Activity Recognition Using Predictive Analytics

*Pratik Bhuwalka*

**Overview**

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. The aim of this project is to predict the manner in which participants perform a barbell lift. The data comes from http://groupware.les.inf.puc-rio.br/har wherein 6 participants were asked to perform the same set of exercises correctly and incorrectly with accelerometers placed on the belt, forearm, arm, and dumbell.

For the purpose of this project, the following steps would be followed:

1. Data Preprocessing

2. Exploratory Analysis

3. Prediction Model Selection

4. Predicting Test Set Output

**Data Preprocessing**

First, we load the training and testing set from the online sources and then split the training set further into training and test sets.

**library**(caret)

**setwd**("~/Projects/R/Coursera-Practical-Machine-Learning-Assignment-1/")

trainURL <- ["http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv" testURL <-](http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) ["http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"](http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

training <- **read.csv**(**url**(trainURL))

testing <- **read.csv**(**url**(testURL))

label <- **createDataPartition**(training$classe, p = 0.7, list = FALSE)

train <- training[label, ]

test <- training[-label, ]

From among 160 variables present in the dataset, some variables have nearly zero variance whereas some contain a lot of NA terms which need to be excluded from the dataset. Moreover, other 5 variables used for identification can also be removed.

NZV <- **nearZeroVar**(train) train <- train[ ,-NZV] test <- test[ ,-NZV]

label <- **apply**(train, 2, function(x) **mean**(**is.na**(x))) > 0.95 train <- train[, -**which**(label, label == FALSE)]

test <- test[, -**which**(label, label == FALSE)]

train <- train[ , -(1:5)]

test <- test[ , -(1:5)]

As a result of the preprocessing steps, we were able to reduce 160 variables to 54.

**Exploratory Analysis**

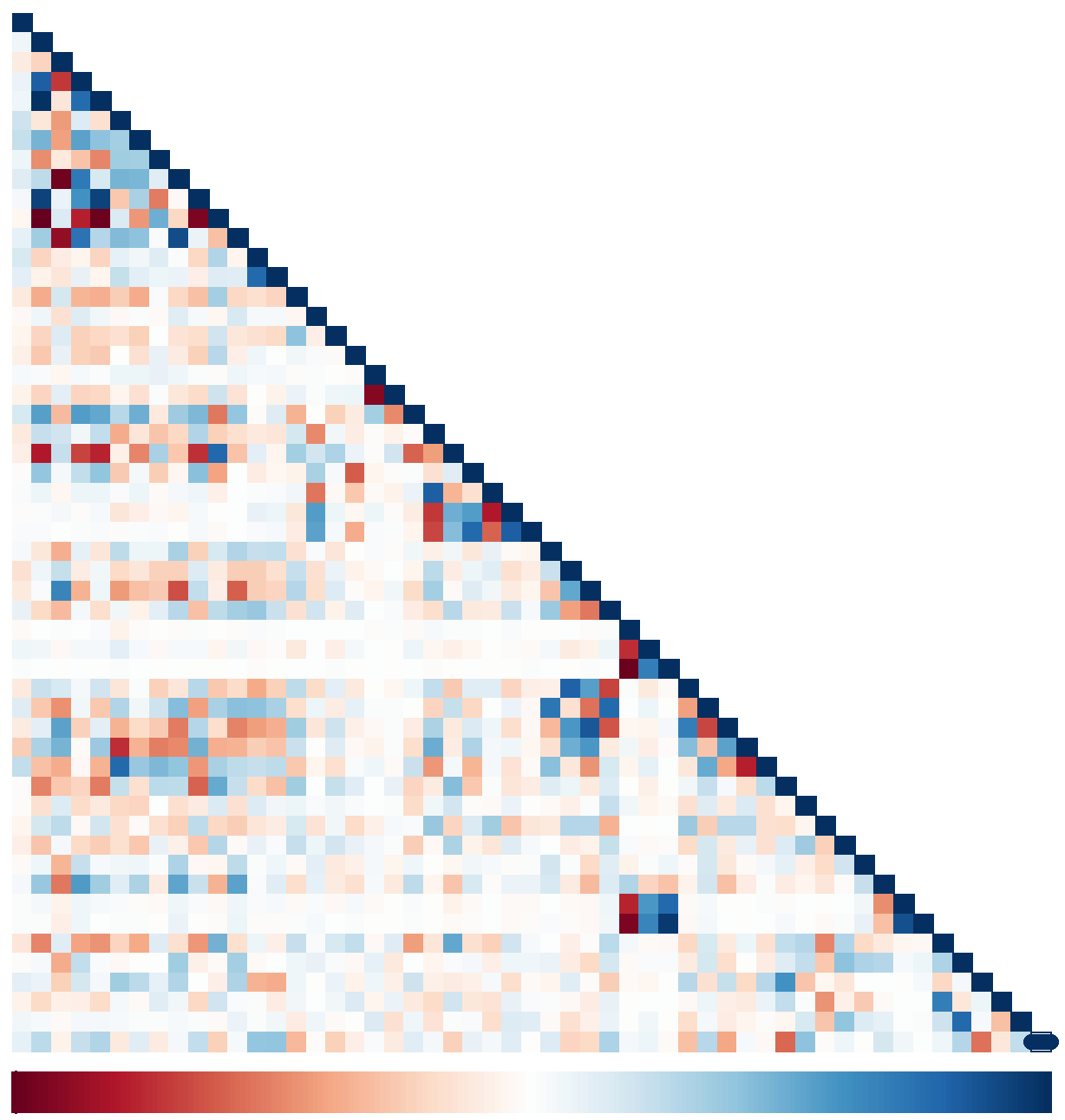
Now that we have cleaned the dataset off absolutely useless varibles, we shall look at the dependence of these variables on each other through a correlation plot.

**library**(corrplot)

corrMat <- **cor**(train[,-54])

**corrplot**(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = **rgb**(0,0,0))

num\_window roll\_belt pitch\_belt yaw\_belt total\_accel\_belt gyros\_belt\_x gyros\_belt\_y gyros\_belt\_z accel\_belt\_x accel\_belt\_y accel\_belt\_z magnet\_belt\_x magnet\_belt\_y magnet\_belt\_z roll\_arm pitch\_arm yaw\_arm total\_accel\_arm gyros\_arm\_x gyros\_arm\_y gyros\_arm\_z accel\_arm\_x accel\_arm\_y accel\_arm\_z magnet\_arm\_x magnet\_arm\_y magnet\_arm\_z roll\_dumbbell pitch\_dumbbell yaw\_dumbbell



num\_window roll\_belt

pitch\_belt

yaw\_belt total\_accel\_belt

gyros\_belt\_x

gyros\_belt\_y gyros\_belt\_z

accel\_belt\_x accel\_belt\_y

accel\_belt\_z magnet\_belt\_x

magnet\_belt\_y magnet\_belt\_z

roll\_arm pitch\_arm

yaw\_arm

total\_accel\_arm

total\_accel\_dumbbell gyros\_dumbbell\_x gyros\_dumbbell\_y gyros\_dumbbell\_z accel\_dumbbell\_x accel\_dumbbell\_y accel\_dumbbell\_z magnet\_dumbbell\_x magnet\_dumbbell\_y magnet\_dumbbell\_z roll\_forearm pitch\_forearm yaw\_forearm total\_accel\_forearm gyros\_forearm\_x gyros\_forearm\_y gyros\_forearm\_z accel\_forearm\_x accel\_forearm\_y accel\_forearm\_z magnet\_forearm\_x magnet\_forearm\_y magnet\_forearm\_z

−1 −0.8 −0.6 −0.4 −0.2 0 0.2 0.4 0.6 0.8 1

gyros\_arm\_x gyros\_arm\_y

gyros\_arm\_z

accel\_arm\_x accel\_arm\_y

accel\_arm\_z magnet\_arm\_x

magnet\_arm\_y magnet\_arm\_z

roll\_dumbbell pitch\_dumbbell

yaw\_dumbbell

total\_accel\_dumbbell gyros\_dumbbell\_x

gyros\_dumbbell\_y gyros\_dumbbell\_z

accel\_dumbbell\_x accel\_dumbbell\_y

accel\_dumbbell\_z magnet\_dumbbell\_x

magnet\_dumbbell\_y magnet\_dumbbell\_z

roll\_forearm pitch\_forearm

yaw\_forearm

total\_accel\_forearm gyros\_forearm\_x

gyros\_forearm\_y gyros\_forearm\_z

accel\_forearm\_x accel\_forearm\_y

accel\_forearm\_z magnet\_forearm\_x

magnet\_forearm\_y magnet\_forearm\_z

In the plot above, darker gradient correspond to having high correlation. A Principal Component Analysis can be run to further reduce the correlated variables but we aren’t doing that due to the number of correlations being quite few.

**Prediction Model Selection**

We will use 3 methods to model the training set and thereby choose the one having the best accuracy to predict the outcome variable in the testing set. The methods are Decision Tree, Random Forest and Generalized Boosted Model.

A confusion matrix plotted at the end of each model will help visualize the analysis better.

**Decision Tree**

**library**(rpart) **library**(rpart.plot) **library**(rattle)

**set.seed**(13908)

modelDT <- **rpart**(classe ~ ., data = train, method = "class")

**fancyRpartPlot**(modelDT)

*11*

A

.28 .19 .17 .16 .18

100%

*yyes* **rroll\_belt < 130** *no*

*22*

A

.31 .21 .19 .18 .11

92%

**pitch\_forearm < −34**

*55*

A

.24 .23 .21 .20 .12

84%

**num\_window >= 46**

*10*

A

.26 .24 .22 .20 .09

80%

**magnet\_dumbbell\_y < 440**

*20*

A

.29 .19 .25 .19 .09

68%

**rroll\_forearm < 124**

*21*

B

.03 .56 .04 .25 .11

12%

**total\_accel\_dumbbell >= 5.5**

*40*

A

.42 .19 .19 .16 .04

43%

**num\_window < 242**

*41*

C

.08 .18 .35 .23 .16

25%

**magnet\_dumbbell\_y < 292**

*42*

B

.05 .74 .06 .03 .12

8%

**rroll\_belt >= −0.6**

*81*

A

.30 .21 .25 .19 .05

32%

**magnet\_dumbbell\_z < −28**

*82*

C



.10 .14 .50 .15 .11

15%

**magnet\_forearm\_z < −245**

*83*

D

.06 .25 .11 .34 .24

10%

**accel\_forearm\_x >= −100**

*163*

C

.11 .22 .35 .26 .07

23%

**accel\_dumbbell\_y >= −40**

*165*

C

.03 .15 .55 .16 .11

14%

**num\_window < 88**

*166*

E

.05 .31 .16 .15 .33

7%

**rroll\_dumbbell < 40**

*326*

D

.12 .25 .25 .29 .08

19%

**rroll\_belt >= 126**

*652*

C

.00 .37 .59 .03 .01

5%

*653*

D

.16 .21 .15 .38 .10

15%

**pitch\_belt < −43 num\_window < 278**

*1306*

C

.00 .46 .54 .00 .00

2%

*1307*

D

.18 .17 .10 .43 .12

13%

**num\_window < 260 pitch\_belt < −42**

*44 80*

*162*

*1304 1305*

*2612 2613*

*2614 2615 327*

*164*

*330 331*

*332 333 167*

*84 85 43 11 33*

A A A

B C B C A

D C A

B C B

E D B

E D E E

.99 .01 .00 .00 .00 .77 .13 .00 .07 .03 .74 .19 .02 .05 .00 .00 .93 .00 .05 .02 .00 .03 .95 .02 .00 .00 1.00 .00 .00 .00 .00 .00 1.00 .00 .00 .40 .35 .16 .08 .02 .13 .12 .08 .52 .14 .00 .04 .92 .04 .00 .81 .06 .00 .05 .08 .15 .85 .00 .00 .00 .02 .08 .60 .18 .12 .05 .82 .01 .08 .04 .05 .19 .19 .16 .40 .08 .14 .04 .68 .07 .06 .81 .07 .03 .03 .00 .00 .00 .00 1.00 .00 .15 .00 .75 .10 .00 .00 .00 .20 .80 .01 .00 .00 .00 .99

8% 11%

10% 2%

3% 1%

1% 3%

10% 3% 1%

1% 13% 1%

5% 4%

7% 1%

4% 4% 8%

Rattle 2017−Aug−16 01:03:52 Yash\_Kumar\_Singh

predictDT <- **predict**(modelDT, test, type = "class") confMatDT <- **confusionMatrix**(predictDT, test$classe) confMatDT

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 1505 233 44 80 29

## B 39 609 36 21 25

## C 21 76 818 143 88

## D 86 145 51 612 131

## E 23 76 77 108 809

##

## Overall Statistics

##

## Accuracy : 0.7397

## 95% CI : (0.7283, 0.7509)

## No Information Rate : 0.2845

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.6696

## Mcnemar's Test P-Value : < 2.2e-16

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.8990 0.5347 0.7973 0.6349 0.747

## Specificity 0.9083 0.9745 0.9325 0.9161 0.9409

## Pos Pred Value 0.7959 0.8342 0.7138 0.5971 0.7402

## Neg Pred Value 0.9577 0.8972 0.9561 0.9276 0.9430

## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839

## Detection Rate 0.2557 0.1035 0.1390 0.1040 0.1375

## Detection Prevalence 0.3213 0.1240 0.1947 0.1742 0.1857

## Balanced Accuracy 0.9037 0.7546 0.8649 0.7755 0.8443

**Random Forest**

**library**(caret)

**set.seed**(13908)

control <- **trainControl**(method = "cv", number = 3, verboseIter=FALSE)

modelRF <- **train**(classe ~ ., data = train, method = "rf", trControl = control)

modelRF$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: 0.24%

## Confusion matrix:

## A B C D E class.error

## A 3904 1 0 0 1 0.0005120328

## B 7 2645 5 1 0 0.0048908954

## C 0 4 2392 0 0 0.0016694491

## D 0 0 8 2243 1 0.0039964476

## E 0 0 0 5 2520 0.0019801980

predictRF <- **predict**(modelRF, test)

confMatRF <- **confusionMatrix**(predictRF, test$classe)

confMatRF

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 1674 5 0 0 0

## B 0 1130 1 0 0

## C 0 3 1025 6 0

## D 0 1 0 958 3

## E 0 0 0 0 1079

##

## Overall Statistics

##

## Accuracy : 0.9968

## 95% CI : (0.995, 0.9981)

## No Information Rate : 0.2845

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.9959

## Mcnemar's Test P-Value : NA

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 1.0000 0.9921 0.9990 0.9938 0.9972

## Specificity 0.9988 0.9998 0.9981 0.9992 1.0000

## Pos Pred Value 0.9970 0.9991 0.9913 0.9958 1.0000

## Neg Pred Value 1.0000 0.9981 0.9998 0.9988 0.9994

## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839

## Detection Rate 0.2845 0.1920 0.1742 0.1628 0.1833

## Detection Prevalence 0.2853 0.1922 0.1757 0.1635 0.1833

## Balanced Accuracy 0.9994 0.9959 0.9986 0.9965 0.9986

**Generalized Boosted Model**

**library**(caret)

**set.seed**(13908)

control <- **trainControl**(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE) modelGBM <- **train**(classe ~ ., data = train, trControl = control, method = "gbm", verbose = FALSE) modelGBM$finalModel

## A gradient boosted model with multinomial loss function.

## 150 iterations were performed.

## There were 53 predictors of which 44 had non-zero influence.

predictGBM <- **predict**(modelGBM, test)

confMatGBM <- **confusionMatrix**(predictGBM, test$classe)

confMatGBM

## Confusion Matrix and Statistics

##

## Reference

## Prediction A B C D E

## A 1669 15 0 0 0

## B 5 1110 9 10 9

## C 0 13 1012 7 2

## D 0 1 5 945 13

## E 0 0 0 2 1058

##

## Overall Statistics

##

## Accuracy : 0.9845

## 95% CI : (0.981, 0.9875)

## No Information Rate : 0.2845

## P-Value [Acc > NIR] : < 2.2e-16

##

## Kappa : 0.9804

## Mcnemar's Test P-Value : NA

##

## Statistics by Class:

##

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.9970 0.9745 0.9864 0.9803 0.9778

## Specificity 0.9964 0.9930 0.9955 0.9961 0.9996

## Pos Pred Value 0.9911 0.9711 0.9787 0.9803 0.9981

## Neg Pred Value 0.9988 0.9939 0.9971 0.9961 0.9950

## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839

## Detection Rate 0.2836 0.1886 0.1720 0.1606 0.1798

## Detection Prevalence 0.2862 0.1942 0.1757 0.1638 0.1801

## Balanced Accuracy 0.9967 0.9838 0.9909 0.9882 0.9887

As Random Forest offers the maximum accuracy of 99.75%, we will go with Random Forest Model to predict our test data class variable.

**Predicting Test Set Output**

predictRF <- **predict**(modelRF, testing)

predictRF

## [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