysis Using Random Forest Prediction

gv

November 5, 2016

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it.

For this project, six participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set.

Data Pre-Processing

I loaded the test and training data and removed a few uneccessary variables that contained ID and timestamp information.

```
library(AppliedPredictiveModeling)
library(caret)
library(pgmm)
require(ggplot2)
require(dplyr)

# read in training and test data

inTrain <- read.csv("training.csv", na.strings = c("NA", ""))

inTest <- read.csv("testing.csv", na.strings = c("NA", ""))

# remove first few columns with ID and time information.

inTrain <- inTrain[,7:160]
inTest <- inTest[,7:160]
```

There are several variables that are mostly empty(NA’s). For example, the variable “kurtosis_roll_belt” has 19216 missing values. In order to reduce the size of the data set, they have been removed.

```
#  remove mostly empty variables

sum(is.na(inTrain$kurtosis_roll_belt))
```

```
## [1] 19216
```

```
goodCols <- apply(is.na(inTrain),2,sum) < 19200

inTrain <- inTrain[, goodCols]
```

Here are the remaining variables.

```
##  [1] "num_window"           "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
## [37] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"    "roll_forearm"         "pitch_forearm"
## [43] "yaw_forearm"          "total_accel_forearm"  "gyros_forearm_x"
## [46] "gyros_forearm_y"      "gyros_forearm_z"      "accel_forearm_x"
## [49] "accel_forearm_y"      "accel_forearm_z"      "magnet_forearm_x"
## [52] "magnet_forearm_y"     "magnet_forearm_z"     "classe"
```

There are 54 variables remaining, “classe” is the variable we are trying to predict, and the other 53 are potential predictors.

Model Build Method

1. Partition the training data

The training data set is quite large with 19622 observations of 54 variables. I partitioned the training data set, using 30% of the data for model building, and reserving 70% of the training data to be used to estimate the out of sample accuracy.

```
#  partition training data into smaller data set

partTrain <- createDataPartition(y=inTrain$classe,p=0.3,list=FALSE)
inTrainSmall <- inTrain[partTrain,]

# reserve the rest for out of sample accuracy estimation
valDat <- inTrain[-partTrain,]
```

2. Build a Random Forest model with 3-fold cross validation

I used Random Forest prediction to build a model that predicts the value of the categorical variable “classe”. The caret package in R implements the random forest algorithm and **allows for cross validation at the same time**. I used “K-fold” cross validation with K=3.

```
# create random forest model

modelRF <- train(classe~.,data=inTrainSmall, method="rf", trControl=trainControl(method="cv",num
ber=3), prox=TRUE,allowParallel=TRUE)
print(modelRF)
```

```
## Random Forest
##
## 5889 samples
##   53 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 3926, 3926, 3926
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9716420  0.9641023
##   27    0.9845475  0.9804495
##   53    0.9733401  0.9662769
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
print(modelRF$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, proximity = TRUE,      allowParallel = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.9%
## Confusion matrix:
##      A    B    C   D    E class.error
## A 1672    1    0   0    1 0.001194743
## B   11 1119    7   3    0 0.018421053
## C    0    8 1016   3    0 0.010710808
## D    0    0   11 954    0 0.011398964
## E    0    2    0   6 1075 0.007386888
```

3. Estimate the Out of Sample accuracy

I estimated the out of sample accuracy by running the final model on the validation data.

```
# estimate test set error rate using validation data

valPred <- predict(modelRF,valDat)
confusionMatrix(valPred,valDat$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906   38    0    0    0
##          B    0 2600   12    7    3
##          C    0   17 2379   50    2
##          D    0    2    4 2194   15
##          E    0    0    0    0 2504
##
## Overall Statistics
##
##                Accuracy : 0.9891
##                  95% CI : (0.9872, 0.9907)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9862
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9785   0.9933   0.9747   0.9921
## Specificity            0.9961   0.9980   0.9939   0.9982   1.0000
## Pos Pred Value         0.9904   0.9916   0.9718   0.9905   1.0000
## Neg Pred Value         1.0000   0.9949   0.9986   0.9951   0.9982
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1893   0.1732   0.1598   0.1823
## Detection Prevalence   0.2872   0.1909   0.1783   0.1613   0.1823
## Balanced Accuracy      0.9981   0.9883   0.9936   0.9864   0.9960
```

4. Conclusion

The prediction accuracy on the validation data is 99.2%, which is very high for out of sample accuracy. This indicates that the model is certainly not “overfitted”, and should work very well to predict whether weights were lifted correctly or not.