Weight Lifting Incorrect Method Classification

A. Trivelli

May 5, 2016

Executive Summary

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. Through leverging these devices, test subjects were setup with several of these devices and then instructed to perform dumbell weight exercises both correctly and incorrectly in 5 different manners (A through E).

The purpose of this submission is to explain how machine learning algorithms were leveraged to train a predictive model and then leverage the model to predict 20 test cases further on.

Thanks to http://groupware.les.inf.puc-rio.br/har group for letting us utilize their underlying Weight Lifting data for this analysis.

Outcome

After pre-processing the data to handle large numbers of blanks and NAs in the raw data, several machine learning models were explored including Rpart and Random Forest. It was the original intent to attempt to stack several models for good predictive results but this gave way to extremely good results from leveraging the Random Forest model, albeit slow.

The Rpart model only obtained an accurracy of approximately .49, as per the confusion matrix, but it did show some of the key variables that were leveraged but also suffered from not being able to predict the 'D' Classe case.

Based upon the rpart model exploration, an initial Random Forest was created and then was leveraged to determine the top 20 variables of importance. These variables were then leverged for subsequent random forest models, one leveraging K fold Cross Validation of 3 and another leveraging K fold cross validation with K set to 10. With each of these, the Cross validation was performed within the Random Forest training model.

The RF model with a K-fold of 3 had an accuracy of .9983 against a validation data set that was set aside and the RF model with a K-fold of 10 also had an accuracy of .9975. With each I am concerned with overfitting but given that a Cross validation leveraging a K fold of 10 (my final model).

To address the overfitting concern, a Pruned RF (20 variables) with a Cross Validation of K-Fold= 10 was built that produced Accuracy of 1.0 against the validation data set. This results in my final model.

Overall, I am expecting an out-of-sample error rate of less than 5%.

Methods

Pre-Processing:

- Initially loaded both the training and test datasets and noticed a large number of NAs and blanks.
- Given the above, reloaded both training and test to have NAs inserted instead of blanks for subsequent pre-processing.
- I noticed that many of the summary variables were afflicted with the NAs/blanks and made the determination that any column with NAs will be removed from the training and test data sets, which reduced the activity related variables down to 60.
- In addition, during an intial run in the rpart model, the model was leveraging the X field exclusively. Given this and some thought, I removed the X, ID, person, timestamp fields up to the first 7 fields leaving the activity measurement related fields.

Model Selection, Parameter Selection and Cross Validation:

The initial intent was to create a stacked model leveraging several models based upon their overall results. This was facilidated by the creation of a validation data set from the training set. Given this model selection process was as follows, given that this is a classification problem with linear variables:

- Run a basic Rpart tree with all 60 available variables for two puposes consisting of:
    - Get an overall feel how a model will react to what I think is an embedded timeseries with the data as denoted by the datas original authors (e.g. sliding sampling window).
    - See the accuracy of rpart model in this application
- Basic Random Forest K-3 CV- run a basic Random Forest with all the available variable. From this run, investigate the accuarcy, see how this model performs against the validation test set and lastly leverage this model run to assist in subsequent paramater/variable selection by investigating the Variables of Importance.

- Basic Random Forest K-10 CV- same as the basic Random Forest with an increased Kfold Cross validation from 3 to 10.

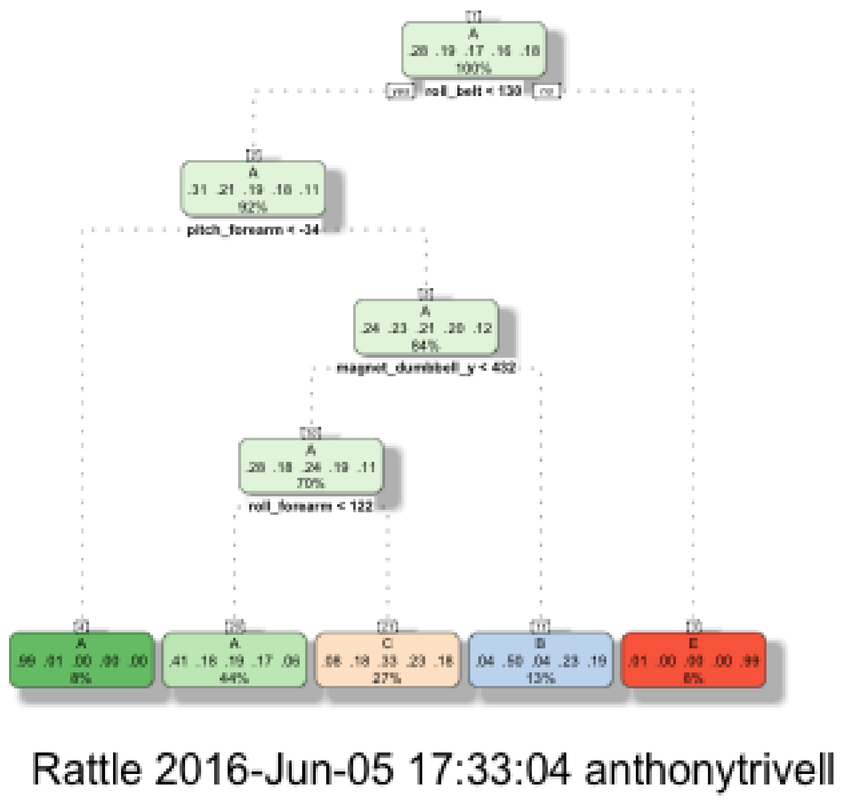
- Pruned Random Forest- Assuming that the basic Random Forest was farily reasonable, leverage the preliminary information from the previous RF model and limit the Pruned Random Forest to a limited number of parameters/variables along with K-fold(3)/Cross validation and retest against the validation data set.

The variables utilized for the Pruned Random Forest are as follows - roll_belt + magnet_dumbbell_y+ pitch_belt + pitch_forearm + pitch_belt + accel_forearm_x + magnet_dumbbell_z + roll_forearm + yaw_arm + gyros_belt_z + accel_dumbbell_y + gyros_dumbbell_y + magnet_belt_z + accel_dumbbell_z + gyros_forearm_y +magnet_belt_x + magnet_forearm_z + gyros_arm_y + total_accel_dumbbell + roll_dumbbell

- Pruned Random Forest K-10 CV- This model is identical to the Pruned Random Forest with the exception of changing the Cross Validation K-Fold to 10. This is my final model that is utilzed to predict against the test data set because, even though it leverages only 20 variables, in a Cross Validation with K-Fold = 10, it had an Accuracy of 1.0 against the validation data set. I am assuming that the higher accuracy rate is due to a combination of the limited variables (20), a K-Fold Cross validation of 10 and therefore less over-fitting.

Rpart Model

The initial model/technique to be explored was the Rpart algorithm. THis was utilized as more as an exploratory on how the a tree model would react to the training data.



The Rpart model's confusion matrix, as depicted below, reports an Accuracy of .4968 which is obviously inferior. Given this, the rpart model will not be considered as as part of a stacking approach.

Basic Random Forest- Cross Validation K-Fold= 3

A basic random forest is executed, including all 60 variables, along with its Confusion Matrix. The random forest is run with a K Fold Cross Validation with K set to 3 and number of trees set to 50. This decision was made due to the long running training period with the default ntree setting.

The resulting Confusion Matrix against the held back validation set is as follows:

```
## build RF model
##rfFit <- train( classe~ . , data= training, method= "rf", prox=TRUE,do.trac
e= TRUE, ntree= 50, trcontrol= trainControl(method="cv"), number=3)

##save(rfFit,file="rfFit.RData")

load("./rfFit.Rdata")

## Run confusion matrix back against the training data
confusionMatrix( predict( rfFit, val), val$classe )
```

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 1143 0 0 0 0 B 0 811 1 0 1 C 0 0 722 2 0 D 1 0 1 687 1 E 0 0 0 0 751

Overall Statistics

```
           Accuracy : 0.9983
             95% CI : (0.9965, 0.9993)
No Information Rate : 0.2776
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.9979
```

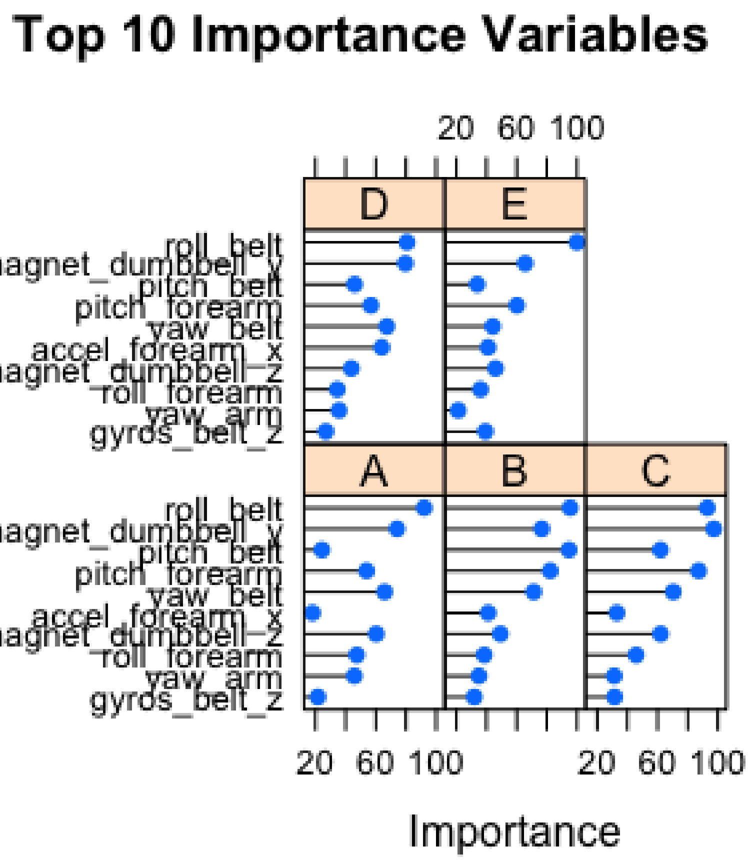
Mcnemar's Test P-Value : NA

Statistics by Class:

```
                 Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.9991 1.0000 0.9972 0.9971 0.9973 Specificity 1.0000 0.9994 0.9994 0.9991 1.0000 Pos Pred Value 1.0000 0.9975 0.9972 0.9957 1.0000 Neg Pred Value 0.9997 1.0000 0.9994 0.9994 0.9994 Prevalence 0.2776 0.1968 0.1757 0.1672 0.1827 Detection Rate 0.2774 0.1968 0.1752 0.1667 0.1822 Detection Prevalence 0.2774 0.1973 0.1757 0.1674 0.1822 Balanced Accuracy 0.9996 0.9997 0.9983 0.9981 0.9987

```
## Top 20 Importance Variables
variableImp <- varImp(rfFit)
## plot( rfFit , main= "Accuracy Analysis vs. Number of Variables")
plot( varImp(rfFit), top=10, main= "Top 10 Importance Variables")
```



```
#variableImp
#top20Factors <- row.names(variableImp$importance)[1:20]
```

Random Forest - K-Fold(3) with Variables limited to Top 20 Important Variable

The next model that was explored was a similiar Random Forest as previously done, but with the training input variables limited to the Top 20 Important Variables as determined. This model also had excellent predictive result and benefitted from a quicker training time due to the limited variables.

The confusion matrix, when the trained model was run against the held back validation data with an accuracy of 0.9971.

Random Forest -All Variables with Cross Validation K-Fold 10 and ntree = 50

The next model created was a Random Forest, with the full complement of 60 input variables but with Cross Validation K-Fold set at 10. This resulted in a Confusion Matrix against the validation data set of 0.9975.

Pruned Random Forest (Top 20 Importance Variables) with Cross Validation K-Fold 10 and ntree = 50

The last and Final model created was a Random Forest, with the Top 20 Variables of Importnace but with Cross Validation K-Fold set at 10. As depicted below, this model has an accuracy of 1.0.

```
#prunedrfFi10t <- train( classe ~ roll_belt + magnet_dumbbell_y+ pitch_belt +
pitch_forearm + pitch_belt + accel_forearm_x + magnet_dumbbell_z + roll_forea
rm + yaw_arm + gyros_belt_z  + accel_dumbbell_y + gyros_dumbbell_y + magnet_b
elt_z + accel_dumbbell_z  + gyros_forearm_y +magnet_belt_x + magnet_forearm_z
+ gyros_arm_y + total_accel_dumbbell + roll_dumbbell ,data= training, method=
"rf", prox=TRUE,do.trace= TRUE, ntree= 50, trcontrol= trainControl(method="cv
"), number=10)

#save(prunedrfFi10t, file="prunedrfFi10t.RData")

load( "./prunedrfFi10t.RData" )
```

The Confusion Matrix for the Pruned RF with Cross Validation K-Fold set to 10 is as follows along with the error curves, which with their flattening between 30 to 40 trees confirm the lowering of the ntree to 50 was a reasonable choice. Based upon this results (Accuracy= 1.0) with a limited # of parameters and Cross Validation with K-Fold= 10, this is the final model to used.

Confusion Matrix and Statistics

```
      Reference
```

Prediction A B C D E A 1144 0 0 0 0 B 0 810 0 0 0 C 0 1 724 4 1 D 0 0 0 685 2 E 0 0 0 0 750

Overall Statistics

```
           Accuracy : 0.9981
             95% CI : (0.9962, 0.9992)
No Information Rate : 0.2776
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.9975
```

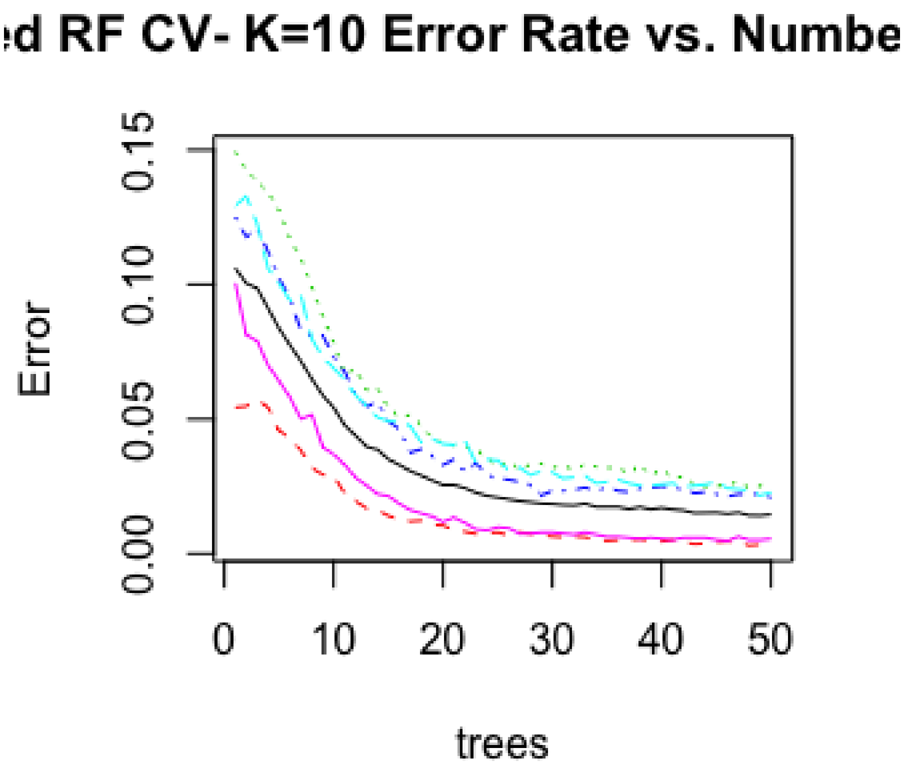
Mcnemar's Test P-Value : NA

Statistics by Class:

```
                 Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 1.0000 0.9988 1.0000 0.9942 0.9960 Specificity 1.0000 1.0000 0.9982 0.9994 1.0000 Pos Pred Value 1.0000 1.0000 0.9918 0.9971 1.0000 Neg Pred Value 1.0000 0.9997 1.0000 0.9988 0.9991 Prevalence 0.2776 0.1968 0.1757 0.1672 0.1827 Detection Rate 0.2776 0.1966 0.1757 0.1662 0.1820 Detection Prevalence 0.2776 0.1966 0.1771 0.1667

0.1820 Balanced Accuracy 1.0000 0.9994 0.9991 0.9968 0.9980



Predictions of the 20 Test Cases:

```
predict( prunedrfFi10t,testing[testing$problem_id== 1,])
```

[1] B Levels: A B C D E

```
predict( prunedrfFi10t,testing[testing$problem_id== 2,])
```

[1] A Levels: A B C D E

```
predict( prunedrfFi10t,testing[testing$problem_id== 3,])
```

[1] B Levels: A B C D E

```
predict( prunedrfFi10t,testing[testing$problem_id== 4,])
```

[1] A Levels: A B C D E

```
predict( prunedrfFi10t,testing[testing$problem_id== 5,])
```

[1] A Levels: A B C D E

```
predict( prunedrfFi10t,testing[testing$problem_id== 6,])
```

[1] E Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 6,])
```

[1] E Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 8,])
```

[1] B Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 9,])
```

[1] A Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 10,])
```

[1] A Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 11,])
```

[1] B Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 12,])
```

[1] C Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 13,])
```

[1] B Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 14,])
```

[1] A Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 15,])
```

[1] E Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 16,])
```

[1] E Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 17,])
```

[1] A Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 18,])
```

[1] B Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 19,])
```

[1] B Levels: A B C D E

```r
predict( prunedrfFi10t,testing[testing$problem_id== 20,])
```

[1] B Levels: A B C D E

Appendix/Code

```
##load in libraries needed

library(caret)
library(randomForest)
library(rattle)
library(e1071)
library(rpart.plot)
## download and load training and testing data


download.file( "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv", "pml-training.csv.webarchive")

download.file( "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testi
ng.csv", "pml-testing.csv.webarchive")

## Load in the training and test, but due to large number of blanks in initia
l load reload converting ## blanks to NAs for cleanup


training <- read.csv("./pml-training.csv.webarchive", sep= ",", na.strings= c
("NA",""))
testing <- read.csv("./pml-testing.csv.webarchive", sep= ",", na.strings= c("
NA",""))

dim(training)
dim(testing)
str(training)
summary(training)
colList <- names(training)
noCols <- length(colList)
noCols
## convert all the factors to numerics
training[,7:(noCols-1)] <- sapply(training[,7:(noCols-1)],as.numeric)
testing[,7:(noCols-1)] <- sapply(testing[,7:(noCols-1)], as.numeric)

## there are a lot of columns that have blanks/NA- went back an reloaded blan
ks as NA in order cleanup
naMap <- sapply(training, anyNA )

training <- training[,!naMap]
dim(training)
testing <- testing[,!naMap]
dim(testing)

## remove the first 7 columns, especially the X as the rpart is using it as t
he principle factor
training <- training[ ,8:60]
```

```r
testing <- testing[ ,8:60]

# create validation data set using Train
set.seed(2345)
inTrain <- createDataPartition(y=training$classe, p=0.7, list=FALSE)
training <- training[inTrain,]
val <- training[-inTrain,]

trainRpartFit <- train( classe ~ . , method= "rpart", data= training)

fancyRpartPlot(trainRpartFit$finalModel)

confusionMatrix( predict( trainRpartFit, training), training$classe )

## build RF model
##rfFit <- train( classe~ . , data= training, method= "rf", prox=TRUE,do.trac
e= TRUE, ntree= 50, trcontrol= trainControl(method="cv"), number=3)

##save(rfFit,file="rfFit.RData")

load("./rfFit.Rdata")

## Run confusion matrix back against the training data
confusionMatrix( predict( rfFit, val), val$classe )

## Top 20 Importance Variables
variableImp <- varImp(rfFit)
## plot( rfFit , main= "Accuracy Analysis vs. Number of Variables")
plot( varImp(rfFit), top=10, main= "Top 10 Importance Variables")
#variableImp
#top20Factors <- row.names(variableImp$importance)[1:20]

## prunedrfFit <- train( classe ~ roll_belt + magnet_dumbbell_y+ pitch_belt +
pitch_forearm + pitch_belt + accel_forearm_x + magnet_dumbbell_z + roll_forea
rm + yaw_arm + gyros_belt_z  + accel_dumbbell_y + gyros_dumbbell_y + magnet_b
elt_z + accel_dumbbell_z  + gyros_forearm_y +magnet_belt_x + magnet_forearm_z
+ gyros_arm_y + total_accel_dumbbell + roll_dumbbell ,data= training, method=
"rf", prox=TRUE,do.trace= TRUE, ntree= 50, trcontrol= trainControl(method="cv
"), number=3)

# save(prunedrfFit, file="prunedrfFit.RData")

load("./prunedrfFit.Rdata" )

## Run Confusion Matrix of Pruned RF model against Validation data
confusionMatrix( predict( prunedrfFit, val), val$classe )

#rfFit10 <- train( classe~ . , data= training, method= "rf", prox=TRUE,do.tra
ce= TRUE, ntree= 50, trcontrol= trainControl(method="cv"), number=10, allowPa
rallel= TRUE)
```

```r
# save(rfFit10, file= "rfFit10.Rdata")

load("./rfFit10.Rdata")

## Random Forest with Cross Validation K-Fold= 10 against Validation data
confusionMatrix( predict( rfFit10, val), val$classe )
plot(prunedrfFit$finalModel, main= "Full Random Forest with CV K-fold= 10")

# save(prunedrfFit, file="prunedrfFit.RData")

load("./prunedrfFit.Rdata" )

# rfFit10 <- train( classe~ . , data= training, method= "rf", prox=TRUE,do.tr
ace= TRUE, ntree= 50, trcontrol= trainControl(method="cv"), number=10, allowP
arallel= TRUE)

#save(rfFit10, file= "rfFit10.Rdata")
 load("./rfFit10.Rdata")

confusionMatrix( predict( rfFit10, training), training$classe )

#prunedrfFi10t <- train( classe ~ roll_belt + magnet_dumbbell_y+ pitch_belt +
pitch_forearm + pitch_belt + accel_forearm_x + magnet_dumbbell_z + roll_forea
rm + yaw_arm + gyros_belt_z  + accel_dumbbell_y + gyros_dumbbell_y + magnet_b
elt_z + accel_dumbbell_z  + gyros_forearm_y +magnet_belt_x + magnet_forearm_z
+ gyros_arm_y + total_accel_dumbbell + roll_dumbbell ,data= training, method=
"rf", prox=TRUE,do.trace= TRUE, ntree= 50, trcontrol= trainControl(method="cv
"), number=10)

#save(prunedrfFi10t, file="prunedrfFi10t.RData")

load( "./prunedrfFi10t.RData" )

confMatrix <- confusionMatrix( predict( prunedrfFi10t, val), val$classe )

confMatrix

plot(prunedrfFi10t$finalModel, main= "Pruned RF CV- K=10 Error Rate vs. Numbe
r of Trees")
```