Machine Learning - Excercise Analysis

Mike

1/19/2022

Executive Summary

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

DATA

The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
The test data are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Load data

```
# load data files from working directory ('data' for web lab)
library(readr)
x_test<-read.table("pml-testing.csv", header=TRUE, sep=",", na.strings=c("NA"
,"","#DIV/0!"))
x_train<-read.table("pml-training.csv", header=TRUE, sep=",", na.strings=c("N
A","","#DIV/0!"))
```

Review and clean data

```
# remove variables with significant NA's
# remove 'new_window' variable with only 1 factor level in training set
# remove colums (2 & 5) as non-numeric variables
testdata<-x_test[,colSums(is.na(x_test))==0]
traindata<-x_train[,colSums(is.na(x_train))==0]
testdata<-subset(testdata, select=-c(X,raw_timestamp_part_1,raw_timestamp_par
t_2,num_window,new_window,user_name,cvtd_timestamp,problem_id))
```

```
traindata<-subset(traindata, select=-c(X,raw_timestamp_part_1,raw_timestamp_p
art_2,num_window,new_window,user_name,cvtd_timestamp))

# Identifying variables that will not be good predictors
library(caret)
## Loading required package: ggplot2
## Loading required package: lattice
nsv<-nearZeroVar(traindata,saveMetrics=TRUE)
nsv
```

CROSS VALIDATION and DATA PARTITIONING

```
# Using 'caret' package; partitioning training data into trainbuild and train
test data sets
library(caret)
library(kernlab)

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

traindata$classe<-as.factor(traindata$classe)
inTrain <- createDataPartition(y=traindata$classe, p=0.75, list=FALSE)
trainbuild<-traindata[inTrain,]
traintest<-traindata[-inTrain,]
dim(trainbuild)

## [1] 14718    53

dim(traintest)

## [1] 4904   53
```

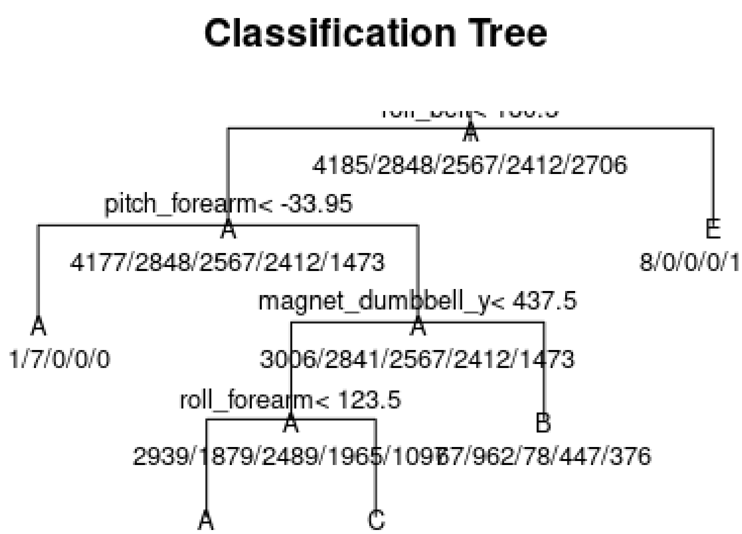
BUILD & EVALUATE DECISION TREE MODELS

1) Build RPART Classification Tree

```
# build the model with 'rpart'...one of R's regression/classification tree fu
nctions
# library(caret)
library(rpart)
library(rpart.plot)
set.seed(32343)
modelRPART<-train(classe ~., method="rpart",data=trainbuild)
finModRPART<-modelRPART$finalModel
print(finModRPART)
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 13477  9300 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1178     7 A (0.99 0.0059 0 0 0) *
##      5) pitch_forearm>=-33.95 12299  9293 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 437.5 10369  7430 A (0.28 0.18 0.24 0.19 0.11
)
##         20) roll_forearm< 123.5 6466  3834 A (0.41 0.18 0.18 0.17 0.06) *
##         21) roll_forearm>=123.5 3903  2595 C (0.079 0.18 0.34 0.23 0.18) *
##       11) magnet_dumbbell_y>=437.5 1930   968 B (0.035 0.5 0.04 0.23 0.19)
*
##    3) roll_belt>=130.5 1241     8 E (0.0064 0 0 0 0.99) *
```

```
# plot the classification tree
plot(modelRPART$finalModel,uniform=TRUE, main="Classification Tree")
text(modelRPART$finalModel,use.n=TRUE,all=TRUE,cex=0.8)
```



Estimate the performance of the RPART model on the traintest data

```r
set.seed(32343)
predictRPART <- predict(finModRPART, traintest, type = "class")
confusionMatrix(factor(traintest$classe), predictRPART)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1271   23   95    0    6
##          B  377  334  238    0    0
##          C  405   31  419    0    0
##          D  363  131  310    0    0
##          E  137  117  249    0  398
##
## Overall Statistics
##
##                Accuracy : 0.4939
##                  95% CI : (0.4798, 0.508)
##     No Information Rate : 0.5206
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.3385
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4978  0.52516  0.31960       NA  0.98515
## Specificity            0.9473  0.85590  0.87865   0.8361  0.88822
## Pos Pred Value         0.9111  0.35195  0.49006       NA  0.44173
## Neg Pred Value         0.6347  0.92364  0.77970       NA  0.99850
## Prevalence             0.5206  0.12969  0.26733   0.0000  0.08238
## Detection Rate         0.2592  0.06811  0.08544   0.0000  0.08116
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7226  0.69053  0.59913       NA  0.93669
```

Accuracy of RPART Classification model is: 49.39% Expected out-sample-error is: 50.61%

——————————————————————————-

2) Random Forest Classification Tree

```r
# Fit a Random Forest Model to the trainbuild data
# uses resampling with 5-fold cross-validation
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

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

set.seed(333)
modelRF <- train(classe ~ ., data = trainbuild, method = "rf", trControl = tr
ainControl(method = "cv", 5), ntree = 20)

modelRF

## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11775, 11774, 11775, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9862751  0.9826360
##   27    0.9898762  0.9871926
##   52    0.9820627  0.9773075
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.

finModRF<-modelRF$finalModel
print(finModRF)

##
## Call:
##  randomForest(x = x, y = y, ntree = 20, mtry = min(param$mtry,      ncol(x
)))
##                Type of random forest: classification
##                      Number of trees: 20
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 1.64%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4160   17    6    0    2 0.005973716
## B   39 2776   24    3    6 0.025280899
## C    3   28 2512   20    3 0.021044427
## D    5    5   42 2353    7 0.024461028
## E    1   13   10    8 2673 0.011829945
```

Random Forest OOB estimate of error rate is: 1.45%

Therefore, we test the Random Forest prediction model on the traintest data

```
predictRF <- predict(finModRF, traintest, type="class")
confusionMatrix(traintest$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1388    5    2    0    0
##          B    7  935    4    3    0
##          C    0   10  843    2    0
##          D    1    2   15  784    2
##          E    0    0    3    4  894
##
## Overall Statistics
##
##                Accuracy : 0.9878
##                  95% CI : (0.9843, 0.9907)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9845
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9821   0.9723   0.9887   0.9978
## Specificity            0.9980   0.9965   0.9970   0.9951   0.9983
## Pos Pred Value         0.9950   0.9852   0.9860   0.9751   0.9922
## Neg Pred Value         0.9977   0.9957   0.9941   0.9978   0.9995
## Prevalence             0.2847   0.1941   0.1768   0.1617   0.1827
## Detection Rate         0.2830   0.1907   0.1719   0.1599   0.1823
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9961   0.9893   0.9847   0.9919   0.9980
```

Accuracy of Random Forest Classification model is: 98.78% Expected out-of-sample error is: 1.22%

APPLY RANDOM FOREST MODEL TO TEST DATA

Make final prediction with RF model on testdata

```
predictRFfinal <- predict(finModRF, testdata, type="class")
predictRFfinal
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Evaluate the 'importance' of each variable on the model

```
importance(finModRF)
```

```
##                      MeanDecreaseGini
## roll_belt                  1461.22516
## pitch_belt                  661.11476
## yaw_belt                    828.28671
## total_accel_belt             41.36858
## gyros_belt_x                 35.43886
## gyros_belt_y                 56.96623
## gyros_belt_z                203.11309
## accel_belt_x                 40.03494
## accel_belt_y                 35.55609
## accel_belt_z                253.39294
## magnet_belt_x               183.45000
## magnet_belt_y               253.14815
## magnet_belt_z               235.59382
## roll_arm                    147.59884
## pitch_arm                    57.14357
## yaw_arm                     186.75462
## total_accel_arm              32.36093
## gyros_arm_x                  65.20687
## gyros_arm_y                  91.55732
## gyros_arm_z                  33.10616
## accel_arm_x                 125.79117
## accel_arm_y                  72.69814
## accel_arm_z                  50.89480
## magnet_arm_x                 99.07146
## magnet_arm_y                118.41316
## magnet_arm_z                106.91597
## roll_dumbbell               293.40659
## pitch_dumbbell               55.71394
## yaw_dumbbell                130.11622
## total_accel_dumbbell        290.22463
## gyros_dumbbell_x             66.67775
## gyros_dumbbell_y            107.49814
## gyros_dumbbell_z             33.49572
## accel_dumbbell_x            113.06308
## accel_dumbbell_y            364.72523
## accel_dumbbell_z            207.15682
## magnet_dumbbell_x           233.69454
## magnet_dumbbell_y           683.12160
```

```
## magnet_dumbbell_z          612.00190
## roll_forearm               729.29183
## pitch_forearm              917.66996
## yaw_forearm                125.61928
## total_accel_forearm         31.54066
## gyros_forearm_x             26.35514
## gyros_forearm_y             47.06111
## gyros_forearm_z             42.82381
## accel_forearm_x            307.05449
## accel_forearm_y             83.51370
## accel_forearm_z            171.72426
## magnet_forearm_x            79.01739
## magnet_forearm_y           114.58439
## magnet_forearm_z           292.34087
```