Prediction Assignment

Wayne Hockensmith

April 4 2017

Excecutive Over View

Using devices such as Fitbit, Nike Fuelband, or Jawbone Up it is now possible to collect a large amount of data about personal activity relatively inexpensively. These types 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, our goal was to use data from accelerometers on the belt, forearm, arm, and dumbbells of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The data was collected from the exercises then split into two sets for this project.

- Training data set for the models: "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
- Test data set: "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

The training data set has a variable "classe" this is an A through E grade on how well the participant did the exercise. The test data does not have this grade. So the goal of this project is to predict how well the participant did the exercise in the test data set and assign them a grade. The use of any of the variables in the training data was permitted to use as predictors. Create a report describing how the model or models were built, what cross validation was used, what the expected out of sample error is, and why which choices were made. Use the prediction model to predict 20 different test cases.

Conclusion

The object of this project was to build a model that would be trained from a data set containing workouts and how the workout was graded, then predict the status of each workout of another data set that did not have the status of the work out using this model. The predicted workout status for the 20 test cases in the second data set is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
B	A	B	A	A	E	D	B	A	A	B	C	B	A	E	E	A	B	B	B

A few models were built and adjusted with all over 98% accuracy. The deciding factor on a which model to use was a model that was easy to build, change based on the number of variables and rows of data, and time it takes for the model to run. All models and methods tried gave the same above results for the 20 test cases.

Modeling and Analysis

I used the Random Forest Model 'rf' method to create a model for my final model. This model brought the predicted outcome of the test data set to 98.87892% accuracy. The model was Cross-Validated (3 fold) with sample sizes of 9159, 9158, 9157. The first Ranom Forest model I used was Cross-validated (5 fold) with a 99.17015% accuracy, however the time to run was increased by approximatly 300%, so I chose the 3 fold Cross-validation.

Other models tested but not used:

1. Parallel Random Forest.

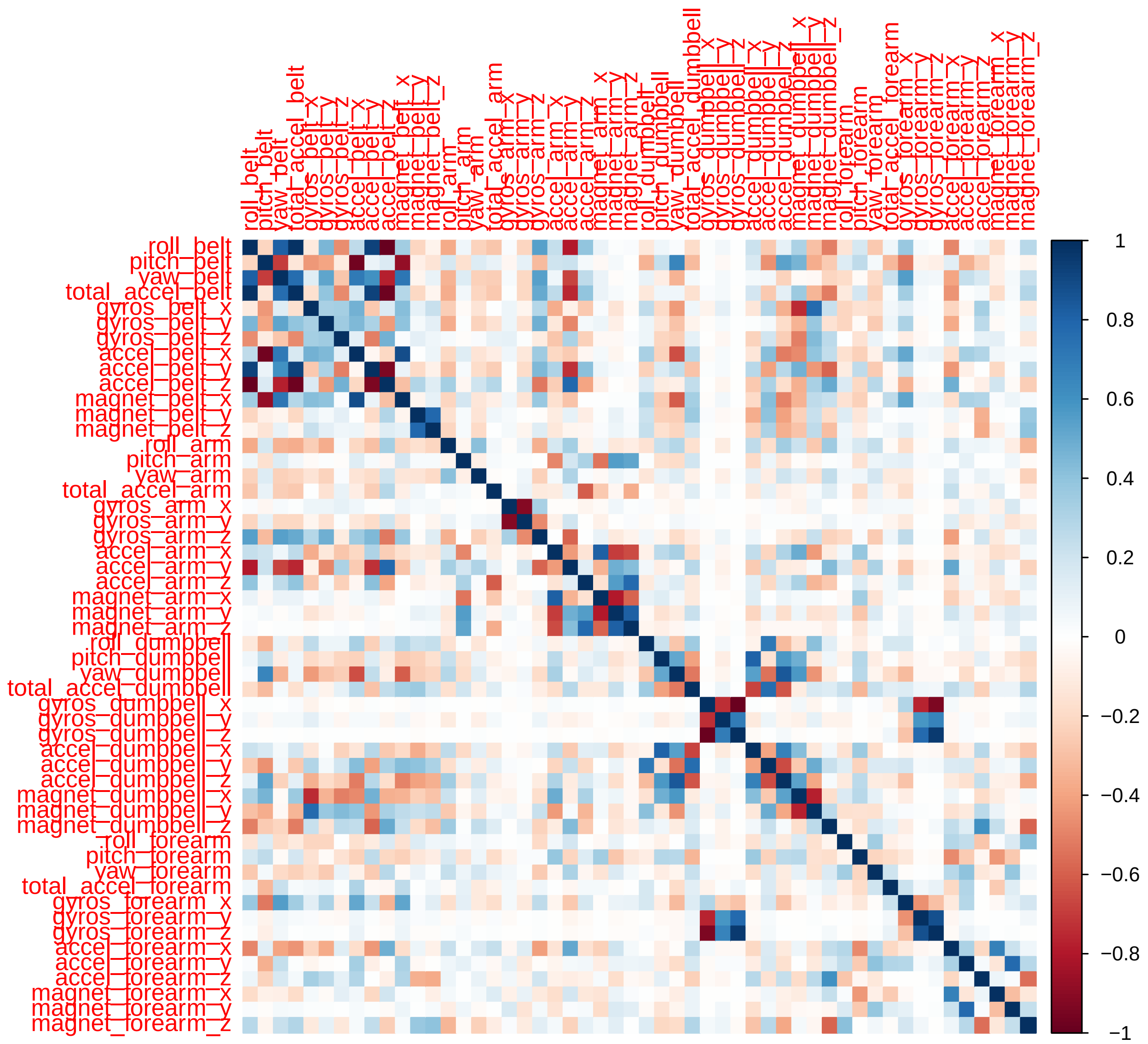
2. Random Forest Rule-Based Model.

The model was then ran through a confusion matrix to help validate the accuracy.

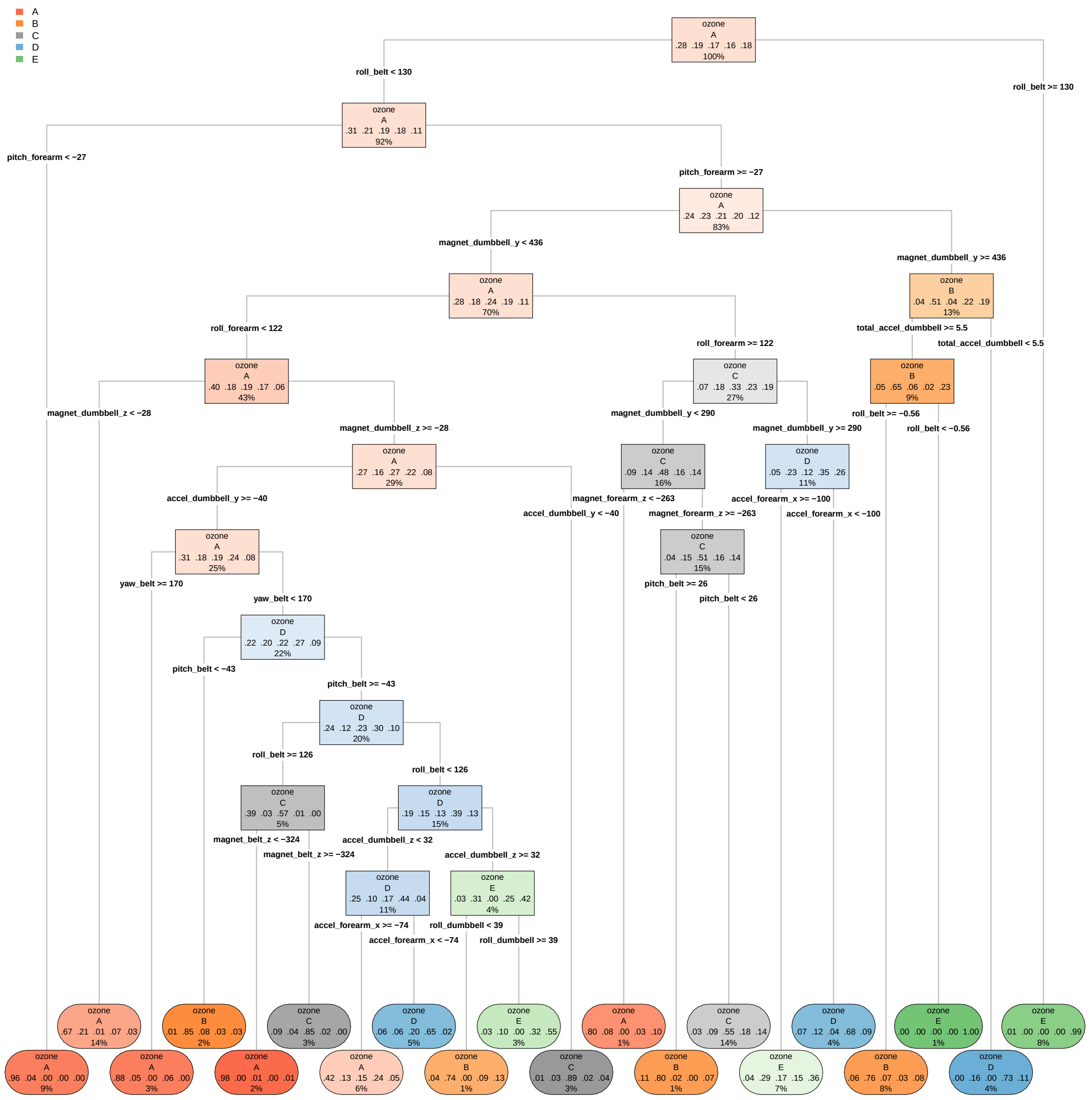
	A	B	C	D	E
A	1671	3	0	0	0
B	5	1131	3	0	0
C	0	2	1022	0	0
D	0	0	9	954	1
E	0	0	2	1	1079

Confusion Matrix Graph

The confusion matrix give an accuracy of 99.52%.



Descision Tree Graph



A
B
C
D
E
ozone A .28 .19 .17 .16 .18 100%
roll_belt < 130
roll_belt >= 130
ozone A .31 .21 .19 .18 .11 92%
pitch_forearm < −27
pitch_forearm >= −27
ozone A .24 .23 .21 .20 .12 83%
magnet_dumbbell_y < 436
magnet_dumbbell_y >= 436
ozone A .28 .18 .24 .19 .11 70%
ozone B .04 .51 .04 .22 .19 13%
roll_forearm < 122
roll_forearm >= 122
total_accel_dumbbell >= 5.5
total_accel_dumbbell < 5.5
ozone A .40 .18 .19 .17 .06 43%
ozone C .07 .18 .33 .23 .19 27%
ozone B .05 .65 .06 .02 .23 9%
magnet_dumbbell_z < −28
magnet_dumbbell_z >= −28
magnet_dumbbell_y < 290
magnet_dumbbell_y >= 290
roll_belt >= −0.56
roll_belt < −0.56
ozone A .27 .16 .27 .22 .08 29%
ozone C .09 .14 .48 .16 .14 16%
ozone D .05 .23 .12 .35 .26 11%
accel_dumbbell_y >= −40
accel_dumbbell_y < −40
magnet_forearm_z < −263
magnet_forearm_z >= −263
accel_forearm_x >= −100
accel_forearm_x < −100
ozone A .31 .18 .19 .24 .08 25%
ozone C .04 .15 .51 .16 .14 15%
yaw_belt >= 170
yaw_belt < 170
pitch_belt >= 26
pitch_belt < 26
ozone D .22 .20 .22 .27 .09 22%
pitch_belt < −43
pitch_belt >= −43
ozone D .24 .12 .23 .30 .10 20%
roll_belt >= 126
roll_belt < 126
ozone C .39 .03 .57 .01 .00 5%
ozone D .19 .15 .13 .39 .13 15%
magnet_belt_z < −324
magnet_belt_z >= −324
accel_dumbbell_z < 32
accel_dumbbell_z >= 32
ozone D .25 .10 .17 .44 .04 11%
ozone E .03 .31 .00 .25 .42 4%
accel_forearm_x >= −74
accel_forearm_x < −74
roll_dumbbell < 39
roll_dumbbell >= 39
ozone A .67 .21 .01 .07 .03 14%
ozone B .01 .85 .08 .03 .03 2%
ozone C .09 .04 .85 .02 .00 3%
ozone D .06 .06 .20 .65 .02 5%
ozone E .03 .10 .00 .32 .55 3%
ozone A .80 .08 .00 .03 .10 1%
ozone C .03 .09 .55 .18 .14 14%
ozone D .07 .12 .04 .68 .09 4%
ozone E .00 .00 .00 .00 1.00 1%
ozone E .01 .00 .00 .00 .99 8%
ozone A .96 .04 .00 .00 .00 9%
ozone A .88 .05 .00 .06 .00 3%
ozone A .98 .00 .01 .00 .01 2%
ozone A .42 .13 .15 .24 .05 6%
ozone B .04 .74 .00 .09 .13 1%
ozone C .01 .03 .89 .02 .04 3%
ozone B .11 .80 .02 .00 .07 1%
ozone E .04 .29 .17 .15 .36 7%
ozone B .06 .76 .07 .03 .08 8%
ozone D .00 .16 .00 .73 .11 4%

R Environment Setup and Model Run

Below is the final code for the Random Forest Model ‘rf’ method. I did not include the other models to conserve space and your time.

```r
library(randomForest)
library(rpart)
library(caret)
library(rpart.plot)
library(corrplot)
#library(rattle)

raw_training_data <- read.csv("pml-training.csv")
raw_test_data <- read.csv("pml-testing.csv")
```

```r
dim(raw_training_data)
dim(raw_test_data)
Head(raw_training_data)
Head(raw_test_data)
raw_training_data <- raw_training_data[,colSums(is.na(raw_training_data))==0]
raw_test_data <- raw_test_data[,colSums(is.na(raw_test_data))==0]

classe <- raw_training_data$classe
# Remove timestamps and window variables from training data set.
training_data_removed <- grepl("^X|timestamp|window",names(raw_training_data))
# Replace the raw_training_data set with the data with removed variables.
raw_training_data <- raw_training_data[,!training_data_removed]
# Ensure the data is now numeric and rename to cleaned.
training_data_cleaned <- raw_training_data[,sapply(raw_training_data, is.numeric)]
training_data_cleaned$classe <-classe

# Remove timestamps and window variables from test data set.
test_data_removed <- grepl("^X|timestamp|window",names(raw_test_data))
# Replace the raw_test_data set with the data with removed variables.
raw_test_data <- raw_test_data[,!test_data_removed]
# Ensure the data is now numeric and rename to cleaned.
test_data_cleaned <- raw_test_data[,sapply(raw_test_data, is.numeric)]

set.seed(196869) # Set for reproducable results.
# Split the training data into two, one for training and one for validating.
inTrain <- createDataPartition(training_data_cleaned$classe, p=0.70, list=F)
training_data <- training_data_cleaned[inTrain,]
test_data <- training_data_cleaned[-inTrain,]

# Direct which variable is to be trained.
classeIndex <- which(names(training_data_cleaned) == "classe")
# Find the corrolations in the variables.
correlations <- cor(training_data[, -classeIndex], as.numeric(training_data$classe))
correlations
# Determine the best correlations.
best_correlations <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.25)
best_correlations
# Set model controls and run.
RF_control <- trainControl(method="cv", 3)
RF_Model <- train(classe~., data=training_data, method="rf", trControl=RF_control, ntree=250)
RF_Model
# Predict the test_data and run through the confusion matrix.
RF_prediction <- predict(RF_Model, test_data)
confusionMatrix(test_data$classe, RF_prediction)
# What is the Accuracy of the prediction using post re-sampling.
RF_accuracy <- postResample(RF_prediction, test_data$classe)
RF_accuracy
# Run the model against the 20 test cases.
RF_result <- predict(RF_Model, test_data_cleaned[,-length(names(test_data_cleaned))])
RF_result
# Plot the Confusion Matrix and discision tree.
corrPlot <- cor(training_data[, -length(names(training_data))])
corrplot(corrPlot, method="color")
```

```
tree_graph <- rpart(classe ~ ., data=training_data, method="class")
#prp(tree_graph)
 rpart.plot(tree_graph,type=4,fallen=T, round=0, leaf.round=9,clip.right.labs=F,prefix="ozone\n", branch
#fancyRpartPlot(rxAddInheritance(tree_graph))
```