TITLE: Machine Learning Course Project: March 2015

A Predictive Model for Evaluating the Execution of a Unilateral Dumbbell Bicepts Curl

With Dataset from the Paper: "Qualitative Activity Recognition of Weight Lifting Exercises"

I. Executive Summary:

Using the Weight Lifting Exercises (WLE) dataset to predict whether or not a Unilateral Dumbbell Biceps Curl was performed according to an established standard(see II. for paper citation), three random forest classification models were developed and compared: (1) 54 predictor variables, (2) 30 variables and (3) 2 variables. In conclusion, The 2-variable model accuracy and out of sample error measures were about as good as for the 54 variable model while reducing overfitting issues apparent with the latter, and the 2 variable model predicted the exercise class in the final 20 test case set 100% accurately. The 2 classification variables are: (1) num _ window and (2) row_belt, found from cross validation and GINI variable importance criterion. Extremely accurate prediction results for even a 2 variable model indicate some degree of dataset dependency (or overfitting) may still exist. If the final model was used to predict exercise classe on data derived from a new independent experiment using different subjects and sensor measurements, it may not be as robust nor the predictions as accurate.

II. Introduction: Data Source and Project Questions:

The Weight Lifting Exercises (WLE) dataset contains body sensor accelerometer data from six males (ages 20-28) who performed 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different ways(i.e., classes): 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).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Paper Citation: *Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.*

WLE Dataset documentation: http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf

Project Goals:

1. Build and crossvalidate a predictive model for evaluating the execution of a Unilateral Dumbbell Biceps Curl, using class of exercise as the predicted value. Discuss model selection process and quantify uncertainty (out of sample error).

2. Use the model to predict class of exercise for a test set of 20 subjects.

III. Data Analysis and Model Building: The following steps were taken:

Step 1. Download Raw Dataset, Perform Basic Exploratory Data Analysis and Create Tidy Dataset

This code reads in the csv file assuming it has been downloaded to a local directory and displays the dimensions and basic structure of the raw dataset. The raw dataset contains 160 columns. After removing all columns which have NA or missing values and verifying only the rows with complete cases are retained (ie, sum of not complete cases=0), and the timestamp columns removed (cols 1,3-5) the tidy dataset mydf3 has 55 columns and the same number of rows as the raw data.

```
setwd("~/Actuarial/Coursera/Data Science/Machine Learning/Assignment")

mydf<-read.csv(file="pml-training.csv")
str(mydf)
```

```
## '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 ...
##  $ 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 ...
##  $ 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 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1
1 1 1 1 ...
```

```
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1
1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21
...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609
...
##  $ magnet_belt_z           : int  -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         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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             : int  -288 -290 -289 -289 -289 -289 -289 -289
-288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110
...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124
-122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372
-369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334
...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516
...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-
0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-
0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1
```

```
1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-
0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-
0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1
1 1 1 1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1
1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
# Count number of NA values for each column, keep only the columns with zero
NA values
dfNA<-data.frame(x=1:ncol(mydf),y=1)
for(i in 1:ncol(mydf)){
  dfNA[i,2]<-sum(is.na(mydf[,i]))
}

mydf2<-data.frame(mydf[,which(dfNA[,2]==0)])
dfMISS<-data.frame(x=1:ncol(mydf2),y=1)
for(i in 1:ncol(mydf2)){
  dfMISS[i,2]<-sum(ifelse(mydf2[,i]=="",1,0))
}
mydf3<-data.frame(mydf2[,which(dfMISS[,2]==0)])
row.names(mydf3)<-NULL

# Remove time stamp columns, keep num_window which captures the time stamp in
buckets
# This cleaning step brings mydf3 down to only 54 predictors and the classe
column
mydf3<-mydf3[,-1]
mydf3<-mydf3[,-2:-5]
dim(mydf3)
```

```
## [1] 19622    55
```

```
# Cleaned final dataset contains no incomplete cases/rows
cc<-complete.cases(mydf3)
sum(!cc)
```

```
## [1] 0
```

Step 2. Partition Tidy dataset and create 3 new datasets via random samples, training,testing, cross validation Random sampling creates 3 new datasets: 60% training, 20% testing and 20% for cross validation

```
library(caret)
set.seed(42)
inTrain<-createDataPartition(y=mydf3$classe,p=0.60,list=FALSE)
training<-mydf3[inTrain,]
testing1<-mydf3[-inTrain,]
inCV<-createDataPartition(y=testing1$classe,p=0.50,list=FALSE)
crossv<-testing1[inCV,]
testing<-testing1[-inCV,]
```

Step 3. Create first random Forest model on training data with all 54 predictors using randomForest package, predict the classe outcomes using testing data Why Random Forest modeling? Compared with other types of modeling such as single tree CART or GLM, the algorithms are known for superior predictive accuracy, and are better for non-linear multi-variate models. Disadvantages include difficulty of interpretability and overfitting. Overfitting issues were identified and addressed in the cross validation code in step 3.5. The randomForest package was selected due to concerns about long execution times and the large size of the datasets as compared to the caret package.

```
library(randomForest)
set.seed(42)
modFit.54<-randomForest(data=training,x=training[,-
55],y=training$classe,ntree=100,mtry=3,importance=TRUE,proximity=TRUE)

#Predict using testing data
set.seed(42)
predict.54<-
predict(modFit.54,newdata=testing,type="response",norm.votes=TRUE,predict.all
=FALSE,proximity=FALSE,nodes=FALSE)
table(testing$classe,predict.54)

##    predict.54
##        A    B    C    D    E
##   A 1116    0    0    0    0
##   B    1  758    0    0    0
##   C    0    4  680    0    0
##   D    0    0    8  634    1
##   E    0    0    0    4  717
```

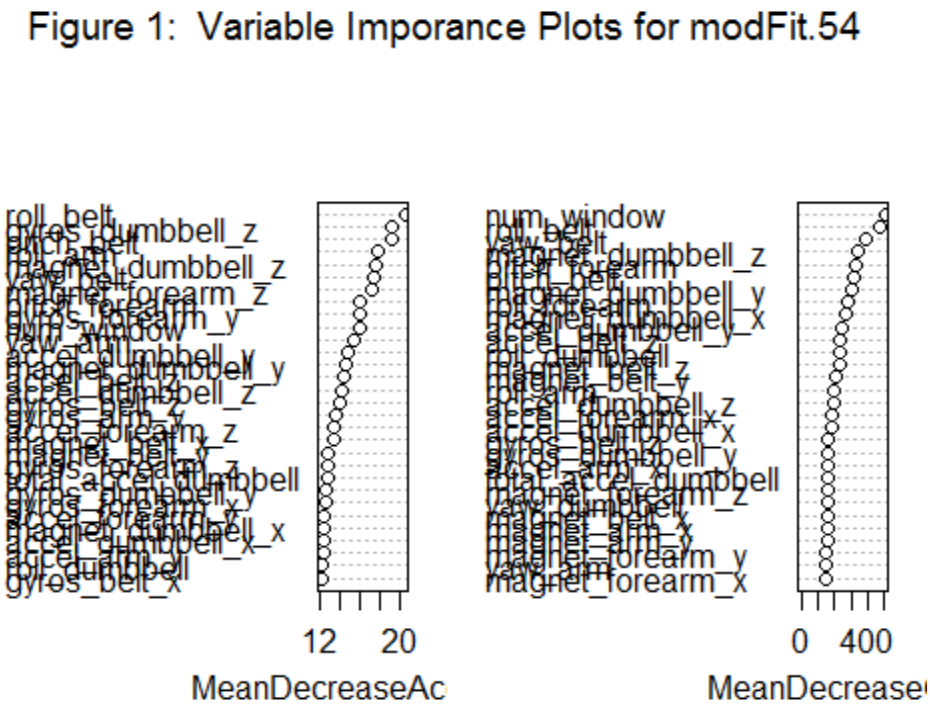
Step 3.5: Perform Cross Validation on the 54 predictor model training data The rfcv function was used for cross validation. It creates many models, starting with 54 predictors and building smaller and smaller models with the n=# predictors variable reduced by step = 0.7. The first n=54, the second n=54x0.7=38 and so on down to n=1. In conclusion, the cross-validation error only slightly increases as the number of predictors are reduced. The error difference between using 54 predictors and using 2 predictors is quite low (<.02) which suggests a 2 predictor model might be about as good as the 54 predictor model.

```
#See http://stats.stackexchange.com/questions/112556/r-random-forest-need-help-understanding-the-rfcv-function
rfcv.54 <- rfcv(trainx = training[,-55], trainy = training[,55], scale = "log", step=0.7)
rfcv.54$error.cv
```

```
##          54          38          26          19          13           9
## 0.004415761 0.004161005 0.004076087 0.004330842 0.003311821 0.002292799
##           6           4           3           2           1
## 0.002547554 0.002547554 0.015964674 0.014521060 0.481997283
```

Step 4. Based on the 54 predictor model, Assess Variable Importance Using the varimpPlot function, the two plots (see **Figure 1**) show the Mean Decrease Accuracy measure on the left plot and the Mean Decrease GINI measure in the right plot. The variables on the y-axis are ranked from most to least important from top to bottom. There are 30 variables shown in the plot out of 54.

```
library(randomForest)
set.seed(42)
varImpPlot(modFit.54,sort=TRUE,main="Figure 1:  Variable Imporance Plots for modFit.54")
```




```
impmodFit.v4<-importance(modFit.54)
impmodFit<-impmodFit.v4
#write.csv(impmodFit.v4,"impmodFit-v4.csv")
```

Step 5. Create second random Forest model on testing data with the top 30 MDA predictors using randomForest package, predict the classe outcomes using cross validation data From the variable importance cross validation in Step 4 there are 30 variables with highest importance. This code shows the predictions of classe in a confusion matrix form, against the true values of classe in the cross validation data. There still appears to be significant overfitting with 30 variables.

```
testorder<-order(-impmodFit[,6])
testorder.30<-testorder[1:30]
mydf.30<-testing[,testorder.30]
mydf5<-cbind(mydf.30,testing$classe)
colnames(mydf5)[31]<-"classe"
set.seed(42)
modFit.30<-randomForest(data=mydf5,x=mydf5[,-
31],y=mydf5$classe,ntree=100,mtry=3,importance=TRUE,proximity=TRUE)

#Predict using crossv data
set.seed(42)
predict.30<-
predict(modFit.30,newdata=crossv,type="response",norm.votes=TRUE,predict.all=
FALSE,proximity=FALSE,nodes=FALSE)
table(crossv$classe,predict.30)
```

```
##    predict.30
##        A    B    C    D    E
##   A 1111    4    0    1    0
##   B   20  726   12    0    1
##   C    0    4  678    2    0
##   D    1    0   19  623    0
##   E    0    0    2    5  714
```

Step 6. Create final random Forest model on testing data with the top 2 GINI predictors using the caret package, predict the classe outcomes using cross validation data From the cross validation in Step 3.5 and the variable importance exercise in step 4, the top 2 Mean Decrease GINI variables have a clear break from the other 28 variables. The next code chunk gives predictions of classe of the top 2 variables in a confusion matrix form, against the true values of classe in the cross validation data. Perhaps surprisingly, the predictions are nearly as accurate as the 54 or 30 variable models.

Figure 2 provides a scatter plot of the 2 variables, colored by classe type. The plot indicates strong grouping of the classe variables A,B,C,D,E for the upper value of "row _ belt'. Especially for classe E. This provides general, visual evidence that the row_belt variable seems to have high importance in classifying the outcome variable.

```
# Create model for 2 variable GINI
testorder<-order(-impmodFit[,7])
testorder.2G<-testorder[1:2]
mydf.2G<-testing[,testorder.2G]
mydf6<-cbind(mydf.2G,testing$classe)
```

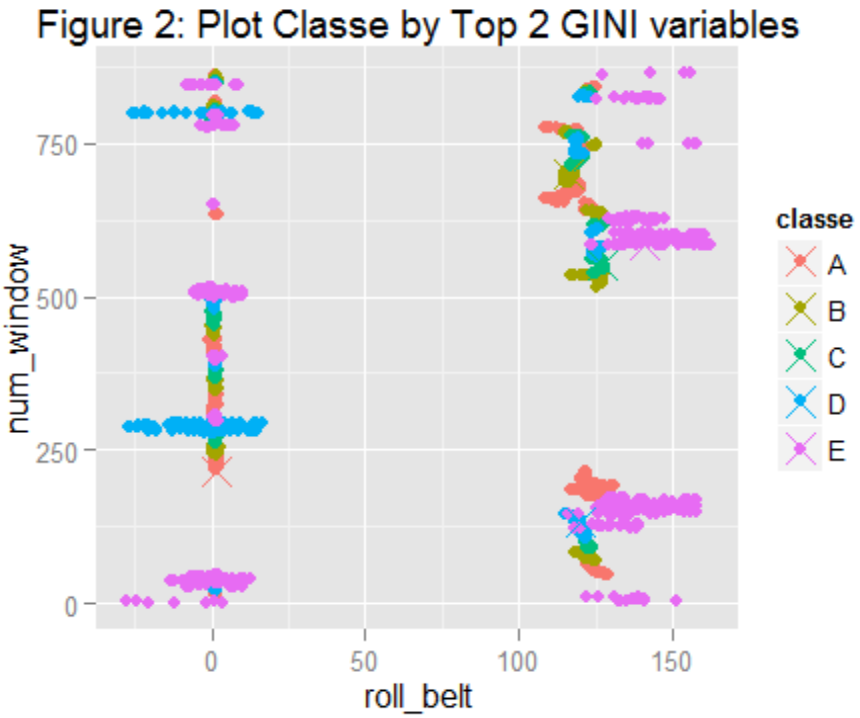
```r
colnames(mydf6)[3]<-"classe"
set.seed(42)
modFit.2G<-randomForest(data=mydf6,x=mydf6[,-
3],y=mydf6$classe,ntree=100,importance=TRUE,proximity=TRUE)

#Predict using crossv data 2 var MDA
set.seed(42)
predict.2G<-
predict(modFit.2G,newdata=crossv,type="response",norm.votes=TRUE,predict.all=
FALSE,proximity=FALSE,nodes=FALSE)
table(crossv$classe,predict.2G)
```

```
##    predict.2G
##        A    B    C    D    E
##   A 1110    1    4    1    0
##   B    8  746    3    0    2
##   C    0    6  674    1    3
##   D    0    5    5  633    0
##   E    2    3    1    0  715
```

```r
# Plot classe against 2 variables GINI
library(caret)
set.seed(42)
modFit.caret.2G <- train(classe~ .,data=mydf6,method="rf",prox=TRUE)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the
grid to 1 .
```

```r
modFit.2.PlotG <- classCenter(mydf6[,c(1,2)], mydf6$classe,
modFit.caret.2G$finalModel$prox)
modFit.2.PlotG <- as.data.frame(modFit.2.PlotG); modFit.2.PlotG$classe <-
rownames(modFit.2.PlotG)
p <- qplot(roll_belt,num_window, col=classe,data=mydf6,main="Figure 2: Plot
Classe by Top 2 GINI variables")
p +
geom_point(aes(x=roll_belt,y=num_window,col=classe),size=5,shape=4,data=modFi
t.2.PlotG)
```



IV. Results: Comparison of 3 Models, Cross Validation and Out of Sample Error

Final Confusion Matrices for the 3 models Figure 3 gives the confusion matrices for the 54, 30 and 2 variable models. Each row indicates the "true" class of the outcome and the columns show the predicted values. On the diagonal are the cases where predicted values matched the true values.

```
#install.packages("gridExtra")
library(gridExtra)
plot54<-table(crossv$classe,predict.54)
#grid.table(plot54)
plot30<-table(crossv$classe,predict.30)
#grid.table(plot30)
plot2G<-table(crossv$classe,predict.2G)
#grid.table(plot2G)

g1<-tableGrob(plot54)
g2<-tableGrob(plot30)
g3<-tableGrob(plot2G)
#png("imageALL.png")
grid.arrange(g1,g2,g3, ncol=1, main = "Figure 3:Confusion Matrices for 3 RF
Models:54(top) 30(mid) 2(bottom)")
```

Figure 3:Confusion Matrices for 3 RF Models:54(top) 30(mid) 2(bottom

A	1116	0	0	0	0
B	1	758	0	0	0
C	0	4	680	0	0
D	0	0	8	634	1

A	1111	4	0	1	0
B	20	726	12	0	1
C	0	4	678	2	0
D	1	0	19	623	0

A	1110	1	4	1	0
B	8	746	3	0	2
C	0	6	674	1	3
D	0	5	5	633	0

```
#graphics.off()
```

Out of Sample Error Comparisons for the 3 models Out of sample error (OOS) tests model error using an independent dataset from the training dataset upon which the model was built. To compute OOS error, predictions are calculated using the crossv dataset, which was set aside and not touched during the training exercises. Although it is a completely separate dataset from the training, the crossv data comes from the same original raw source and thus, is not strictly independent in terms of how another new set of subjects would perform the dumbbell exercise, how the sensors would detect motion etc.

Below are the accuracy measures for each of the three models using the confusion matrix prediction on the cross validation dataset and matrix algebra. Because the outcome variable is discrete, not continuous, simple measures derived from the confusion matrix may be sufficient (classification variable where classe=A,B,C,D,E).

Accuracy = Sum of [i,i] diagonal / Sum of entire matrix where i=1,2,...5

```
accuracy.54<-sum(diag(as.matrix(plot54)))/sum(plot54)
accuracy.30<-sum(diag(as.matrix(plot30)))/sum(plot30)
accuracy.2G<-sum(diag(as.matrix(plot2G)))/sum(plot2G)

print("The accuracy statistic for the 54 var model is:")

## [1] "The accuracy statistic for the 54 var model is:"

accuracy.54
```

```
## [1] 0.9954117
```

```
print("The accuracy statistic for the 30 var model is:")
```

```
## [1] "The accuracy statistic for the 30 var model is:"
```

```
accuracy.30
```

```
## [1] 0.9819016
```

```
print("The accuracy statistic for the 2 (GINI derived) var model is:")
```

```
## [1] "The accuracy statistic for the 2 (GINI derived) var model is:"
```

```
accuracy.2G
```

```
## [1] 0.9885292
```

Confusion Matrix and error statistics for the 2 variable GINI model Here are the out of sample error statistics, sensitivities, specificities etc for the final, 2 variable model.

```
confusionMatrix(plot2G)
```

```
## Confusion Matrix and Statistics
##
##    predict.2G
##        A    B    C    D    E
##   A 1110    1    4    1    0
##   B    8  746    3    0    2
##   C    0    6  674    1    3
##   D    0    5    5  633    0
##   E    2    3    1    0  715
##
## Overall Statistics
##
##                Accuracy : 0.9885
##                  95% CI : (0.9847, 0.9916)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9855
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9911   0.9803   0.9811   0.9969   0.9931
## Specificity            0.9979   0.9959   0.9969   0.9970   0.9981
## Pos Pred Value         0.9946   0.9829   0.9854   0.9844   0.9917
## Neg Pred Value         0.9964   0.9953   0.9960   0.9994   0.9984
## Prevalence             0.2855   0.1940   0.1751   0.1619   0.1835
## Detection Rate         0.2829   0.1902   0.1718   0.1614   0.1823
```

```
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9945   0.9881   0.9890   0.9969   0.9956
```

V. Predict Classe for 20 Test Cases

The following code reads the pml-testing.csv dataset, containing 20 test cases, and predicts the classes using 54,30,2 variable models. The predictions for each of the three models are identical. This is another way of demonstrating that the 2 factor model is sufficiently accurate for this particular dataset.

```
setwd("~/Actuarial/Coursera/Data Science/Machine Learning/Assignment")
subm.test<-read.csv(file="pml-testing.csv")

# Count number of NA values for each column, keep only the columns with zero
NA values
dfNA<-data.frame(x=1:ncol(subm.test),y=1)
for(i in 1:ncol(subm.test)){
  dfNA[i,2]<-sum(is.na(subm.test[,i]))
}

subm.mydf2<-data.frame(subm.test[,which(dfNA[,2]==0)])
dfMISS<-data.frame(x=1:ncol(subm.mydf2),y=1)
for(i in 1:ncol(subm.mydf2)){
  dfMISS[i,2]<-sum(ifelse(subm.mydf2[,i]=="",1,0))
}
subm.mydf3<-data.frame(subm.mydf2[,which(dfMISS[,2]==0)])
row.names(subm.mydf3)<-NULL

# Remove time stamp columns, keep num_window which captures the time stamp in
buckets
# This cleaning step brings mydf3 down to only 54 predictors and the classe
column
subm.mydf3<-subm.mydf3[,-1]
subm.mydf3<-subm.mydf3[,-2:-5]

#setwd("~/Actuarial/Coursera/Data Science/Machine
Learning/Assignment/submission files")
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

set.seed(42)
predict.54.sub<-
predict(modFit.54,newdata=subm.mydf3,type="response",norm.votes=TRUE,predict.
all=FALSE,proximity=FALSE,nodes=FALSE)
```

```
set.seed(42)
predict.30.sub<-
predict(modFit.30,newdata=subm.mydf3,type="response",norm.votes=TRUE,predict.
all=FALSE,proximity=FALSE,nodes=FALSE)

set.seed(42)
predict.2.sub<-
predict(modFit.2G,newdata=subm.mydf3,type="response",norm.votes=TRUE,predict.
all=FALSE,proximity=FALSE,nodes=FALSE)

answers54<-as.character(predict.54.sub)
answers30<-as.character(predict.30.sub)
answers2<-as.character(predict.2.sub)
print("The 20 predicted classes for the 54 var model are:")
```

```
## [1] "The 20 predicted classes for the 54 var model are:"
```

```
answers54
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

```
print("The 20 predicted classes for the 30 var model are:")
```

```
## [1] "The 20 predicted classes for the 30 var model are:"
```

```
answers30
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

```
print("The 20 predicted classes for the 2 var model are:")
```

```
## [1] "The 20 predicted classes for the 2 var model are:"
```

```
answers2
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

```
#pml_write_files(answers11)
```

VI. Conclusion

In summary, a 2-variable random forest classification model produced accuracy and out of sample error measures about as good as for the full 54 variable model while reducing overfitting issues apparent with the latter, and the 2 variable model predicted the exercise class in the final 20 test case set 100% accurately. As wearable device derived sensor data become more available, it would be interesting to verify and validate the model created in this assignment using independent data on new/different subjects performing the same 5 exercises. It would appear that the models chosen by the authors of the original paper may

suffer from overfitting and/or high data dependency - and, such a validation exercise may help to answer similar concerns.