

Practical Machine Learning Project: Weight Lifting Exercise Classification

John Slough II

12 July 2015

Introduction

This project is concerned with identifying the execution type of an exercise, the Unilateral Dumbbell Biceps Curl. The dataset includes readings from motion sensors on participants bodies'. These readings will be used to classify the performed exercise into five categories: 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). Please see the website <http://groupware.les.inf.puc-rio.br/har> for more information.

The Data

Processing:

```
#setwd("~/Desktop/Courses/Coursera/Machine Learning/Project")
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-10
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ggthemes)
```

```
library(gridExtra)
```

```
library(ggplot2)
```

```
library(grid)
```

```
train = read.csv("pml-training.csv",header=TRUE)
```

```
train_used = train[,c(8:11,37:49,60:68,84:86,102,113:124,140,151:160)]
```

The raw dataset contained 19622 rows of data, with 160 variables. Many variables contained largely missing data (usually with only one row of data), so these were removed from the dataset (these variables were also not used in the final 20 testing dataset). In addition, variables not concerning the movement sensors were also removed. This resulted in a dataset of 53 variables.

To understand the structure of the data a bit better, density plots were made of a selection of the data. These are displayed below.

```
A=ggplot() + geom_density(aes(x=gyros_belt_x, colour="red", data=train_used) +  
  geom_density(aes(x=gyros_belt_y, colour="green", data=train_used)+  
  geom_density(aes(x=gyros_belt_z, colour="blue", data=train_used)+  
  theme_few()+xlab("Gyro Belt (xyz)"))
```

```

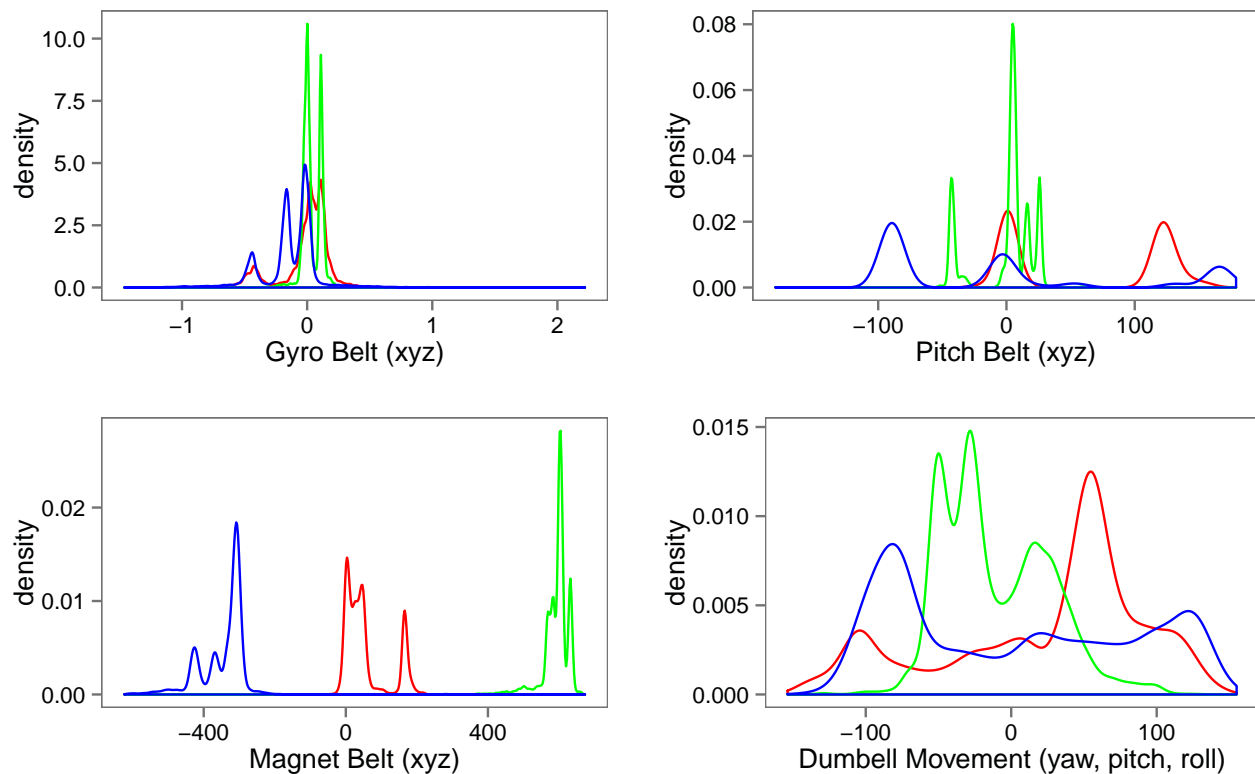
B=ggplot() +geom_density(aes(x=roll_belt, colour="red", data=train_used) +
  geom_density(aes(x=pitch_belt, colour="green", data=train_used)+
  geom_density(aes(x=yaw_belt, colour="blue", data=train_used))+theme_few()+
  xlab("Pitch Belt (xyz)")

C=ggplot() +geom_density(aes(x=magnet_belt_x, colour="red", data=train_used) +
  geom_density(aes(x=magnet_belt_y, colour="green", data=train_used)+
  geom_density(aes(x=magnet_belt_z, colour="blue", data=train_used))+theme_few()+
  xlab("Magnet Belt (xyz)")

D=ggplot() +geom_density(aes(x=roll_dumbbell, colour="red", data=train_used) +
  geom_density(aes(x=pitch_dumbbell, colour="green", data=train_used)+
  geom_density(aes(x=yaw_dumbbell, colour="blue", data=train_used))+theme_few()+
  xlab("Dumbbell Movement (yaw, pitch, roll)")

Dplots=arrangeGrob(A, B, C, D ,nrow = 2, ncol = 2)
grid.draw(Dplots)

```



Partitioning the Data

The dataset was partitioned into training and testing datasets, with 60% of the original data going to the training set and 40% to the testing set. The model was built with the training dataset, then tested on the testing dataset. The following code performs this procedure:

```

# partition training dataset into 60/40 train/test
train_part = createDataPartition(train_used$classe, p = 0.6, list = FALSE)
training = train_used[train_part, ]
testing = train_used[-train_part, ]
##

```

The Model

Many methods of classification were attempted, including naive Bayes, multinomial logistic regression, and Support Vector Machines. It was determined that the Random Forest method produced the best results. In addition, pre-processing using principal component analysis was attempted however this greatly reduced the prediction accuracy.

Cross validation was not used, as, according to the creators of the Random Forest algorithm Leo Breiman and Adele Cutler, there is no need for cross-validation. See http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr for more information.

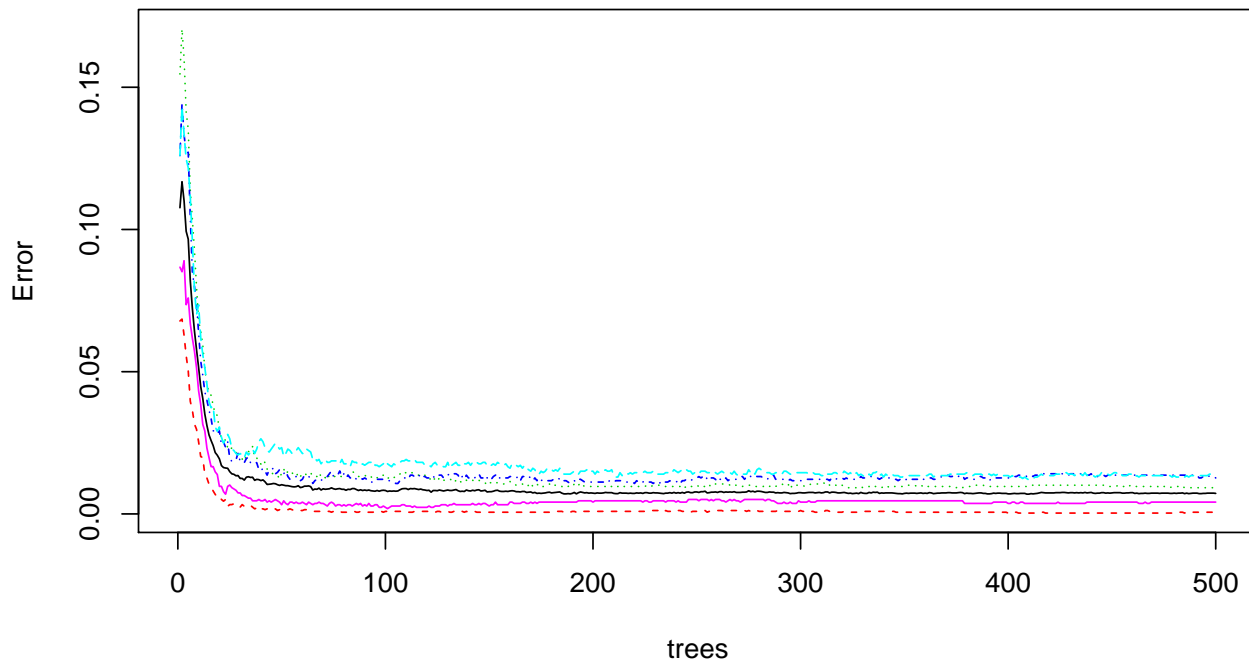
The R code is shown below, as is the confusion matrix. The out of the bag error rate in the training and the confusion matrix is shown below. For informational purposes a plot of the error rate versus number of trees is also shown.

```
set.seed(1777)
random_forest=randomForest(classe~.,data=training,ntree=500,importance=TRUE)
random_forest

##
## Call:
## randomForest(formula = classe ~ ., data = training, ntree = 500,      importance = TRUE)
##              Type of random forest: classification
##              Number of trees: 500
## No. of variables tried at each split: 7
##
##              OOB estimate of  error rate: 0.72%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346     1     0     1     0 0.0005973716
## B   12 2258     9     0     0 0.0092145678
## C    0   23 2028     3     0 0.0126582278
## D    0    0   25 1903     2 0.0139896373
## E    0    0    3    6 2156 0.0041570439

plot(random_forest,main="Random Forest: Error Rate vs Number of Trees")
```

Random Forest: Error Rate vs Number of Trees



Variable Importance

It may be of interest to know which variables were most ‘important’ in the building of the model. This can be seen by plotting the mean decrease in accuracy and the mean decrease in the Gini coefficient per variable. In short, the more the accuracy of the random forest decreases due to the exclusion (or permutation) of a single variable, the more important that variable is deemed to be. The mean decrease in the Gini coefficient is a measure of how each variable contributes to the homogeneity of the nodes and leaves in the resulting random forest. (from <https://dinsdalelab.sdsu.edu/metag.stats/code/randomforest.html>)

```
imp=importance(random_forest)
impL=imp[,c(6,7)]
imp.ma=as.matrix(impL)
imp.df=data.frame(imp.ma)

write.csv(imp.df, "imp.df.csv", row.names=TRUE)
imp.df.csv=read.csv("imp.df.csv",header=TRUE)

colnames(imp.df.csv)=c("Variable", "MeanDecreaseAccuracy", "MeanDecreaseGini")
imp.sort = imp.df.csv[order(-imp.df.csv$MeanDecreaseAccuracy),]

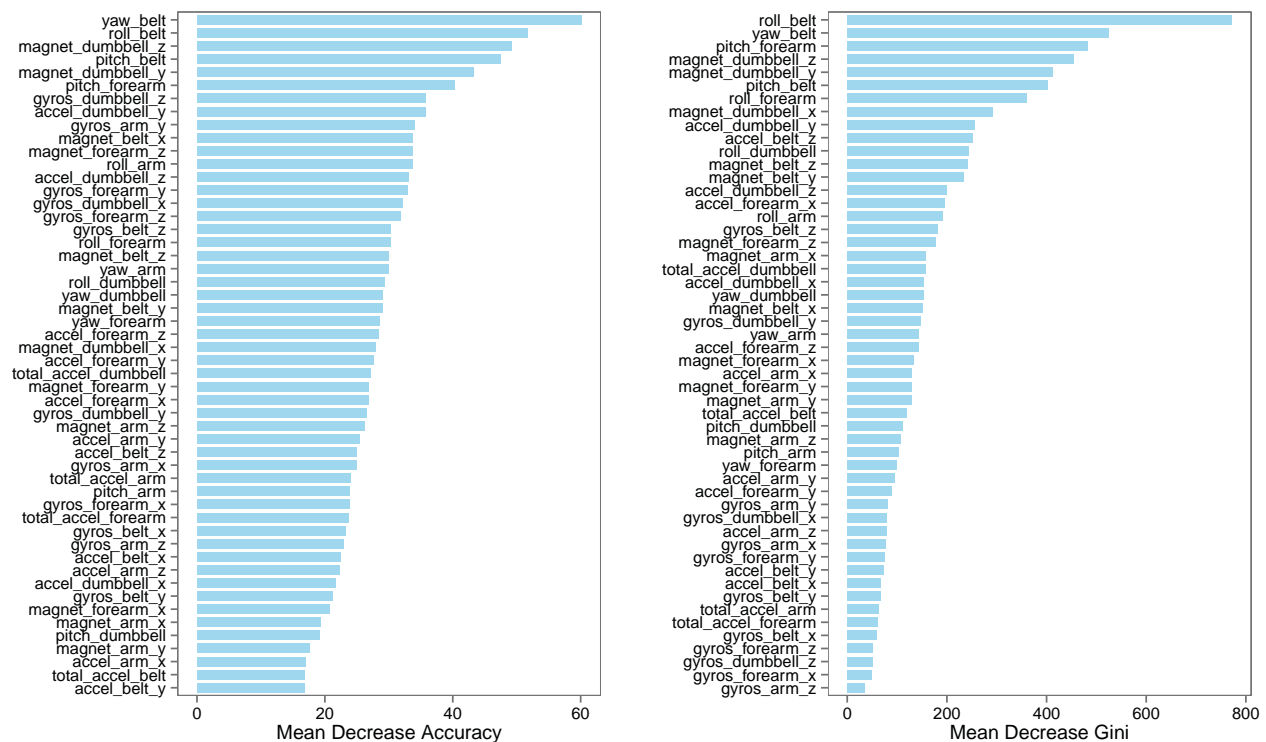
imp.sort = transform(imp.df.csv,
  Variable = reorder(Variable, MeanDecreaseAccuracy))

VIP=ggplot(data=imp.sort, aes(x=Variable, y=MeanDecreaseAccuracy)) +
  ylab("Mean Decrease Accuracy")+xlab("")+
  geom_bar(stat="identity", fill="skyblue", alpha=.8, width=.75)+
  coord_flip()+theme_few()

imp.sort.Gini <- transform(imp.df.csv,
  Variable = reorder(Variable, MeanDecreaseGini))
```

```
VIP.Gini=ggplot(data=imp.sort.Gini, aes(x=Variable, y=MeanDecreaseGini)) +
  ylab("Mean Decrease Gini")+xlab("")+
  geom_bar(stat="identity",fill="skyblue",alpha=.8,width=.75)+
  coord_flip()+theme_few()
```

```
VarImpPlot=arrangeGrob(VIP, VIP.Gini,ncol=2)
grid.draw(VarImpPlot)
```



Model Applied to Testing Dataset

```
test_predictions = predict(random_forest, newdata=testing)
CM = confusionMatrix(test_predictions,testing$classe)
CM
```

Confusion Matrix and Statistics

```
##
##           Reference
## Prediction   A    B    C    D    E
##           A 2231   15    0    0    0
##           B    1 1495   16    0    0
##           C    0    8 1352   16    0
##           D    0    0    0 1270    4
##           E    0    0    0    0 1438
```

Overall Statistics

```
##
##           Accuracy : 0.9924
##           95% CI : (0.9902, 0.9942)
```

```
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9903
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.9996   0.9848   0.9883   0.9876   0.9972
## Specificity          0.9973   0.9973   0.9963   0.9994   1.0000
## Pos Pred Value       0.9933   0.9888   0.9826   0.9969   1.0000
## Neg Pred Value       0.9998   0.9964   0.9975   0.9976   0.9994
## Prevalence           0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate       0.2843   0.1905   0.1723   0.1619   0.1833
## Detection Prevalence 0.2863   0.1927   0.1754   0.1624   0.1833
## Balanced Accuracy    0.9984   0.9911   0.9923   0.9935   0.9986
```

The model was applied to the testing dataset and generated predictions for the class of weightlifting type. Above is the code that was used and the confusion matrix for the testing dataset. The accuracy is very high, 0.9924. The model accurately predicted all of the 20 test subjects.

Cross Validation

Just in case the grader feels it is necessary to do cross validation, I have added the code and error rates from the CV from the caret package. The cross-validation error is shown below. The model with CV leads to slightly poorer performance than the previous one.

```
CV = trainControl(method = "cv", number = 5, allowParallel = T, verboseIter = F)
CVmodel = train(classe ~ ., data = training, method = "rf", prox = F, trControl = CV)
CVmodel
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9421, 9421, 9421, 9420, 9421
## Resampling results across tuning parameters:
##
##  mtry  Accuracy  Kappa      Accuracy SD  Kappa SD
##    2    0.9875171 0.9842075  0.002374385  0.003003771
##   27    0.9877720 0.9845300  0.002499365  0.003162878
##   52    0.9800448 0.9747543  0.005153059  0.006523229
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
predsCVmodel = predict(CVmodel, newdata = testing)

confMatrix = confusionMatrix(predsCVmodel, testing$classe)
confMatrix
```

```

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 2227   17    0    0    0
##           B    3 1484   22    1    0
##           C    1   16 1337   20    2
##           D    0    1    9 1265    7
##           E    1    0    0    0 1433
##
## Overall Statistics
##
##           Accuracy : 0.9873
##           95% CI : (0.9845, 0.9896)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9839
##           McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9978  0.9776  0.9773  0.9837  0.9938
## Specificity      0.9970  0.9959  0.9940  0.9974  0.9998
## Pos Pred Value   0.9924  0.9828  0.9717  0.9867  0.9993
## Neg Pred Value   0.9991  0.9946  0.9952  0.9968  0.9986
## Prevalence       0.2845  0.1935  0.1744  0.1639  0.1838
## Detection Rate   0.2838  0.1891  0.1704  0.1612  0.1826
## Detection Prevalence 0.2860  0.1925  0.1754  0.1634  0.1828
## Balanced Accuracy 0.9974  0.9867  0.9857  0.9905  0.9968

```