Machine Learning - Course Project

Oliver Gonzalez

November 18, 2017

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 â€“ 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 Processing and Data cleaning

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.

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. We must Download and clean de data from division errors and empty strings replacing them by NA values.

Cross Validation & Data Partitioning

Now lets partition de data into a trainning (60%) and test (40%) sets. Also lets remove any NA and DIV/0 values.

```
train_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
test_url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
training_set<- read.csv(url(train_url), na.strings=c("NA","#DIV/0!",""))
testing_set <- read.csv(url(test_url), na.strings=c("NA","#DIV/0!",""))
inTrain <- createDataPartition(y=training_set$classe, p=0.60, list=FALSE)
myTraining <- training_set[inTrain, ]; myTesting <- training_set[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
## [1] 7846  160
removeNAcols   <- function(x) { x[ , colSums( is.na(x) ) < nrow(x) ] }
myTraining <- removeNAcols(myTraining)
myTesting  <- removeNAcols(myTesting)

complete   <- function(x) {x[,sapply(x, function(y) !any(is.na(y)))] }
incomplete <- function(x) {names( x[,sapply(x, function(y) any(is.na(y)))] )
}

trtr.na.var    <- incomplete(myTraining)
trts.na.var    <- incomplete(myTesting)
myTraining <- complete(myTraining)
myTesting  <- complete(myTesting)

#Lets remove the columns that are not predictors
myTraining_fset <- myTraining[,8:length(myTraining)]

#lets clear the variables with near zero variance
vNearZero <- nearZeroVar(myTraining_fset, saveMetrics = TRUE)
#Checking the result to see that there are zero values left
vNearZero
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.065934    8.76358696   FALSE FALSE
## pitch_belt            1.017544   13.83322011   FALSE FALSE
## yaw_belt              1.052632   14.55502717   FALSE FALSE
## total_accel_belt      1.081769    0.23777174   FALSE FALSE
## gyros_belt_x          1.087173    1.09544837   FALSE FALSE
## gyros_belt_y          1.150382    0.55197011   FALSE FALSE
## gyros_belt_z          1.088571    1.33322011   FALSE FALSE
## accel_belt_x          1.065359    1.30774457   FALSE FALSE
## accel_belt_y          1.150108    1.12941576   FALSE FALSE
## accel_belt_z          1.099222    2.42866848   FALSE FALSE
## magnet_belt_x         1.022624    2.53906250   FALSE FALSE
## magnet_belt_y         1.009709    2.39470109   FALSE FALSE
## magnet_belt_z         1.010638    3.59205163   FALSE FALSE
## roll_arm             52.897436   19.53974185   FALSE FALSE
## pitch_arm            76.444444   22.20618207   FALSE FALSE
## yaw_arm              31.738462   21.53532609   FALSE FALSE
## total_accel_arm       1.000000    0.55197011   FALSE FALSE
## gyros_arm_x           1.087542    5.30740489   FALSE FALSE
## gyros_arm_y           1.513072    3.07404891   FALSE FALSE
## gyros_arm_z           1.056604    1.94463315   FALSE FALSE
## accel_arm_x           1.138614    6.44531250   FALSE FALSE
## accel_arm_y           1.141732    4.41576087   FALSE FALSE
## accel_arm_z           1.118421    6.39436141   FALSE FALSE
## magnet_arm_x          1.037037   11.09035326   FALSE FALSE
## magnet_arm_y          1.054545    7.25203804   FALSE FALSE
## magnet_arm_z          1.115942   10.53838315   FALSE FALSE
## roll_dumbbell         1.025000   87.66983696   FALSE FALSE
## pitch_dumbbell        2.390244   85.60631793   FALSE FALSE
## yaw_dumbbell          1.123288   87.25373641   FALSE FALSE
## total_accel_dumbbell  1.075518    0.34816576   FALSE FALSE
## gyros_dumbbell_x      1.053521    1.95312500   FALSE FALSE
## gyros_dumbbell_y      1.309735    2.22486413   FALSE FALSE
## gyros_dumbbell_z      1.058997    1.63043478   FALSE FALSE
## accel_dumbbell_x      1.045918    3.42221467   FALSE FALSE
```

```
## accel_dumbbell_y      1.100671    3.83831522   FALSE FALSE
## accel_dumbbell_z      1.097222    3.40523098   FALSE FALSE
## magnet_dumbbell_x     1.155340    8.91644022   FALSE FALSE
## magnet_dumbbell_y     1.219048    6.93783967   FALSE FALSE
## magnet_dumbbell_z     1.072727    5.59612772   FALSE FALSE
## roll_forearm         12.392473   14.93716033   FALSE FALSE
## pitch_forearm        60.631579   20.95788043   FALSE FALSE
## yaw_forearm          15.151316   14.30027174   FALSE FALSE
## total_accel_forearm   1.202817    0.56046196   FALSE FALSE
## gyros_forearm_x       1.044728    2.30129076   FALSE FALSE
## gyros_forearm_y       1.021645    6.04619565   FALSE FALSE
## gyros_forearm_z       1.107143    2.41168478   FALSE FALSE
## accel_forearm_x       1.056604    6.58967391   FALSE FALSE
## accel_forearm_y       1.000000    8.18614130   FALSE FALSE
## accel_forearm_z       1.076923    4.64504076   FALSE FALSE
## magnet_forearm_x      1.037037   11.96501359   FALSE FALSE
## magnet_forearm_y      1.347826   15.20889946   FALSE FALSE
## magnet_forearm_z      1.081081   13.38315217   FALSE FALSE
## classe                1.469065    0.04245924   FALSE FALSE
```

Model Selection

The selected model for this task will be Random Forest because it generates an internal unbiased estimate of the generalization error as the forest building progresses, Random Forest works well with a mixture of numerical and categorical features

```
#The model chosen is the random forest method
keep <- names(myTraining_fset)
#fit model- RANDOM FOREST
set.seed(1235)

modFit <- randomForest(classe~., data = myTraining_fset)
print(modFit)
##
## Call:
##  randomForest(formula = classe ~ ., data = myTraining_fset)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.73%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3345    2    0    0    1 0.0008960573
## B   19 2252    8    0    0 0.0118473014
## C    0   12 2034    8    0 0.0097370983
## D    0    0   27 1902    1 0.0145077720
## E    0    0    2    6 2157 0.0036951501
qplot(roll_belt, magnet_dumbbell_y, colour=classe, data=myTraining_fset)
```

```
predict1 <- predict(modFit, myTesting, type = "class")
confusionMatrix(myTesting$classe, predict1)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    3    0    0    0
##          B    8 1509    1    0    0
##          C    0   15 1350    3    0
##          D    0    0   11 1273    2
##          E    0    0    1    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.9943
##                  95% CI : (0.9923, 0.9958)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9882   0.9905   0.9969   0.9986
## Specificity            0.9995   0.9986   0.9972   0.9980   0.9997
## Pos Pred Value         0.9987   0.9941   0.9868   0.9899   0.9986
## Neg Pred Value         0.9986   0.9972   0.9980   0.9994   0.9997
## Prevalence             0.2851   0.1946   0.1737   0.1628   0.1838
## Detection Rate         0.2841   0.1923   0.1721   0.1622   0.1835
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9979   0.9934   0.9938   0.9974   0.9992
predict_trainingset <- predict(modFit, myTraining, type = "class")
confusionMatrix(myTraining$classe, predict_trainingset)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Conclusion

As we can se from the results for the Training set, the random forest method is the best fit model and has been selected for the test data to submit the final results for this assignment as it shows an 100% accuracy.

```
predict_testset <- predict(modFit, testing_set, type = "class")
print(predict_testset)
##  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
```