Building a Model to Predict Classe

Maria Lauve

August 18, 2017

Assignment

Six subjects are asked to repeatedly perform an activity five different ways. Each of the five ways of performing the task is assigned to a “classe” (A, B, C, D, or E).

For each repetition of the activity, a number of metrics are recorded from accelerometers placed on the belt, arm, dumbbell, and forearm of the subjects.

The purpose of this assignment is to build a model that is able to predict which of the five methods for performing a task a subject is using based on the recorded accelerometer measurements.

Setting up the data and environment

The first thing I do is call on the necessary libraries and import the test and training sets.

```
library(caret)
```

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

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

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
train = read.csv("./pml-training.csv")
test = read.csv("./pml-testing.csv")
```

Data exploration

The next thing I do is some data exploration to better understand the nature of the data.

I confirm that there are five classes: A, B, C, D, E. And that there are six subjects: adelmo, charles, eurico, jeremy, pedro.

I also run some trellis plots to see how the reported accelerometer measurements vary by user_name (subject) and classe.

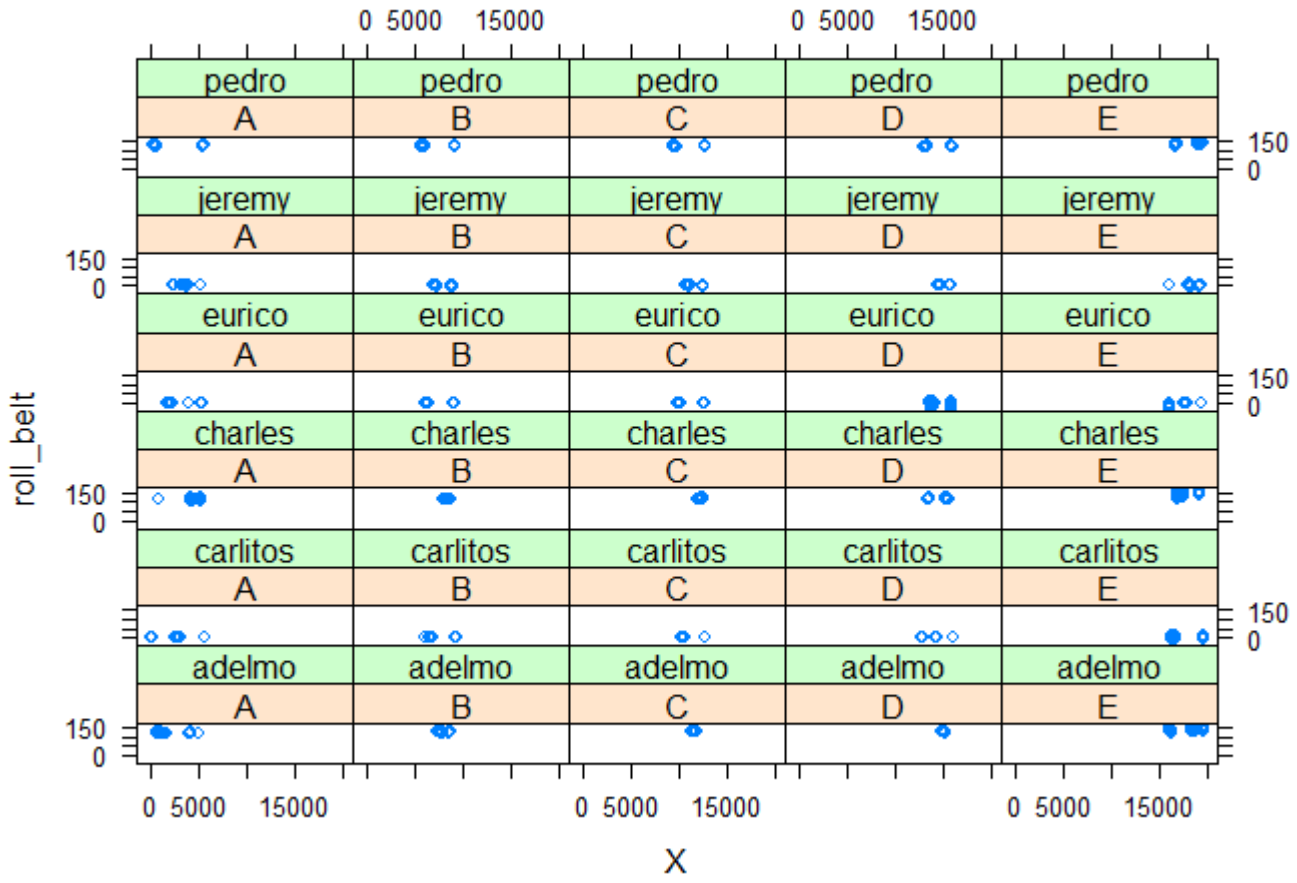
```
unique(train$classe)
```

```
## [1] A B C D E
## Levels: A B C D E
```

```
unique(train$user_name)
```

```
## [1] carlitos pedro    adelmo   charles  eurico   jeremy
## Levels: adelmo carlitos charles eurico jeremy pedro
```

```
#Example trellis plot by user_name, classe
xyplot(roll_belt ~ X | classe * user_name, train)
```



Building a model

As a first attempt at builing a model, I decide to try a model that includes all accelerator measurements that appear to be populated in the dataset.

Using these features, I first attempt to run a single decision-tree model using the rpart function. To save on computing power, I use a 10-fold cross-validation model that repeats one time.

```
# Set up caret to perform 10-fold cross validation repeated X times
caret.control <- trainControl(method="repeatedcv",number=10,repeats=1)

rpart.cv <- train(classe ~ .,
                  data = train[, features],
                  method = "rpart",
                  trControl = caret.control,
                  tuneLength = 7)
```

The results of the cross-validation suggest that the predictive power of this initial model is not very good.

```
# Display the results of the cross validation run
rpart.cv
```

```
## CART
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 17661, 17660, 17659, 17662, 17659, 17659, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02001139  0.6432615  0.55146076
##   0.02008261  0.6418343  0.54975513
##   0.02193420  0.5563640  0.42548694
##   0.02983905  0.5308861  0.38828876
##   0.03567868  0.5092778  0.35925176
##   0.05998671  0.4165795  0.20896217
##   0.11515454  0.3235575  0.05970544
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.02001139.
```

```
# Display the standard deviation of the model
cat(paste("\nCross validation standard deviation:",
          sd(rpart.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation standard deviation: 0.0209450932990245
```

Finally, I run the same model, but use a random forest in place of rpart. Again, I save on computing power by using a 5-tree model even though using many more trees may have improved the predicting power.

```
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 1)

# For random forest
rf.cv <- train(classe ~ .,
               data = train[, features],
               method = "rf",
               trControl = caret.control,
               tuneLength = 7,
               ntree = 5,
               importance = TRUE)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Display the results of the cross validation run
rf.cv
```

```
## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 17659, 17659, 17661, 17660, 17659, 17659, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9684551  0.9600940
##   10    0.9856293  0.9818218
##   18    0.9859855  0.9822716
##   27    0.9860874  0.9824004
##   35    0.9862911  0.9826607
##   43    0.9837450  0.9794413
##   52    0.9816531  0.9767951
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 35.
```

```
# Display the standard deviation
cat(paste("\nCross validation standard deviation:",
          sd(rf.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation standard deviation: 0.00354101833428266
```

Interpreting the results

The model output indicates that, within the training set, the model has 99 percent accuracy using the model's "best" parameters.

We can further tell which variables had the most explanator power by graphing the "importance" of variables using the "importance" function.

Bigger values indicate more powerful predictors.

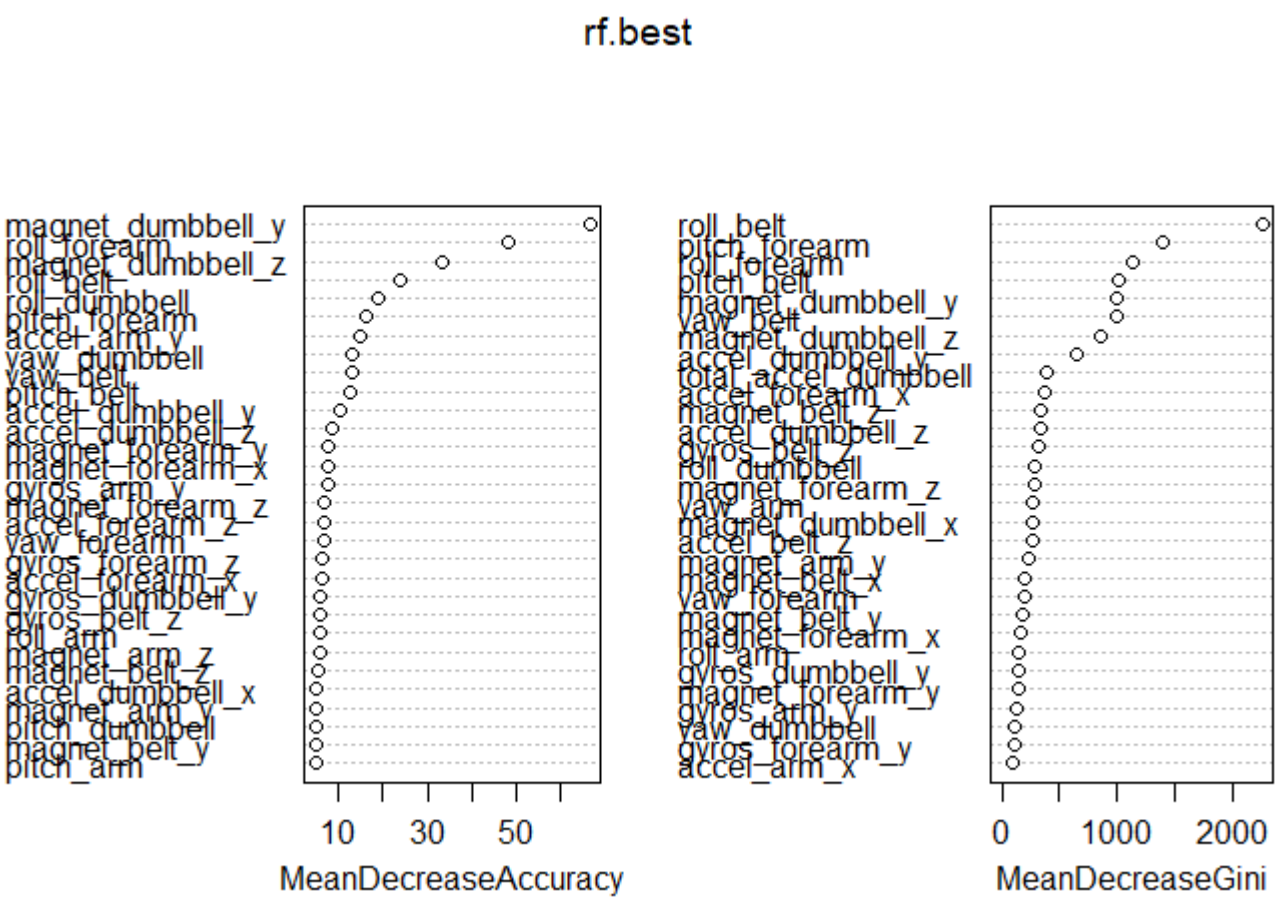
```
# Pull out the the trained model using the best parameters
rf.best <- rf.cv$finalModel

# Show variable importance
importance(rf.best)
```

```
##                             A         B          C         D          E
## roll_belt            10.0112860 11.155012 19.2114658 11.262294 12.0237432
## pitch_belt            4.6604590 23.235911 10.2222641  5.326815  4.4586969
## yaw_belt             19.1863305  5.940858  7.5152305 14.343930 11.4188974
## total_accel_belt      1.2498354  2.268302  1.3328078  1.537276  1.4932911
## gyros_belt_x          2.0326503  1.976972  2.8907626  2.886063  1.0935035
## gyros_belt_y          0.4887558  2.320403  3.0755732  1.586925  1.7046345
## gyros_belt_z          2.7013938 11.630871  3.0086589  4.158167  5.2387276
## accel_belt_x          4.2152368  1.465724  1.3729866  2.520920  2.4221116
## accel_belt_y          1.1180340  1.324670  1.1689249  1.295170  1.2585249
## accel_belt_z          1.3011438  2.672514  3.6762308  1.935474  1.9560518
## magnet_belt_x         4.7034118  3.955776  2.5249656  3.067071  3.7572555
## magnet_belt_y         3.1110489  3.149594  8.3391763  2.391106 10.1136793
## magnet_belt_z         2.7151496  3.147575  3.3265723  3.459628  2.8364995
## roll_arm              2.9813920  1.747335  4.3233271  4.167569 13.2263631
## pitch_arm             2.5866949  5.487693  2.3943390  3.628115  1.2719304
## yaw_arm               4.2044197  3.023585  1.3835213  4.338914  3.9596633
## total_accel_arm       1.4821358  3.453693  3.6910458  2.297952  1.5886649
## gyros_arm_x           2.7402076  5.553777  4.0657912  3.495831  1.9079051
## gyros_arm_y           7.0960839  4.569347  1.8800873  3.103405  1.8027780
## gyros_arm_z           0.8992785  1.863899  1.9302598  2.740422 -0.4231080
## accel_arm_x           1.4761255  2.837662  3.1392036  3.165073  2.6221510
## accel_arm_y           3.9772426  9.220143  0.8744618  1.954918  3.1360939
## accel_arm_z           1.0519274  3.256515  1.7394232  4.403754  3.1418853
## magnet_arm_x          3.1952916  6.615258  2.0599715  2.519611  2.1317636
## magnet_arm_y          3.6532203  2.952236  3.0202157  5.736481  3.1932554
## magnet_arm_z          2.3506801  5.160069  2.1529336  2.515822  1.9208218
## roll_dumbbell         3.1055614  8.794504  2.6194676  4.013568  7.3500040
## pitch_dumbbell        2.4986241  3.168350  2.6458930  1.924506  2.9878071
## yaw_dumbbell          4.1823653  7.420693  3.7942619  3.732768  2.5824465
## total_accel_dumbbell  2.6192728  7.803162  2.3160473  4.525199  3.3301254
## gyros_dumbbell_x      1.3263371  7.125632  3.3321631  2.412832  1.8246927
## gyros_dumbbell_y      4.0038698  7.491825  2.4284424  3.340041  4.8761175
## gyros_dumbbell_z      2.5685936  2.547050  2.2538632  2.045071  2.1007167
## accel_dumbbell_x      2.0733906  1.393371  3.8907018  2.717914  1.3585605
## accel_dumbbell_y      6.6234572 14.464112 19.5539481  5.075800  3.3711513
## accel_dumbbell_z      4.3186073 17.174480  2.7826700  5.447742  6.9884523
## magnet_dumbbell_x     1.9693369  1.333045  2.7549040  3.725825  3.6319404
## magnet_dumbbell_y    20.8510219 20.982583 42.6284670 29.823841 20.0868210
## magnet_dumbbell_z    32.3975077 12.934311  9.6286779 14.892744 10.4668113
## roll_forearm         55.9915211 15.706881 35.9802260 23.736525 17.1358449
## pitch_forearm         6.8356324 17.220965 19.6010810  5.534631 14.6555414
## yaw_forearm           4.5664793  4.037091  3.2032674  5.648517  2.4818038
## total_accel_forearm   3.1104045  2.003408  2.1754283  2.351627  0.4330661
## gyros_forearm_x       1.2959609  1.118034  1.7499725  1.696984  1.1180340
## gyros_forearm_y       1.6346410  4.668897  3.0939322  3.374281  6.7518474
## gyros_forearm_z       2.1938784  8.670422  2.2978830  3.694709  1.9483519
## accel_forearm_x       2.4202247  8.055920  3.9038841  7.421877  4.4572284
## accel_forearm_y       1.9263657  2.960973  4.2230785  2.090014  2.4922855
## accel_forearm_z       2.3385450  2.311397  2.4018839  2.474047  2.6589563
## magnet_forearm_x      3.7596477  3.213820  4.1546116  3.920283  3.3860996
## magnet_forearm_y      4.6738422  4.719831  2.8221295  6.737079  3.7387159
## magnet_forearm_z      3.2272325  4.026585  8.3344673  3.742252  6.1970255
```

```
##                      MeanDecreaseAccuracy MeanDecreaseGini
## roll_belt                       23.628062       2263.63552
## pitch_belt                      12.606964       1020.30902
## yaw_belt                        13.059978        994.65061
## total_accel_belt                 2.045747         64.18686
## gyros_belt_x                     2.955301         51.78873
## gyros_belt_y                     2.913048         40.48342
## gyros_belt_z                     5.791810        326.28750
## accel_belt_x                     3.056224         33.86289
## accel_belt_y                     2.521560         25.76367
## accel_belt_z                     2.390425        263.68272
## magnet_belt_x                    4.003649        206.89614
## magnet_belt_y                    4.846743        177.39443
## magnet_belt_z                    5.284921        344.63749
## roll_arm                         5.782061        150.64996
## pitch_arm                        4.824403         90.59872
## yaw_arm                          3.305696        269.81211
## total_accel_arm                  3.604546         33.85844
## gyros_arm_x                      4.104294         76.60414
## gyros_arm_y                      7.488216        137.15251
## gyros_arm_z                      3.147563         21.29092
## accel_arm_x                      3.990271        101.57335
## accel_arm_y                     14.976388         63.02913
## accel_arm_z                      2.254164         31.78909
## magnet_arm_x                     3.779117         70.78573
## magnet_arm_y                     4.950572        226.96901
## magnet_arm_z                     5.647479         88.34181
## roll_dumbbell                   18.694985        287.60895
## pitch_dumbbell                   4.923860         72.54743
## yaw_dumbbell                    13.208141        108.12882
## total_accel_dumbbell             4.812328        381.25719
## gyros_dumbbell_x                 4.299986         89.52883
## gyros_dumbbell_y                 5.809181        145.51035
## gyros_dumbbell_z                 3.390292         43.37404
## accel_dumbbell_x                 4.973432         42.89375
## accel_dumbbell_y                10.313063        653.20094
## accel_dumbbell_z                 8.312017        335.85898
## magnet_dumbbell_x                2.624074        263.71836
## magnet_dumbbell_y               66.528743       1002.46372
## magnet_dumbbell_z               33.435837        862.77332
## roll_forearm                    48.023706       1129.69000
## pitch_forearm                   16.375275       1387.40072
## yaw_forearm                      6.695464        199.32395
## total_accel_forearm              4.569093         47.02584
## gyros_forearm_x                  1.606955         11.41102
## gyros_forearm_y                  4.300857        106.87737
## gyros_forearm_z                  6.327247         48.74721
## accel_forearm_x                  6.123088        371.95707
## accel_forearm_y                  4.047467         73.74942
## accel_forearm_z                  6.737785         76.57387
## magnet_forearm_x                 7.669448        167.66015
## magnet_forearm_y                 7.701459        141.54185
## magnet_forearm_z                 6.741495        287.50841
```

```
varImpPlot(rf.best)
```



Conclusion

No further adjustments to the model were made given that this model managed to correctly predict 20/20 of the test set.