Coursera Machine Learning Project

Daniel Veit

5/2/2022

Introduction

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.

What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Analysis

Summary of Approach

1. Load the data.

2. Perform cross-validation to built a valid model, using 70% of the original data for model building (training data) and 30% of the data for testing (testing data).
3. Clean the data by excluding variables that are inexplainable or missing data.
4. Perform exploratory data analysis.
5. Perform Principle Component Analysis (PCA) to reduce the number of variables.
6. Apply Random Forest Method to build a model using the training data.
7. Validate the model using the testing data set.
8. Apply the model to estimate classes of 20 observations.

Load packages and libraries

```
suppressMessages(library(ggplot2))
suppressMessages(library(caret))
suppressMessages(library(randomForest))
```

1. Load the Data and Perform Exploratory Data Analysis

```
DataTraining <- read.csv("pml-training.csv")
DataTesting  <- read.csv("pml-testing.csv")
```

2. Cross validation

Use 70% of the original data for model building (training data) and 30% of the original data for testing (testing data)

```
Train <- createDataPartition(y=DataTraining$classe,p=.70,list=F)
Training <- DataTraining[Train,]
Testing <- DataTraining[-Train,]
```

3. Clean Training Data

Exclude variables with over 95% missing data:

```
Clean <- grep("name|timestamp|window|X", colnames(Training), value=F)
TrainingClean <- Training[,-Clean]
TrainingClean[TrainingClean==""] <- NA
RateNA <- apply(TrainingClean, 2, function(x) sum(is.na(x)))/nrow(TrainingClean)
TrainingClean <- TrainingClean[!(RateNA>0.95)]
```

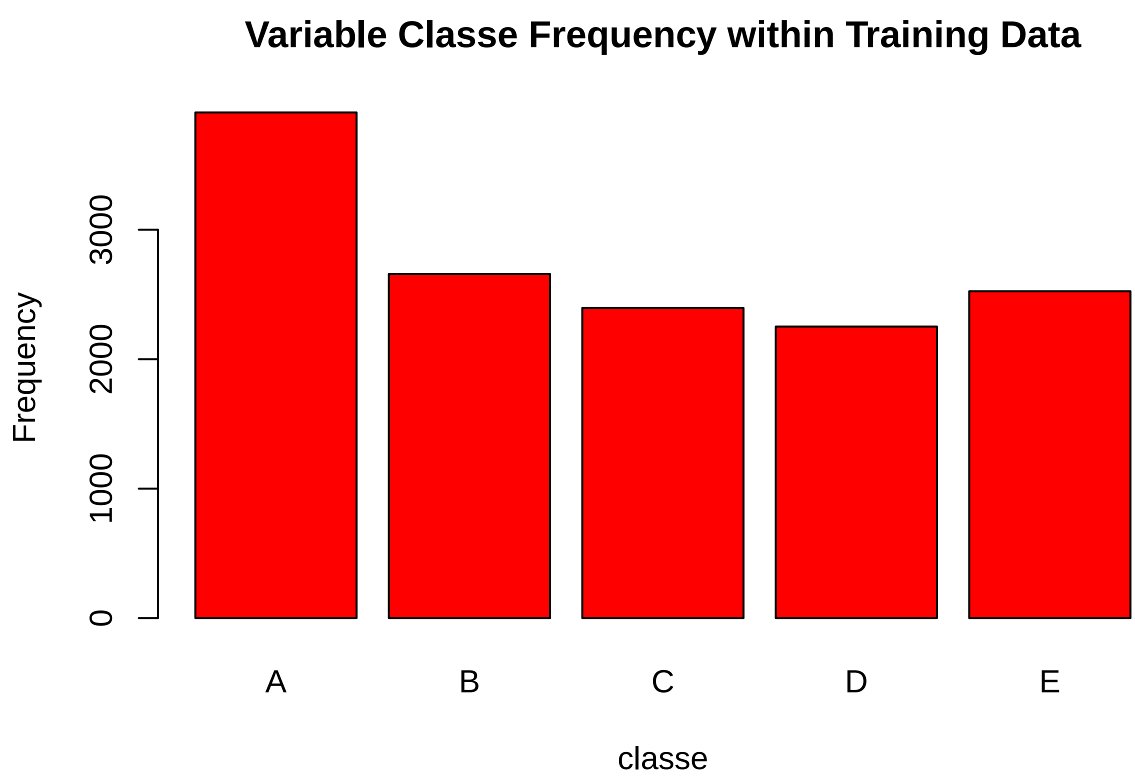
4. Perform Exploratory Analysis

Perform exploratory data analysis:

```
str(TrainingClean)
```

```
## 'data.frame':    13737 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.42 1.43 1.45 1.45 1.43 1.42 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.07 8.13 8.16 8.17 8.18 8.18 8.21 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.03 0.03 0.02 0.02 ...
##  $ gyros_belt_y        : num  0 0 0 0.02 0 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0 -0.02 -0.02 -0.02 ...
##  $ accel_belt_x        : int  -21 -22 -20 -21 -22 -20 -21 -21 -22 -22 ...
##  $ accel_belt_y        : int  4 4 5 2 4 2 4 2 2 4 ...
##  $ accel_belt_z        : int  22 22 23 24 21 24 22 23 23 21 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -2 1 -3 -5 -2 -8 ...
##  $ magnet_belt_y       : int  599 608 600 600 603 602 609 596 602 598 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -302 -313 -312 -308 -317 -319 -310 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 21.8 21.7 21.6 21.5 21.5 21.4 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.02 -0.03 -0.03 -0.03 -0.03 0 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0 0 -0.02 -0.02 0 0 -0.03 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -288 -288 -290 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 109 110 110 111 111 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -124 -122 -124 -123 -123 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -374 -372 -369 -376 -366 -363 -371 ...
##  $ magnet_arm_y        : int  337 337 344 337 338 341 334 339 343 331 ...
##  $ magnet_arm_z        : int  516 513 513 506 510 518 516 509 520 523 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 12.8 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.3 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -85.1 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0.02 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 0 0 0 0 0 0 -0.02 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -233 -234 -232 -235 -233 -233 -234 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 46 47 48 47 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -270 -272 -269 -270 -269 -270 -268 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -554 -555 -549 -558 -564 -554 -554 ...
##  $ magnet_dumbbell_y   : int  293 296 298 292 300 292 291 299 291 295 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -68 -74 -65 -69 -64 -65 -68 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28 27.8 27.7 27.7 27.6 27.5 27.2 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.8 -63.8 -63.9 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.03 0.02 0.02 0.02 0 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 0 -0.02 0 0 -0.02 0.02 -0.02 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 -0.02 0 -0.02 -0.02 -0.02 -0.03 -0.03 ...
##  $ accel_forearm_x     : int  192 192 196 189 193 193 190 193 191 193 ...
##  $ accel_forearm_y     : int  203 203 204 206 205 204 205 205 203 202 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -213 -214 -215 -214 -215 -214 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -17 -9 -16 -22 -17 -11 -14 ...
##  $ magnet_forearm_y    : num  654 661 658 655 660 653 656 657 657 659 ...
##  $ magnet_forearm_z    : num  476 473 469 473 474 476 473 465 478 478 ...
##  $ classe              : chr  "A" "A" "A" "A" ...
```

```
plot(as.factor(TrainingClean$classe), col="red", main="Variable Classe Frequency within Training Data",
```



Based on the plot above, level A has the highest frequency (exercise completed correctly) and the other levels are all within the same order of magnitude of each other.

5. Perform Principle Component Analysis (PCA)

Perform PCA because the number of variables is exceedingly high:

```
PCA <- preProcess(TrainingClean[,1:52],method="pca",pcaComp=25)
TrainingPCA <- predict(PCA,TrainingClean[,1:52])
```

6. Apply Random Forest Method

Apply RFM to build a model using the training data:

```
RandomForestModel <- randomForest(as.factor(TrainingClean$classe) ~ ., data=TrainingPCA, do.trace=F)
importance(RandomForestModel)
```

```
##      MeanDecreaseGini
## PC1          581.4697
## PC2          454.5204
## PC3          507.5636
## PC4          358.6645
```

```
## PC5          571.9466
## PC6          440.4496
## PC7          396.6138
## PC8          702.3663
## PC9          520.5887
## PC10         386.3968
## PC11         349.1841
## PC12         594.6479
## PC13         364.9058
## PC14         650.6507
## PC15         479.3919
## PC16         430.0413
## PC17         413.7959
## PC18         290.1533
## PC19         341.7186
## PC20         368.5004
## PC21         399.9761
## PC22         424.5340
## PC23         238.7951
## PC24         266.4694
## PC25         325.5110
```

7. Validate the Model

Validate the model using the testing data set, first by cleaning the data:

```
TestingClean <- Testing[,-Clean]
TestingClean[TestingClean==""] <- NA
RateNA <- apply(TestingClean, 2, function(x) sum(is.na(x)))/nrow(TestingClean)
TestingClean <- TestingClean[!(RateNA>0.95)]
```

Then performing the same PCA and RFM approach as the training data:

```
TestPCA <- predict(PCA,TestingClean[,1:52])
confusionMatrix(as.factor(TestingClean$classe),predict(RandomForestModel,TestPCA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1659    2   12    1    0
##          B   20 1104   10    0    5
##          C    2   10 1001   11    2
##          D    5    1   39  917    2
##          E    0    2    5    7 1068
##
## Overall Statistics
##
##                Accuracy : 0.9769
##                  95% CI : (0.9727, 0.9806)
##     No Information Rate : 0.2865
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9708
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9840   0.9866   0.9381   0.9797   0.9916
## Specificity            0.9964   0.9927   0.9948   0.9905   0.9971
## Pos Pred Value         0.9910   0.9693   0.9756   0.9512   0.9871
## Neg Pred Value         0.9936   0.9968   0.9864   0.9961   0.9981
## Prevalence             0.2865   0.1901   0.1813   0.1590   0.1830
## Detection Rate         0.2819   0.1876   0.1701   0.1558   0.1815
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9902   0.9896   0.9665   0.9851   0.9944
```

As can be seen by the results of the confusion matrix, the model has an accuracy of 97.5% and is therefore a sufficient model to predict accurate classes.

8. Apply the Model to Estimate Classes of 20 Observations

First by cleaning the data:

```
DataTestingClean <- DataTesting[,-Clean]
DataTestingClean[DataTestingClean==""] <- NA
RateNA <- apply(DataTestingClean, 2, function(x) sum(is.na(x)))/nrow(DataTestingClean)
```

Then performing the same PCA and RFM approach as the training data:

```
DataTestingClean <- DataTestingClean[!(RateNA>0.95)]
DataTestingPCA <- predict(PCA,DataTestingClean[,1:52])
DataTestingClean$classe <- predict(RandomForestModel,DataTestingPCA)
DataTestingClean$classe
```

```
##  [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
```

Conclusions

In this study, a total of 19622 observations from weight lifting exercise were used to analyze and predict correct body movement from others during the exercise. From the 19622 observations, 70% of the observations were used to build a model using the random forest method, while the remaining 30% of the observations were used for model validation (cross-validation). The results of the random forest model yielded an accuracy of 97% for the testing set, which was not used to build the initial model. The specificity was over 99% for all classes and the sensitivity varied between 93%-99%. Overall, the model is well developed to predict the exercise classes during weight lifting.