Practical Machine Learning Project: Weight Lifting Exercise Classification

Ozgur Erciyes
21.November.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.

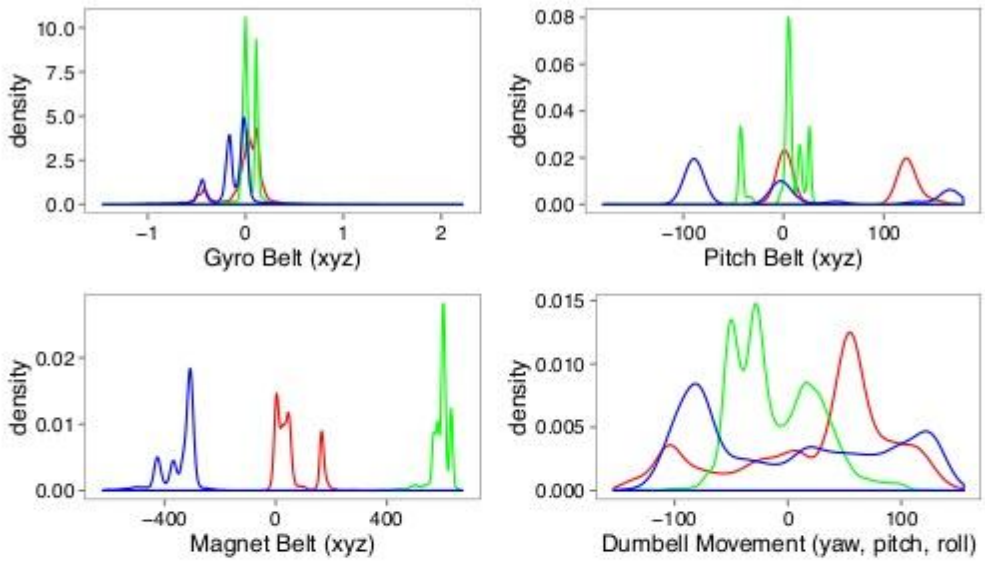
The Data

Processing:

```
setwd("~/Desktop/Courses/Coursera/Data Science Specialization/Machine Learning/Project")

library(caret)
library(randomForest)
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. 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.



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 niave Bayes, multinomial logistic regression, and decision trees. It was determined that the Random Forest method produced the best results. In addition, 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: "In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error." - Leo Breiman and Adele Cutler

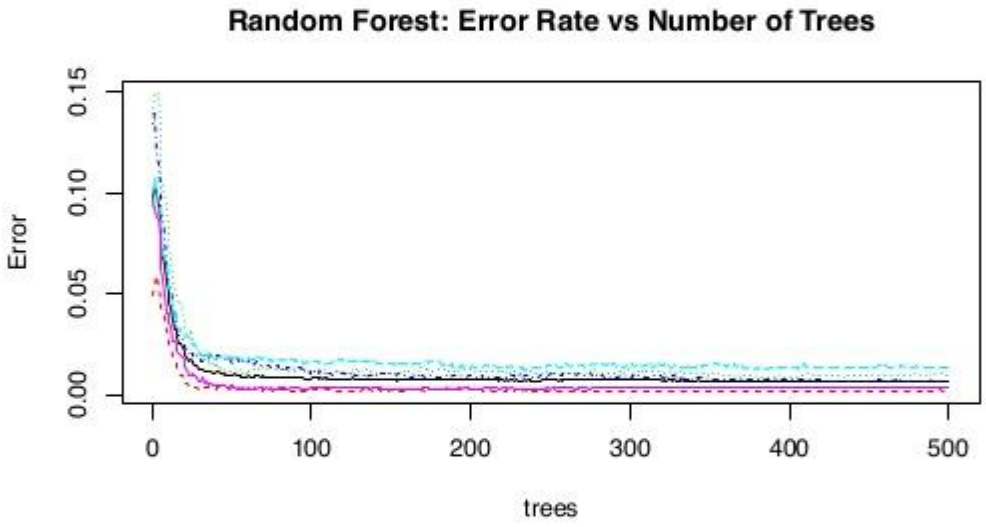
The R code is shown below, as is the confusion matrix. The OOB error rate in the training and the confusion matrix is shown below. For informational purposes a plot of the error rate versus number of trees is also shown.

```
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: 0.65%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3341    6    0    1    0 0.002090800
## B   12 2257   10    0    0 0.009653357
## C    0    9 2041    4    0 0.006329114
## D    0    0   23 1903    4 0.013989637
## E    0    0    2    6 2157 0.003695150
```

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

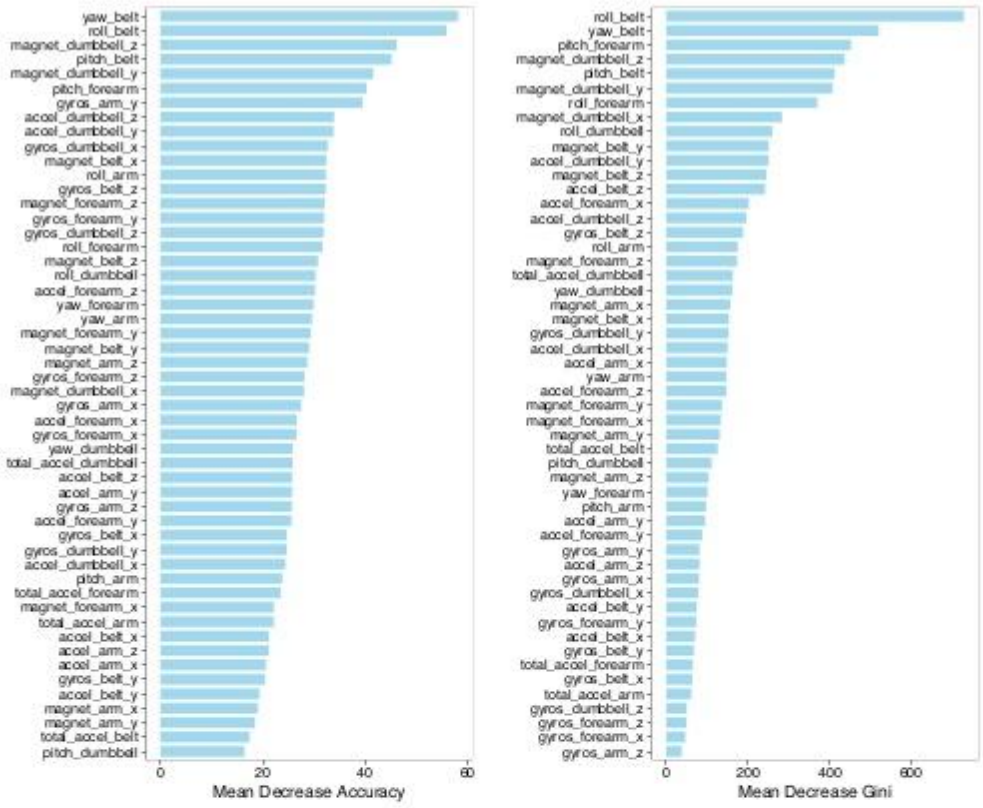


Variable Importance

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 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)
```

```r
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)
confusionMatrix(test_predictions,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    9    0    0    0
##          B    0 1506   12    0    0
##          C    0    3 1353   16    0
##          D    0    0    3 1269    2
##          E    1    0    0    1 1440
##
## Overall Statistics
##
##                Accuracy : 0.994
##                  95% CI : (0.992, 0.9956)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9924
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9921   0.9890   0.9868   0.9986
## Specificity            0.9984   0.9981   0.9971   0.9992   0.9997
## Pos Pred Value         0.9960   0.9921   0.9862   0.9961   0.9986
## Neg Pred Value         0.9998   0.9981   0.9977   0.9974   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1919   0.1724   0.1617   0.1835
## Detection Prevalence   0.2855   0.1935   0.1749   0.1624   0.1838
## Balanced Accuracy      0.9990   0.9951   0.9931   0.9930   0.9992
```

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, at over 99%. 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.

```
## 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, 9420, 9422, 9420
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9881112  0.9849589  0.002973074  0.003762158
##   27    0.9881962  0.9850681  0.003468308  0.004386263
##   52    0.9804688  0.9752894  0.004522250  0.005720106
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229   11    0    0    0
##          B    1 1501   15    0    6
##          C    1    4 1348   23    3
##          D    0    2    5 1262    2
##          E    1    0    0    1 1431
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9904
##                  95% CI : (0.988, 0.9925)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9879
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9987   0.9888   0.9854   0.9813   0.9924
## Specificity            0.9980   0.9965   0.9952   0.9986   0.9997
## Pos Pred Value         0.9951   0.9856   0.9775   0.9929   0.9986
## Neg Pred Value         0.9995   0.9973   0.9969   0.9963   0.9983
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2841   0.1913   0.1718   0.1608   0.1824
## Detection Prevalence   0.2855   0.1941   0.1758   0.1620   0.1826
## Balanced Accuracy      0.9983   0.9927   0.9903   0.9900   0.9960
```