PML-Course-prediction-assignment

c zingler

07/08/2020

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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment. (so cited).

Introduction

Our data is already available for download. So lets begin first steps to set a local working directory for download of files. Then download them for further processing.

Read more: http:/groupware.les.inf.puc-rio.br/har#dataset#ixzz4TjwlLtt3

However lets note we are going to predict the classe variable in the data, classe is defined as follows:

- (Class A) task is done exactly according to the specification
- (Class B) task is done by throwing the elbows to the front
- (Class C) task is done by lifting the dumbbell only halfway
- (Class D) task is done by lowering the dumbbell only halfway
- (Class E) task is done by throwing the hips to the front
- Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Read more: http:/groupware.les.inf.puc-rio.br/har#dataset#ixzz4Tk05eLHD

Executive Summary

Our model building methodology is *Question* Input *Features* Algorithm *Prediction* Evaluation.

The resultant model is a highly accurate fit with greater than 99% accuracy and less than 1% out of sample error.

Our chosen model is Random Forest, as it best models the data's outcomes.

```
## setup environment
## getwd() ## *if necessary to know where the resultant files are stored.
## the project files will be created and stored here.

file.create("pml-training.csv")
```

```
## [1] TRUE
```

```
file.create("pml-testing.csv")
```

```
## [1] TRUE
```

```
## this R command will create new blank files - even if they already exist

download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv","pml-training.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv","pml-testing.csv" )
## this R command will populate the local file from the current internet files, even if they are
## not initialized

library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

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

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

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
## the caret package makes machine learning possible in R
## load data in data frames for processing

train_data <- read.csv("pml-training.csv")
test_data <- read.csv("pml-testing.csv")

## lets clean up the data starting with train_data
## str(train_data) ##*reqired if you want to look at the data structure


## Fix data that is empty("") or Div/0! or NA to zero(0)

train_data[train_data==""]<- NA
train_data[train_data=="#DIV/0!"]<- NA
train_data[is.na(train_data)]<- 0


## let us now remove the first 7 variables to make the data more anonymous
## and since it adds little value to prediction.
```

```r
train_data<-train_data[,-c(1:7)]

## str(train_data) ## required for inspecting data
## now lets do it for test_data
## str(test_data) ## required for inspecting data
## Fix data that is empty("") or Div/0! or NA to zero(0)

test_data[test_data==""]<- NA
test_data[test_data=="#DIV/0!"]<- NA
test_data[is.na(test_data)]<- 0

## let us now remove the first 7 variables to make the data more anonymous and it adds
## little value to prediction.

test_data<-test_data[,-c(1:7)]

## str(test_data) ## required for inspecting data

## lets set and check out the levels of the factor variable

train_data$classe <- as.factor(train_data$classe)
levels(train_data$classe)
```

```
## [1] "A" "B" "C" "D" "E"
```

```r
## this is unnecessary for the test_data as it does not have
## this factor variable - the test set is what we
## will do our prediction on and use this data set to predict this
##factor variable.
```

Prepare for the prediction phase

But first some more cleaning by culling variables that adds little value to making predictions with the resultant algorithm.

```r
## lets set up the sets by getting rid of the near zero variables
## that do not add value to the predictors.
##
dim(train_data)
```

```
## [1] 19622   153
```

```r
dim(test_data)
```

```
## [1]  20 153
```

```r
nzv <- nearZeroVar(train_data,saveMetrics = FALSE)
## so we now remove them from the training and testing data

train_data <- train_data[,-nzv]
test_data <- test_data[,-nzv]
## checking how many prediction variables we have after cleansing.
dim(train_data)
```

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

```r
dim(test_data)
```

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

```r
## we have managed to remove 100 unnecessary variables.
## so lets do the model building, but first lets set a seed to help with repeat runs
set.seed(54321)
```

Prediction modeling

I have decided to try both Random Forests (rf), and CART classification and regression trees (rpart) in a cross-validation control setting (cv).

```r
## partition the training data in to a training and validation
##set
inTrain <- createDataPartition(y=train_data$classe,p=0.6,list = FALSE)
train_data1 <- train_data[inTrain, ]
valid_data1 <- train_data[-inTrain, ]

## check data after split
dim(train_data1)
```

```
## [1] 11776    53
```

```r
dim(valid_data1)
```

```
## [1] 7846   53
```

```r
## so first lets set the cross validation control
control <- trainControl(method = "cv",number = 3, verboseIter = FALSE)

## Random forest first
fit_rf <- train(classe~.,method="rf",dat=train_data1, trControl=control)

fit_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.79%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    0    1    0    0 0.0002986858
## B   18 2246   14    1    0 0.0144800351
## C    0   11 2033   10    0 0.0102239533
## D    0    0   26 1901    3 0.0150259067
## E    0    0    3    6 2156 0.0041570439
```

```r
## now lets look at CART Model

fit_rpart <- train(classe~.,method="rpart",dat=train_data1,trControl=control)

fit_rpart$finalModel
```

```
## n= 11776
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 11776 8428 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 10789 7453 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -34.55 950    3 A (1 0.0032 0 0 0) *
##      5) pitch_forearm>=-34.55 9839 7450 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 8305 5964 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 124.5 5155 3051 A (0.41 0.18 0.18 0.17 0.06) *
##         21) roll_forearm>=124.5 3150 2113 C (0.075 0.18 0.33 0.23 0.19) *
##       11) magnet_dumbbell_y>=439.5 1534  749 B (0.031 0.51 0.048 0.22 0.19) *
##    3) roll_belt>=130.5 987   12 E (0.012 0 0 0 0.99) *
```

```
## now lets find out how accurate they are
```

Evalute models

By doing prediction on the validation set.

```
predrf <- predict(fit_rf,newdata = valid_data1)

predrpart <- predict(fit_rpart, newdata= valid_data1)

## so lets look at the confusion matrix of models

confusionMatrix(valid_data1$classe,predrf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2223    6    2    0    1
##          B   15 1501    2    0    0
##          C    0    9 1351    8    0
##          D    0    0   28 1257    1
##          E    0    1    1    1 1439
##
## Overall Statistics
##
##                Accuracy : 0.9904
##                  95% CI : (0.988, 0.9925)
##     No Information Rate : 0.2852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9879
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9933   0.9895   0.9762   0.9929   0.9986
## Specificity            0.9984   0.9973   0.9974   0.9956   0.9995
```

```
## Pos Pred Value         0.9960   0.9888   0.9876   0.9774   0.9979
## Neg Pred Value         0.9973   0.9975   0.9949   0.9986   0.9997
## Prevalence             0.2852   0.1933   0.1764   0.1614   0.1837
## Detection Rate         0.2833   0.1913   0.1722   0.1602   0.1834
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9958   0.9934   0.9868   0.9942   0.9991
```

```
confusionMatrix(valid_data1$classe,predrpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2030   33  167    0    2
##          B  668  501  349    0    0
##          C  650   34  684    0    0
##          D  573  235  478    0    0
##          E  216  192  378    0  656
##
## Overall Statistics
##
##                Accuracy : 0.4934
##                  95% CI : (0.4823, 0.5045)
##     No Information Rate : 0.5273
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3372
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4907  0.50352  0.33268       NA  0.99696
## Specificity            0.9455  0.85155  0.88187   0.8361  0.89065
## Pos Pred Value         0.9095  0.33004  0.50000       NA  0.45492
## Neg Pred Value         0.6247  0.92193  0.78821       NA  0.99969
## Prevalence             0.5273  0.12682  0.26204   0.0000  0.08386
## Detection Rate         0.2587  0.06385  0.08718   0.0000  0.08361
## Detection Prevalence   0.2845  0.19347  0.17436   0.1639  0.18379
## Balanced Accuracy      0.7181  0.67754  0.60728       NA  0.94381
```

```
## we see the Random forest model is far more accurate
## with an out of sample error < 1% which is quite low,
## and the Accuracy is quite high at > 99%
## the most important variables for this prediction are
varImp(fit_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          60.56
```

```
## yaw_belt               53.60
## magnet_dumbbell_z      46.18
## magnet_dumbbell_y      45.71
## roll_forearm           42.80
## pitch_belt             41.22
## accel_dumbbell_y       22.34
## accel_forearm_x        18.57
## roll_dumbbell          17.14
## magnet_dumbbell_x      16.97
## magnet_belt_z          15.40
## total_accel_dumbbell   14.62
## magnet_forearm_z       13.95
## accel_dumbbell_z       13.87
## magnet_belt_y          13.67
## accel_belt_z           13.44
## gyros_belt_z           11.26
## magnet_belt_x          10.48
## roll_arm               10.02
```

```r
## lastly we have to do a prediction on the supplied Test data
print(predict(fit_rf,newdata = test_data))
```

```
##  [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
```

Conclusion

we see the Random forest model is far more accurate with an out of sample error $< 1\%$ which is quite low, and the Accuracy is quite high at $> 99\%$

Further I applied the final generated classe variables to the prediction quiz and received a perfect score, implying the prediction model is excellent. We can in fact predict the manner in which the exercise was done by wearing these devices.