Practical Machine Learning

Ignacio Peletier

3/5/2021

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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Cleaning

Let's take a look at the size of the datasets and the variables we have available:

```
dim(trainSet)
```

```
## [1] 19622   160
```

```
dim(examinationSet)
```

```
## [1]  20 160
```

Just to clarify, we will have 3 datasets: Train, Test and Examination. The Examination lacks the main variable that we will be predicting: $classe$. The train and test set will both come from the file $pml - training.csv$.

After removing missing values and filtering some columns we have:

```
trainSet <- trainSet[, colSums(is.na(trainSet)) == 0]
examinationSet <- examinationSet[, colSums(is.na(examinationSet)) == 0]

classe <- trainSet$classe
columns <- grepl("^X|timestamp|window", names(trainSet))
trainSet <- trainSet[, !columns]
trainSet <- trainSet[, sapply(trainSet, is.numeric)]
trainSet$classe <- classe
columns <- grepl("^X|timestamp|window", names(examinationSet))
examinationSet <- examinationSet[, !columns]
examinationSet <- examinationSet[, sapply(examinationSet, is.numeric)]

dim(trainSet)
```

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

```
dim(examinationSet)
```

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

Model Training

We will do a 60% to 40% splits with the original training data, this way we obtain both the train data and the test data.

```
set.seed(42)
inTrain <- createDataPartition(trainSet$classe, p=0.60, list=FALSE)
trainData <- trainSet[inTrain, ]
testData <- trainSet[-inTrain, ]
```

We will do a 5-fold cross validation on a Random Forest model with 50 trees:

```
controlCV <- trainControl(method="cv", 5)
rf <- train(classe ~ ., data=trainData, method="rf")
rf
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9861085  0.9824246
##   27    0.9862272  0.9825783
##   52    0.9766375  0.9704488
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

The results on the testing set are:

```
testPred <- predict(rf, testData)
confusionMatrix(testData$classe, testPred)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2226    5    1    0    0
##          B   14 1503    1    0    0
##          C    0    8 1355    5    0
##          D    0    0   10 1272    4
##          E    0    0    2    2 1438
## 
## Overall Statistics
##                                            
##                Accuracy : 0.9934          
##                  95% CI : (0.9913, 0.995) 
##     No Information Rate : 0.2855          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                            
##                   Kappa : 0.9916          
##                                            
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9938   0.9914   0.9898   0.9945   0.9972
## Specificity            0.9989   0.9976   0.9980   0.9979   0.9994
## Pos Pred Value         0.9973   0.9901   0.9905   0.9891   0.9972
## Neg Pred Value         0.9975   0.9979   0.9978   0.9989   0.9994
## Prevalence             0.2855   0.1932   0.1745   0.1630   0.1838
## Detection Rate         0.2837   0.1916   0.1727   0.1621   0.1833
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9963   0.9945   0.9939   0.9962   0.9983
```

We ended up with a pretty high accuracy on the testing set with a narrow 95 CI: $(0.9913, 0.995)$

Predictions in the Examination Set

We now predict on the examination set, where the $classe$ are unknown:

```
examination <- predict(rf, examinationSet[, -length(names(examinationSet))])
examination
```

```
##  [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
```

Feature Importance

Random Forest can be used to show feature importance, here we see which are the most relevant features:

```
varImp(rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          57.62
## yaw_belt               56.00
## roll_forearm           47.44
## pitch_belt             43.70
## magnet_dumbbell_y      42.94
## magnet_dumbbell_z      42.73
## accel_dumbbell_y       21.89
## magnet_dumbbell_x      17.17
## accel_forearm_x        16.92
## roll_dumbbell          16.59
## accel_dumbbell_z       15.32
## magnet_forearm_z       15.24
## magnet_belt_z          14.84
## accel_belt_z           13.57
## total_accel_dumbbell   12.84
## magnet_belt_x          11.06
## magnet_belt_y          10.68
## yaw_arm                10.51
## gyros_belt_z           10.30
```

Clearly, The most important feature is $roll_belt$.