Practical Machine Learning Assignment

Dawid J Duvenhage

September 19, 2017

Conclusion

A reasonably sound “Random Forest Model”" is developed and successfully validated against 20 unknown observations.

Despite a significant amount of missing observations in the training and testing data a surprisingly accurate model is developed that predicts well against blind data.

A. 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).”

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 are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

B. Basic System Setup

B1. Loading libraries

B2. Setting up work directory, file paths, and downloading files

a. Set working directory

b. Important: Set correct file path

c. Create work folder to extract and write data to:

```
if(!file.exists("./assignment")){dir.create("./assignment")}
```

d. Setting file download url’s:

```
fileUrl_training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrl_testing <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

e. Download raw data files:

```
download.file(fileUrl_training,destfile="./assignment/pml-training.csv")
download.file(fileUrl_testing,destfile="./assignment/pml-testing.csv")
```

C. Data Processing, Cleanup, and Exploration

C1. Read the raw data.csv files from the folder location

This line of code will likely take a couple of minutes - be patient!

```
training_raw <- read.csv("./assignment/pml-training.csv")
```

C2. Initial data exploration

Inspecting training raw data file calling “str()” and “length(is.na(training_raw))” reveals several missing data sets in the form of ‘NA’, “#DIV/0”, and blanks as “” (only the first 15 lines of the raw data file is shown below).

```
object.size(training_raw)
```

```
## 19905864 bytes
```

```
str(training_raw, list.len=15)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323084232
 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ 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        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##   [list output truncated]
```

```
length(is.na(training_raw))
```

```
## [1] 3139520
```

```
summary(training_raw$classe)            #or: table(training$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
summary(training_raw$user_name)
```

```
##   adelmo carlitos  charles   eurico   jeremy    pedro
##     3892     3112     3536     3070     3402     2610
```

C3. Clean Data Step 1 - all missing data to 'NA'

To get clean workable "training and testing" data sets the files are reloaded and the "#DIV/0", and blanks as "", replaced as 'NA'.

Reloading the data will likely take a couple of minues - be patient!

```
training_clean <- read.csv("./assignment/pml-training.csv", na.strings=c("NA", "#DIV/0!", ""))
dim(training_clean)
```

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

```
#object.size(training_clean)
#class(training_clean)
#str(training_clean, list.len=15)

testing_clean <- read.csv("./assignment/pml-testing.csv", na.strings=c("NA", "#DIV/0!", ""))
dim(testing_clean)
```

```
## [1]  20 160
```

```
#object.size(testing_clean)
#class(testing_clean)
#str(testing_clean, list.len=15)
```

Peruse "classe" to predict:

```
prop.table(table(training_clean$user_name, training_clean$classe), 1)
```

```
##           
##                    A         B         C         D         E
##   adelmo   0.2993320 0.1993834 0.1927030 0.1323227 0.1762590
##   carlitos 0.2679949 0.2217224 0.1584190 0.1561697 0.1956941
##   charles  0.2542421 0.2106900 0.1524321 0.1815611 0.2010747
##   eurico   0.2817590 0.1928339 0.1592834 0.1895765 0.1765472
##   jeremy   0.3459730 0.1437390 0.1916520 0.1534392 0.1651969
##   pedro    0.2452107 0.1934866 0.1911877 0.1796935 0.1904215
```

C4. Clean Data Step 2 - Remove non-data columns

Rows 1 through 6 are non-data columns (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window), and therefor removed.

```
training_subsetA <- training_clean[, 7:160]
dim(training_subsetA)
```

```
## [1] 19622   154
```

```
testing_subsetA <- testing_clean[, 7:160]
dim(testing_subsetA)
```

```
## [1]  20 154
```

C5. Clean Data Step 3 - Remove ‘all-NA’ data columns

Remove all the columns containing “all” ‘NA’ data.

```
training_subset  <- training_subsetA[,colSums(is.na(training_subsetA))==0]
testing_subset <- testing_subsetA[,colSums(is.na(testing_subsetA))==0]
```

C6. Model Data Partitioning

Split the training data set into two data subsets using a 70/30 split of the data (one for “training” and one for “testing” purposes).

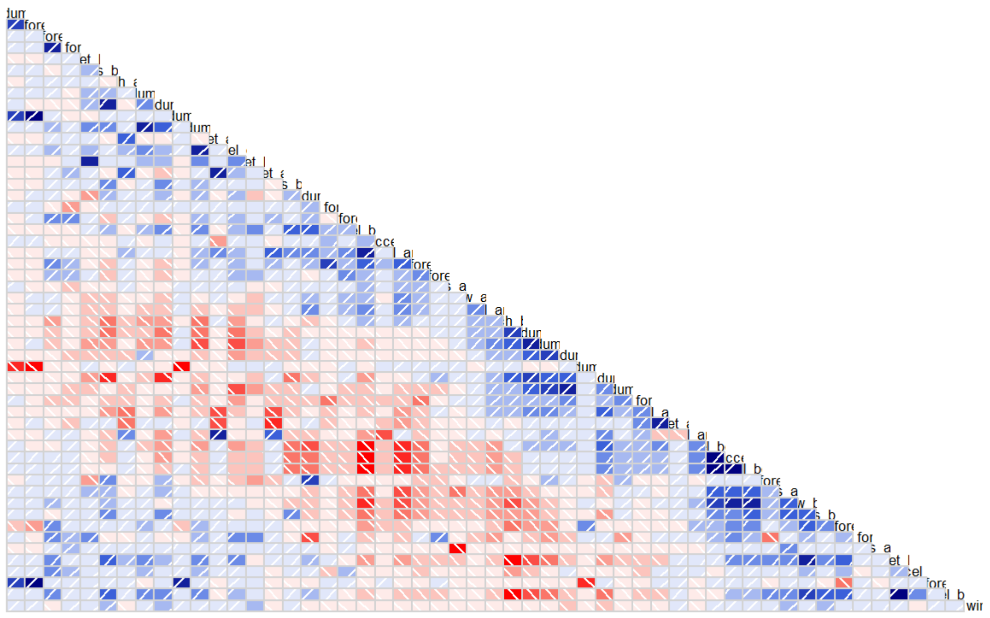
```
train_subset <- training_subset
#dim(train_subset)

test_subset <- testing_subset
#dim(test_subset)

inTrain <- createDataPartition(train_subset$classe, p = 0.70, list = F)

training <- train_subset[inTrain, ]
#dim(training)

testing <- train_subset[-inTrain, ]
#dim(testing)
```

C7. Exploratory Graph on cleaned training data

A Corrgram of “all” the variables in the cleaned training data set reveal a significant number of non-correlated variables (i.e. the red blocks with diagonal line from top left to lower right - the significance of each observation is highlighted by color tone, i.e. the darker the block the more uncorrelated). The blue blocks (with correlation line diagonally from lower left to upper right) indicate variables that do corrolate, with lighter blue blocks indicating lower significance and the darker blue suggesting variables having a stronger correlation.

```
corrgram(training, order=TRUE, lower.panel=panel.shade, upper.panel=NULL, main="Corrgram of Cleaned Training
data")
```

Corrgram of Cleaned Training data



D. Model Development, Selection, Testing, and Validation

D1. Model Development

Model development considers four model types including, a Support Vector Machine (svm), a Classical Decision Inference Tree (rpart), a Conditional Inference Tree (ctree), and a Random Forest (randomForest). Response variables are predicted for each model type.

Support Vector Machine modelFit

```
set.seed(1515151)
modelFit1 <- svm(classe~., data=training)
svm.pred <- predict(modelFit1, training, type="response")
SVMcm1 <- confusionMatrix(data=svm.pred, reference=training$classe, positive='yes')
#can also use: table(training$classe, gbm..pred, dnn=c("Actual", "Predicted"))
```

Classical Decision Inference Tree modelFit

```
set.seed(1515151)
modelFit2 <- rpart(classe~., data=training, method="class")
dtree.pred <- predict(modelFit2, training, type="class")
CDTcm2 <- confusionMatrix(data=dtree.pred, reference=training$classe, positive='yes')
```

Conditional Inference Tree modelFit

```
set.seed(1515151)
modelFit3 <- ctree(classe~., data=training)
ctree.pred <- predict(modelFit3, training, type="response")
CITcm3 <- confusionMatrix(data=ctree.pred, reference=training$classe, positive='yes')
```

Random Forest modelFit

```
set.seed(1515151)
modelFit4 <- randomForest(classe~., data=training, ntree=500, importance=T)
training$pr4 <- predict(modelFit4 ,training)
RFcm4 <- confusionMatrix(data=training$pr4, reference=training$classe, positive='yes')
```

****

D2. Model Selection

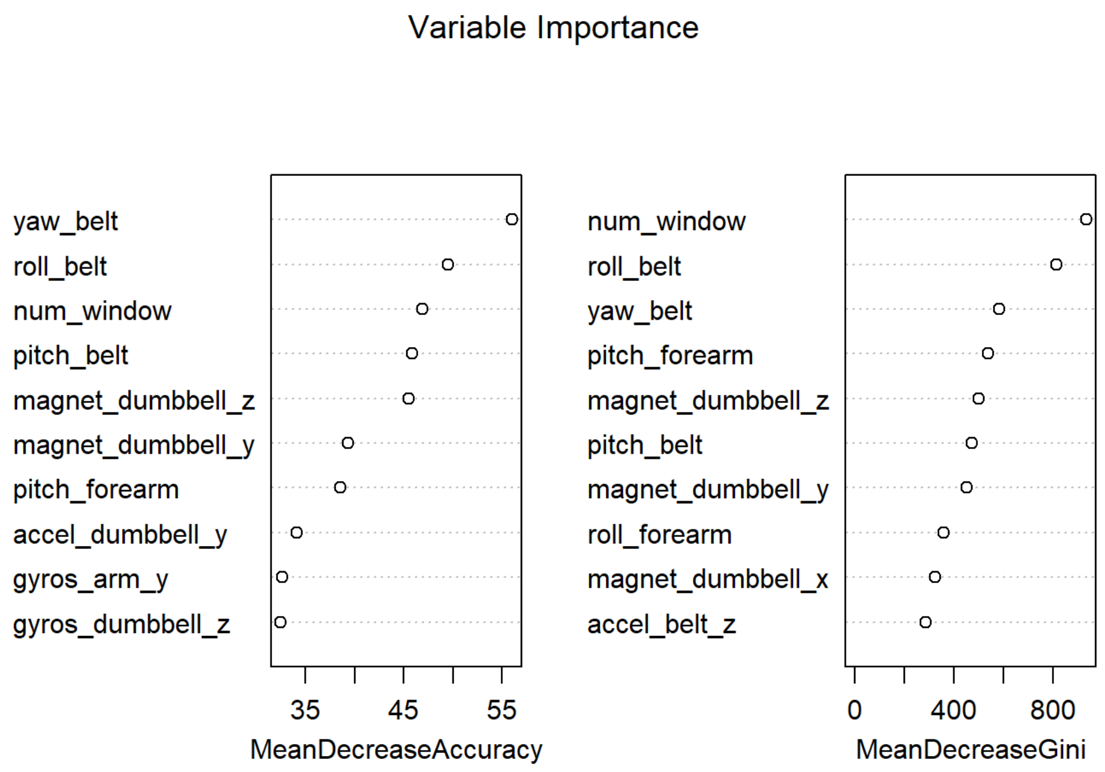
Using “all” of the data variables model accuracy is evaluated. It is clear from the parameters in the table below that the Random Forest fits the algorithm very well, and it is therefore selected for further evaluation, decreasing the number of model variables according to the Variable Importance plot shown below.

```
DevAcc <- data.frame(Model=c("Supp Vec Mach", "Clas Inf Tree", "Cond Inf Tree", "Rand Forest"),
                     Accuracy = (rbind(SVMcm1$overall[1], CDTcm2$overall[1],
                                       CITcm3$overall[1], RFcm4$overall[1])))

DevAcc
```

```
##            Model  Accuracy
## 1 Supp Vec Mach 0.9533377
## 2 Clas Inf Tree 0.8116765
## 3 Cond Inf Tree 0.9301885
## 4   Rand Forest 1.0000000
```

Variable importance plots are useful for evaluating Random Forest models, and ultimately to select model parameters. The plot, generated using the “varImpPlot” function, is used to look at and select variables based on “Model Mean Accuracy” and “Mean Gini values”. The 10 most significant variables are shown in the plot below.

```
varImpPlot(modelFit4, sort=T, main="Variable Importance",n.var=10)
```



```
varImp <- data.frame(importance(modelFit4, type=2))
varImp$Variables <- row.names(varImp)
varImp <- varImp[order(varImp$MeanDecreaseGini,decreasing = T ),]
varImp[1:10,]
```

```
##                   MeanDecreaseGini        Variables
## num_window                934.2824       num_window
## roll_belt                 813.9939        roll_belt
## yaw_belt                  580.9981         yaw_belt
## pitch_forearm             537.2506    pitch_forearm
## magnet_dumbbell_z         498.0012 magnet_dumbbell_z
## pitch_belt                472.7573       pitch_belt
## magnet_dumbbell_y         451.4721 magnet_dumbbell_y
## roll_forearm              358.5890     roll_forearm
## magnet_dumbbell_x         322.5380 magnet_dumbbell_x
## accel_belt_z              285.7875     accel_belt_z
```

From the variable important plot and table above the seven most significant variables, excluding “num_window”, i.e. roll_belt, yaw_belt, pitch_belt, magnet_dumbbell_z, pitch_forearm, magnet_dumbbell_y, and roll_forearm, are selected to build the final prediction model. The prediction accuracy for the adjusted Random Forest Model decreases slightly from 1.000 to 0.9992.

```
set.seed(1515151)
modelFit4_rerun <- randomForest(classe~ roll_belt + yaw_belt + pitch_belt + magnet_dumbbell_z + pitch_forearm + ma
gnet_dumbbell_y + roll_forearm , data=training, ntree=500, importance=T)

training$pr4_rerun <- predict(modelFit4_rerun ,training)
confusionMatrix(data=training$pr4_rerun, reference=training$classe, positive='yes')
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 3896    2    0    0    0
##          B    1 2655    1    0    0
##          C    8    1 2393    1    0
##          D    1    0    2 2251    0
##          E    0    0    0    0 2525
## 
## Overall Statistics
## 
##                Accuracy : 0.9988
##                  95% CI : (0.998, 0.9993)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9984
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9974   0.9989   0.9987   0.9996   1.0000
## Specificity            0.9998   0.9998   0.9991   0.9997   1.0000
## Pos Pred Value         0.9995   0.9992   0.9958   0.9987   1.0000
## Neg Pred Value         0.9990   0.9997   0.9997   0.9999   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2836   0.1933   0.1742   0.1639   0.1838
## Detection Prevalence   0.2838   0.1934   0.1749   0.1641   0.1838
## Balanced Accuracy      0.9986   0.9993   0.9989   0.9996   1.0000
```

****

D3. Model Testing

The redesigned Random Forest model, as expected, shows slightly lower accuracy when evaluated against the test data, but an acceptable out-of-sample error rate (i.e. [1 - accuracy]*100) is maintained.

```
testing$pr4 <- predict(modelFit4_rerun, testing)
RFcm4 <- confusionMatrix(data=testing$pr4, reference=testing$classe,positive='yes')

AccuracyResults <- data.frame(Model=c(RFcm4$overall[1]))
print(AccuracyResults)
```

```
##              Model
## Accuracy 0.9855565
```

```
ErroRate <- (1 - RFcm4$overall[1]) * 100
paste("out-of-sample error rate =", round(ErroRate, 2))
```

```
## [1] "out-of-sample error rate = 1.44"
```

****

D4. Model Validation

The final model (modelFit4_rerun) is validated against the “pml-testing.csv” data supplied (downloaded into the test_subset data frame). The model with 100 % accuracy predicts the 20 cases as supplied. The prediction results are printed below.

```
test_subset$pr4 <- predict(modelFit4_rerun, test_subset)
test_subset$pr4
```

```
##  [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
```