Practical Machine Learning Course Project

//

January 2016

The Weight Lifting Exercises Dataset Experiment

Six young health participants, were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

For data recording while performing the sets, the participants were wearing four 9 degrees of freedom Razor inertial measurement units (IMU), which provide three-axes acceleration, gyroscope and magnetometer. The sensors have been mounted in the users' glove, armband, lumbar belt and dumbbell.

The goal for the Weight Lifting Exercises experiement is to investigate "how (well)" an activity was performed by the wearer.

The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the data set.

Web site:
http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Data Source, the Weight Lifting Exercises Dataset:
http://groupware.les.inf.puc-rio.br/static/WLE/WearableComputing_weight_lifting_exercises_biceps_curl_variations.csv
(http://groupware.les.inf.puc-rio.br/static/WLE/WearableComputing_weight_lifting_exercises_biceps_curl_variations.csv)

Data Sources used for the Course Project:
The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data (for the quiz) are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Data Exploratory and Cleaning

```
#Reading the datasets, training and test
setwd("~/coursera/Practical Machine Learning")
#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")
training<-read.csv("./pml-training.csv",stringsAsFactors = T,na.strings=c("NA", "", "#DI
V/0!"))
quiz<-read.csv("./pml-testing.csv",stringsAsFactors = T, na.strings=c("NA", "", "#DIV/
0!"))

#dimension of the Weight Lifting Exercises Dataset
dim(training)
```

```
## [1] 19622   160
```

```
#Number of observations for each participant and exercice
table(training$user_name,training$classe)
```

```
##
##               A    B    C    D    E
##   adelmo   1165  776  750  515  686
##   carlitos  834  690  493  486  609
##   charles   899  745  539  642  711
##   eurico    865  592  489  582  542
##   jeremy   1177  489  652  522  562
##   pedro     640  505  499  469  497
```

The feature statistics (average, mn, max , var, etc…) are calculated for each window and have missing values for most of the observations, so they are removed from the analysis.
Note that for a specific participant and time of execution (timestamp variable) you know which exercise has been performed, also knowing the window number automatically tell you which type of exercise has been performed and by whom. Since the goal is to identify the type of exercise performed based on sensors measurements then the identification variables are also removed from the data set.

```
#Data cleaning
#Keep only non empty features in the testing file
allvar <- names(training)
vBeg <- c("kurtosis_","skewness_","max_","min_","amplitude_","var_","avg_","stddev_")
va <- vector()
for(i in 1:length(vBeg)){ va <- c(va,grep(vBeg[i],allvar,value = FALSE))}
training <- training[,-c(1:7,va)]
quiz <- quiz[,-c(1:7,va)]
```

```
#Load all the required libraries and suppression of waning messages
suppressWarnings(suppressMessages(library(caret, quietly=T)))
suppressWarnings(suppressMessages(library(rpart, quietly=T)))
suppressWarnings(suppressMessages(library(MASS, quietly=T)))
suppressWarnings(suppressMessages(library(kernlab, quietly=T)))
suppressWarnings(suppressMessages(library(gbm, quietly=T)))
suppressWarnings(suppressMessages(library(plyr, quietly=T)))
suppressWarnings(suppressMessages(library(randomForest, quietly=T)))
```

Split of the Weight Lifting Exercises Dataset in 2 parts, a simple splitting based on the outcome (classe). The first part (train) is used to explore de data and fit the models, the second (test) is used to evaluate the out of sample accuracy of each model.

```
# create training set indexes with 70% of data
set.seed(12345)
inTrain <- createDataPartition(y=training$classe,p=0.7, list=FALSE)
train <- training[inTrain,]
test <- training[-inTrain,]
dim(train)
```

```
## [1] 13737    53
```

```
dim(test)
```

```
## [1] 5885   53
```

Identification of highly correlated predictors, some models may benefit from reducing the level of correlation between the predictors. Only one predictor variable will be kept from the group of correlated variables. It will also significantly reduce the time execution to fit the models.

```
highlyCorDescr <- findCorrelation(cor(train[,-length(train)]), cutoff = .8)
train<-train[-highlyCorDescr[2:length(highlyCorDescr)]]
test<-test[-highlyCorDescr[2:length(highlyCorDescr)]]
quiz<-quiz[-highlyCorDescr[2:length(highlyCorDescr)]]
```

Use of the nearZeroVar function to identify the variables that have no variability, these variables are not useful when we want to construct a prediction model.

```
# print nearZeroVar table
near0<-nearZeroVar(train,saveMetrics=TRUE)
near0
```

```
##                      freqRatio percentUnique zeroVar   nzv
## yaw_belt              1.022161   12.92130742   FALSE FALSE
## total_accel_belt      1.054462    0.19654946   FALSE FALSE
## gyros_belt_x          1.033024    0.96090850   FALSE FALSE
## gyros_belt_y          1.129367    0.48773386   FALSE FALSE
## gyros_belt_z          1.085784    1.18657640   FALSE FALSE
## accel_belt_z          1.103110    2.13292568   FALSE FALSE
## magnet_belt_x         1.069231    2.18388294   FALSE FALSE
## magnet_belt_y         1.123894    2.08196841   FALSE FALSE
## magnet_belt_z         1.024845    3.18118949   FALSE FALSE
## roll_arm             55.488372   17.76224794   FALSE FALSE
## pitch_arm            91.807692   20.13540074   FALSE FALSE
## yaw_arm              33.138889   19.29096600   FALSE FALSE
## total_accel_arm       1.032206    0.47317464   FALSE FALSE
## gyros_arm_y           1.470588    2.66433719   FALSE FALSE
## gyros_arm_z           1.203390    1.70342870   FALSE FALSE
## accel_arm_y           1.163265    3.77083788   FALSE FALSE
## accel_arm_z           1.030612    5.58346073   FALSE FALSE
## magnet_arm_x          1.016129    9.55812768   FALSE FALSE
## magnet_arm_z          1.061728    9.13591032   FALSE FALSE
## roll_dumbbell         1.155556   87.00589648   FALSE FALSE
## pitch_dumbbell        2.048077   84.92392808   FALSE FALSE
## yaw_dumbbell          1.155556   86.36529082   FALSE FALSE
## total_accel_dumbbell  1.080559    0.31302322   FALSE FALSE
## gyros_dumbbell_y      1.207229    1.94365582   FALSE FALSE
## accel_dumbbell_y      1.023121    3.28310403   FALSE FALSE
## magnet_dumbbell_x     1.181818    7.84741938   FALSE FALSE
## magnet_dumbbell_y     1.333333    5.98383927   FALSE FALSE
## magnet_dumbbell_z     1.084034    4.78998326   FALSE FALSE
## roll_forearm         12.013100   13.59831113   FALSE FALSE
## pitch_forearm        62.477273   18.83235059   FALSE FALSE
## yaw_forearm          15.531073   12.88490937   FALSE FALSE
## total_accel_forearm   1.146991    0.50229308   FALSE FALSE
## gyros_forearm_x       1.016173    2.02373153   FALSE FALSE
## gyros_forearm_z       1.154762    2.15476450   FALSE FALSE
## accel_forearm_x       1.111111    5.67081604   FALSE FALSE
## accel_forearm_y       1.029412    7.11945840   FALSE FALSE
## accel_forearm_z       1.009174    4.08386111   FALSE FALSE
## magnet_forearm_x      1.075472   10.50447696   FALSE FALSE
## magnet_forearm_y      1.000000   13.27800830   FALSE FALSE
## magnet_forearm_z      1.095238   11.71289219   FALSE FALSE
## classe                1.469526    0.03639805   FALSE FALSE
```

```
dim(train)
```

```
## [1] 13737    41
```

No predictor has only one distinct value or has a near zero variance, so the 41 variables from that list are those who will be used to build our prediction model.

Methodology

For the course project, 5 machine learning classification methods will be compared:
* Classification Tree, CART (rpart)
* Linear Discriminant Analysis (lda)
* Least Squares Support Vector Machine with Radial Basis Function Kernel (lssvmRadial)
* Stochastic Gradient Boosting (gbm)
* Random Forest (rf)

The models are fitted on the training part of the data set (first split) using a cross validation procedure with k=10 folds, the default cross validation method of the caret train function. The out of sample error rate will be measured on the independent test data set (second split) for all the model.

The model with the best accuracy on the test data set will be the choosen one to answer the quiz test.

- Classification Tree, CART (rpart)

```
#rpart model
set.seed(12345)
modFitrpart <- train(classe ~ ., method = "rpart", data = train, trControl = trainControl
(method = "cv"))
```

```
#rpart model fit
print(modFitrpart)
```

```
## CART
##
## 13737 samples
##    40 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12361, 12364, 12363, 12364, 12364, 12365, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.01963178  0.5778536  0.4691779  0.02236006   0.02690560
##   0.02776930  0.5343899  0.4106954  0.02351234   0.03898975
##   0.03707659  0.3807065  0.1660515  0.10461144   0.17807823
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.01963178.
```

```
pred_rpart<-predict(modFitrpart,newdata=test[-length(test)])
#rpart out of sample accuracy
cmrpart<-confusionMatrix(pred_rpart, test$classe)
cmrpart$overall['Accuracy']
```

```
##  Accuracy
## 0.5682243
```

- Linear Discriminant Analysis (lda)

```
#modFitlda
set.seed(12345)
modFitlda <- train(classe ~ ., method = "lda", data = train, trControl = trainControl(met
hod = "cv"))
```

```
#modFitlda model fit
print(modFitlda)
```

```
## Linear Discriminant Analysis
##
## 13737 samples
##    40 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12361, 12364, 12363, 12364, 12364, 12365, ...
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.6575665  0.5670318  0.01068907   0.01348488
##
##
```

```
pred_lda<-predict(modFitlda,newdata=test[-length(test)])
#lda out of sample accuracy
cmlda<-confusionMatrix(pred_lda, test$classe)
cmlda$overall['Accuracy']
```

```
##  Accuracy
## 0.6484282
```

- Least Squares Support Vector Machine with Radial Basis Function Kernel (lssvmRadial)

```
#modFitsvm
set.seed(12345)
modFitsvm <- train(classe ~ ., method = "lssvmRadial", data = train, trControl = trainCon
trol(method = "cv"))
```

```
#modFitsvm model fit
print(modFitsvm)
```

```
## Least Squares Support Vector Machine with Radial Basis Function Kernel
##
## 13737 samples
##    40 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12361, 12364, 12363, 12364, 12364, 12365, ...
## Resampling results
##
##   Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.7991016  0.7451986  0.04042991   0.05144062
##
## Tuning parameter 'sigma' was held constant at a value of 0.01757531
##
```

```
pred_svm<-predict(modFitsvm,newdata=test[-length(test)])
#lssvmRadial out of sample accuracy
cmsvm<-confusionMatrix(pred_svm, test$classe)
cmsvm$overall['Accuracy']
```

```
##  Accuracy
## 0.8346644
```

- Stochastic Gradient Boosting (gbm)

```
#modFitgbm
set.seed(12345)
modFitgbm <- train(classe ~ ., method = "gbm", data = train, trControl = trainControl(met
hod = "cv"), verbose=FALSE)
```

```
#modFitgbm model fit
print(modFitgbm)
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    40 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12361, 12364, 12363, 12364, 12364, 12365, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD
##   1                   50      0.7248309  0.6507813  0.014716447
##   1                  100      0.7907137  0.7350301  0.009763385
##   1                  150      0.8313352  0.7865311  0.008739113
##   2                   50      0.8335167  0.7889911  0.010583483
##   2                  100      0.8909545  0.8619683  0.008412107
##   2                  150      0.9188346  0.8972863  0.007381244
##   3                   50      0.8788702  0.8466366  0.012351644
##   3                  100      0.9296079  0.9109173  0.008169796
##   3                  150      0.9502099  0.9369875  0.007566751
##   Kappa SD
##   0.018959554
##   0.012499172
##   0.011247408
##   0.013463758
##   0.010717224
##   0.009373944
##   0.015730235
##   0.010389390
##   0.009596822
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
pred_gbm<-predict(modFitgbm,newdata=test[-length(test)])
#gbm out of sample accuracy
cmgbm<-confusionMatrix(pred_gbm, test$classe)
cmgbm$overall['Accuracy']
```

```
##  Accuracy
## 0.9500425
```

- Random Forest (rf)

```
#modFitrf
set.seed(12345)
modFitrf <- train(classe ~ ., method = "rf", data = train, trControl = trainControl(metho
d = "cv"))
```

```
#modFitrf model fit
print(modFitrf)
```

```
## Random Forest
##
## 13737 samples
##    40 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12361, 12364, 12363, 12364, 12364, 12365, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9912632  0.9889469  0.002147138  0.002717251
##   21    0.9898814  0.9871988  0.002236130  0.002829110
##   40    0.9852949  0.9813951  0.002741541  0.003470233
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
pred_rf<-predict(modFitrf,newdata=test[-length(test)])
#rf out of sample accuracy
cmrf<-confusionMatrix(pred_rf, test$classe)
cmrf$overall['Accuracy']
```

```
##  Accuracy
## 0.9870858
```

```
#rf expected out of sample error
1-cmrf$overall['Accuracy']
```

```
##   Accuracy
## 0.01291419
```

The machine learning method with the best accuracy on the test data set is the random forest (0.9871). This is the one choose for the quiz prediction. The expected out of sample error of the random forest model is 1.3%.

```
#Quiz prediction with Random Forest Model
predict(modFitrf,newdata=quiz[-length(test)])
```

```
##  [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
```

The score obtained for the quizz is 20/20.