ML_project

Martin

8/15/2020

Executive Summary

Based on a dataset provide by HAR http://groupware.les.inf.puc-rio.br/har, I will try to train a predictive model to predict what exercise was performed using a dataset with 159 features.

I will use xgboost to train the model here. The final model have an accurracy of nearly 100 % on the training set, about 99% accurary on validation set. It can predict all 20 cases correctly in the quiz when using the test set.

Downloading and reading data

```
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(trainURL, "training.csv")

testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testURL, "testing.csv")

training = read.csv("training.csv",na.strings=c("NA","#DIV/0!",""))
testing = read.csv("testing.csv",na.strings=c("NA","#DIV/0!",""))
```

Removing variables with NA values and variables that are not needed

```
n = NULL
for (i in names(training)){
    if(sum(is.na(training[,i]))/length(training[,i])<0.2){
        n= c(n,i)
    }
}
training2 <- training[,n]

rm = c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window"
rm = which(names(training2) %in% rm)
training3 = training2[,-rm]
training3$classe = factor(training3$classe)
```

Convert all into integers, expect classe

```
classeLevels <- levels(training2$classe)
training4 <- data.frame(data.matrix(training3))
training4$classe <- factor(training4$classe)
str(training4)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ 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    : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : num  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : num  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : num  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : num  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : num  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : num  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : num  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : num  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : num  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : num  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : num  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : num  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : num  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: num  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ 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 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : num  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : num  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : num  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : num  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : num  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : num  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : num  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : num  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : num  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : num  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Data Partitioning

splitting the data into training and validation sets. Test data will be our last data to predcit on.

```
ind = createDataPartition(training4$classe,p=0.8,list = FALSE)
traindat = training4[ind,]
validation = training4[-ind,]
```

Training

We will use K fold cross validation with k=5. We will fit a model using XGBoost and use all variables as possible predictors for classe. These modeling may take a while…

```
control <- trainControl(method="cv", 5, allowParallel = TRUE)
modelXGB <- train(classe ~ ., data=traindat, method="xgbTree", trControl=control)
modelXGB
```

```
## eXtreme Gradient Boosting
##
## 15699 samples
##    52 predictor
##     5 classes: '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12558, 12559, 12560, 12561, 12558
## Resampling results across tuning parameters:
##
##   eta  max_depth  colsample_bytree  subsample  nrounds  Accuracy   Kappa
##   0.3  1          0.6               0.50        50      0.8007516  0.7476521
##   0.3  1          0.6               0.50       100      0.8571254  0.8191130
##   0.3  1          0.6               0.50       150      0.8899301  0.8606188
##   0.3  1          0.6               0.75        50      0.7951468  0.7405415
##   0.3  1          0.6               0.75       100      0.8576364  0.8197888
##   0.3  1          0.6               0.75       150      0.8868093  0.8566746
##   0.3  1          0.6               1.00        50      0.7994788  0.7459445
##   0.3  1          0.6               1.00       100      0.8587821  0.8212219
##   0.3  1          0.6               1.00       150      0.8859179  0.8555476
##   0.3  1          0.8               0.50        50      0.8022171  0.7494763
##   0.3  1          0.8               0.50       100      0.8605654  0.8234282
##   0.3  1          0.8               0.50       150      0.8902489  0.8609998
##   0.3  1          0.8               0.75        50      0.8015813  0.7486400
##   0.3  1          0.8               0.75       100      0.8591010  0.8216134
##   0.3  1          0.8               0.75       150      0.8866188  0.8564299
##   0.3  1          0.8               1.00        50      0.7984600  0.7446760
##   0.3  1          0.8               1.00       100      0.8571268  0.8191468
##   0.3  1          0.8               1.00       150      0.8858544  0.8554790
##   0.3  2          0.6               0.50        50      0.9154743  0.8930604
##   0.3  2          0.6               0.50       100      0.9549022  0.9429393
##   0.3  2          0.6               0.50       150      0.9715270  0.9639781
##   0.3  2          0.6               0.75        50      0.9121613  0.8888438
##   0.3  2          0.6               0.75       100      0.9535643  0.9412488
##   0.3  2          0.6               0.75       150      0.9719734  0.9645442
##   0.3  2          0.6               1.00        50      0.9097423  0.8858102
##   0.3  2          0.6               1.00       100      0.9522904  0.9396508
##   0.3  2          0.6               1.00       150      0.9685969  0.9602743
```

```
##   0.3  2          0.8             0.50       50     0.9152185  0.8927263
##   0.3  2          0.8             0.50      100     0.9594883  0.9487451
##   0.3  2          0.8             0.50      150     0.9747123  0.9680088
##   0.3  2          0.8             0.75       50     0.9142642  0.8915325
##   0.3  2          0.8             0.75      100     0.9566216  0.9451244
##   0.3  2          0.8             0.75      150     0.9719734  0.9645438
##   0.3  2          0.8             1.00       50     0.9127991  0.8896803
##   0.3  2          0.8             1.00      100     0.9545198  0.9424643
##   0.3  2          0.8             1.00      150     0.9699985  0.9620463
##   0.3  3          0.6             0.50       50     0.9641380  0.9546285
##   0.3  3          0.6             0.50      100     0.9850948  0.9811451
##   0.3  3          0.6             0.50      150     0.9914008  0.9891225
##   0.3  3          0.6             0.75       50     0.9613988  0.9511642
##   0.3  3          0.6             0.75      100     0.9845216  0.9804197
##   0.3  3          0.6             0.75      150     0.9912735  0.9889613
##   0.3  3          0.6             1.00       50     0.9598699  0.9492371
##   0.3  3          0.6             1.00      100     0.9830561  0.9785673
##   0.3  3          0.6             1.00      150     0.9905727  0.9880752
##   0.3  3          0.8             0.50       50     0.9623545  0.9523768
##   0.3  3          0.8             0.50      100     0.9852218  0.9813062
##   0.3  3          0.8             0.50      150     0.9920377  0.9899286
##   0.3  3          0.8             0.75       50     0.9638197  0.9542320
##   0.3  3          0.8             0.75      100     0.9858588  0.9821108
##   0.3  3          0.8             0.75      150     0.9912098  0.9888817
##   0.3  3          0.8             1.00       50     0.9622269  0.9522113
##   0.3  3          0.8             1.00      100     0.9849034  0.9809033
##   0.3  3          0.8             1.00      150     0.9915283  0.9892842
##   0.4  1          0.6             0.50       50     0.8259773  0.7797382
##   0.4  1          0.6             0.50      100     0.8817144  0.8502160
##   0.4  1          0.6             0.50      150     0.9097405  0.8857359
##   0.4  1          0.6             0.75       50     0.8270599  0.7810960
##   0.4  1          0.6             0.75      100     0.8799926  0.8480098
##   0.4  1          0.6             0.75      150     0.9057908  0.8807268
##   0.4  1          0.6             1.00       50     0.8244487  0.7777284
##   0.4  1          0.6             1.00      100     0.8788477  0.8465867
##   0.4  1          0.6             1.00      150     0.9044536  0.8790125
##   0.4  1          0.8             0.50       50     0.8297354  0.7843995
##   0.4  1          0.8             0.50      100     0.8826688  0.8514311
##   0.4  1          0.8             0.50      150     0.9090397  0.8848514
##   0.4  1          0.8             0.75       50     0.8263606  0.7801716
##   0.4  1          0.8             0.75      100     0.8806953  0.8489578
##   0.4  1          0.8             0.75      150     0.9064282  0.8815316
##   0.4  1          0.8             1.00       50     0.8254686  0.7790887
##   0.4  1          0.8             1.00      100     0.8804400  0.8486352
##   0.4  1          0.8             1.00      150     0.9042624  0.8787897
##   0.4  2          0.6             0.50       50     0.9321617  0.9141573
##   0.4  2          0.6             0.50      100     0.9668133  0.9580111
##   0.4  2          0.6             0.50      150     0.9792984  0.9738142
##   0.4  2          0.6             0.75       50     0.9307615  0.9123987
##   0.4  2          0.6             0.75      100     0.9675782  0.9589834
##   0.4  2          0.6             0.75      150     0.9807636  0.9756658
##   0.4  2          0.6             1.00       50     0.9271938  0.9079131
##   0.4  2          0.6             1.00      100     0.9658581  0.9568057
##   0.4  2          0.6             1.00      150     0.9791710  0.9736511
```

```
##   0.4  2          0.8               0.50        50      0.9354750  0.9183552
##   0.4  2          0.8               0.50       100      0.9694890  0.9614018
##   0.4  2          0.8               0.50       150      0.9810181  0.9759886
##   0.4  2          0.8               0.75        50      0.9337547  0.9161873
##   0.4  2          0.8               0.75       100      0.9678331  0.9593065
##   0.4  2          0.8               0.75       150      0.9815911  0.9767142
##   0.4  2          0.8               1.00        50      0.9327349  0.9148918
##   0.4  2          0.8               1.00       100      0.9659854  0.9569642
##   0.4  2          0.8               1.00       150      0.9805722  0.9754257
##   0.4  3          0.6               0.50        50      0.9727379  0.9655145
##   0.4  3          0.6               0.50       100      0.9894901  0.9867068
##   0.4  3          0.6               0.50       150      0.9922926  0.9902508
##   0.4  3          0.6               0.75        50      0.9730558  0.9659150
##   0.4  3          0.6               0.75       100      0.9899993  0.9873503
##   0.4  3          0.6               0.75       150      0.9936941  0.9920238
##   0.4  3          0.6               1.00        50      0.9711450  0.9634966
##   0.4  3          0.6               1.00       100      0.9888531  0.9858995
##   0.4  3          0.6               1.00       150      0.9929297  0.9910568
##   0.4  3          0.8               0.50        50      0.9752856  0.9687334
##   0.4  3          0.8               0.50       100      0.9903820  0.9878344
##   0.4  3          0.8               0.50       150      0.9929935  0.9911378
##   0.4  3          0.8               0.75        50      0.9735650  0.9665585
##   0.4  3          0.8               0.75       100      0.9899356  0.9872693
##   0.4  3          0.8               0.75       150      0.9937576  0.9921039
##   0.4  3          0.8               1.00        50      0.9733739  0.9663213
##   0.4  3          0.8               1.00       100      0.9899357  0.9872700
##   0.4  3          0.8               1.00       150      0.9933116  0.9915401
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning parameter 'min_child_weight' was held constant at
##  a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 150, max_depth = 3, eta = 0.4, gamma = 0, colsampl
##  0.8, min_child_weight = 1 and subsample = 0.75.
```

Predictions and performance of model on train and validation data

```r
predict1 <- predict(modelXGB, traindat)
confusionMatrix(traindat$classe, predict1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4    5
##          1 4464    0    0    0    0
##          2    0 3038    0    0    0
##          3    0    0 2738    0    0
##          4    0    0    0 2573    0
##          5    0    0    0    0 2886
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 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: 1 Class: 2 Class: 3 Class: 4 Class: 5
## 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
```

```r
predict2 <- predict(modelXGB, validation)
confusionMatrix(validation$classe, predict2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2    3    4    5
##          1 1113    0    3    0    0
##          2    1  755    3    0    0
##          3    0    2  677    5    0
##          4    0    0    6  637    0
##          5    0    0    1    0  720
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9918, 0.9967)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9932
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            0.9991   0.9974   0.9812   0.9922   1.0000
## Specificity            0.9989   0.9987   0.9978   0.9982   0.9997
## Pos Pred Value         0.9973   0.9947   0.9898   0.9907   0.9986
## Neg Pred Value         0.9996   0.9994   0.9960   0.9985   1.0000
## Prevalence             0.2840   0.1930   0.1759   0.1637   0.1835
## Detection Rate         0.2837   0.1925   0.1726   0.1624   0.1835
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9990   0.9980   0.9895   0.9952   0.9998
```

We can see that model is doing well both on training and validation data. On training set it achieved moren than 99% accuracy and on validation accuracy is also more than 99%.

Predicting Test Data

In the first colum are the predictions for the given test set. 1=A , 2=B and so on.

```
predictest <- predict(modelXGB, testing)
testpred = cbind(predictest)
testpred
```

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