Using a Random Forest Model to Predict Human Weight Lifting Errors

Data preparation

After reading about the Human Activity Recognition (HAR) project at http://groupware.les.inf.puc-rio.br/har , training and test data were downloaded as csv files using the links provided; I changed the "classe" filename to "class". These files were read into R. The train data set had 19622 observations of 160 variables, while the test data set had 20 observations of these same variables. Examination of the two data sets showed that the first column consisted of row numbers, second was subject names, while columns 3-7 were time stamps or window codes, not predictive variables. Many of the columns with HAR motion variables had little or no data. Since one goal of the analysis was to predict the test data set, I removed the first and 3:7 columns, as well as any column that had no data present in the test data set. I tried building models with data sets that both retained and eliminated the subject variable (column 2 in the original data set); this variable made no significant difference in the ability of the models to predict the test data. If the subject variable was removed, the remaining train data set had 19622 observations of 53 variables, while the test data set had 20 observations of the same number of variables; if the subject variable was retained, there were 54 variables. For the final model I retained the subject variable, as I wanted to be able to examine the results by subjects, as well as by class. I used the complete.cases function to determine whether there were missing data, but there were not.

Model building

In order to build my model, I divided the reduced train data set by class into training and testing data sets, using a 0.7 partition in the "createDataPartition" function in the caret package. The training data set had 13737 observations, while the testing data set had 5885 observations. I then checked the relative abundance of classes in the original data set and the testing and training data sets. The proportions of classes in each data set did not differ until the fourth decimal place (Table 1).

Table 1. Comparison of class proportions in original, training and test data sets.

	A	B	C	D	E
original_Proportions	0.2843747	0.1935073	0.1743961	0.1638977	0.1838243
training_Proportions	0.2843416	0.1934920	0.1744195	0.1639368	0.1838101
testing_Proportions	0.2843416	0.1934920	0.1744195	0.1639368	0.1838101

In order to determine whether the variables remaining in the training data set had sufficient variance to build a predictive model, I checked the training data set for any variables that had near zero variance using the nearZeroVar function. All of them had sufficient variation. Using the findCorrelation function from the caret package with a correlation = 0.9, I determined whether any of the training variables were highly correlated, as highly correlated variables would increase the variance. Seven additional variables were highly correlated to other variables in the

data set; these variables were removed from the training and testing data sets, as well as the test data set to be predicted for the class assignment.

The data variable included numbers, integers and the subject factor. I could use all of these types of data in a non-parametric model, but not in a parametric model. I therefore chose to use a random forests model. In the model, I used 46 predictor variables to predict the class variable, and I chose a 10-fold cross-validation, repeated three times, to determine model accuracy. I also used 1000 trees to build the model. After the model was fitted, I used the varImp function to determine variable importance. I plotted a number of aspects of the model and variable correlations, but I do not present that data here. I then used the predict function to predict the testing data set and determine model accuracy using the confusionMatrix function. Finally, I used the predict function to predict the test data for class submission.

Results

The training data set model accuracy varied across different mtry values from 0.986 to 0.992, while the Kappa was 0.982 to 0.990; both had very low standard deviations (Table 2). This in-sample accuracy suggests that the out-of-bag (out of sample, or testing data) accuracy should be very high, so the error should be low (app. 0.01). The best accuracy was given by mtry = 26, which was the value used in the final model. Model accuracy for this model was 0.991 with a standard deviation of 0.002, suggesting an out-of-bag error of 0.009. The most important variables in the model were yaw_belt, followed by pitch_forearm, pitch_belt, magnet_dumbbell_z, magnet_dumbbell_y, roll_forearm, and magnet_belt_y (Table 3). There was a large drop in variable importance after magnet_belt_y (Table 3).

Table 2. Model accuracy, Kappa and standard deviations for model with 47 predictor variables.

mtry	Accuracy	Kappa	Accuracy SD	Kappa SD
2	0.9899063	0.9872301	0.0024054	0.0030433
26	0.9919443	0.9898095	0.0023154	0.0029292
50	0.9856830	0.9818894	0.0040886	0.0051722

Table 3. Variable importance in model building for the first 20 of 47 variables used to predict class.

Rank	Variable	Importance		Rank	Variable	Importance
1	yaw_belt	100		11	roll_dumbbell	27
2	pitch_forearm	83.47		12	magnet_dumbbell_x	25.09
3	pitch_belt	68.2		13	accel_forearm_x	23.42
4	magnet_dumbbell_z	63.74		14	total_accel_dumbbell	20.15
5	magnet_dumbbell_y	52.98		15	accel_dumbbell_z	19.56
6	roll_forearm	49.53		16	magnet_forearm_z	19.52
7	magnet_belt_y	42.74		17	magnet_belt_x	19.27
8	magnet_belt_z	31.68		18	accel_forearm_z	18.31

9	gyros_belt_z	28.53		19	total_accel_belt	18.27
10	accel_dumbbell_y	27		20	roll_arm	16.74

The final model predicted the testing data set with a 1.000 accuracy, with a 95% confidence interval of 0.9997 to 1.000; this accuracy was unusual because testing accuracy was usually lower than training accuracy, but the confidence interval for this prediction was greater than that for the training data set (CI = 0.003). The Kappa statistic for this prediction was also 1. The confusion matrix showed all classes were predicted perfectly (Table 4).

Table 4. Confusion matrix for testing data set predicted by the model trained on the training data set.

	Reference				
Prediction	A	B	C	D	E
A	3906	0	0	0	0
B	0	2658	0	0	0
C	0	0	2396	0	0
D	0	0	0	2252	0
E	0	0	0	0	2525

When the model was used to predict classes for the 20 cases in the class test set, it predicted all of them correctly.

I also tried building the model with the 7 most important variables from the initial run (1-7 in Table 3). This model had a slightly lower accuracy, but overall accuracy and Kappa were still very high (Table 5). The order of importance of the variables changed somewhat in this new model, although yaw_belt was still the most important variable. The final model was built with mtry = 4. When this new model was used to predict the testing data set, the overall accuracy was reduced to 0.9811 with 0.9773 to 0.9845 95% confidence interval; the new Kappa was 0.9761. The confusion matrix showed more errors in class predictions (Table 6), with classes B and C having 50 and 26 false negatives, respectively (Table 6). This new model, however, also predicted all of the 20 test predictions for the class submission accurately. This result, however, was not stable on re-running the model. In some re-iterations, the fourth case would be predicted incorrectly as C rather than as A.

Table 5. Model accuracy, Kappa and standard deviations for new model using the seven most important predictor variables from the original model.

mtry	Accuracy	Kappa	Accuracy SD	Kappa SD
2	0.9787432	0.9731186	0.0049934	0.0063109
4	0.9801994	0.9749597	0.0048627	0.0061466
7	0.9760011	0.9696508	0.0053095	0.0067094

Table 6. Confustion matrix for testing data set predicted by the reduced model that had the 7 most important variables from the original model.

		Reference			
Prediction	A	B	C	D	E
A	1660	10	7	1	0
B	6	1089	8	2	6
C	7	26	1000	2	2
D	0	10	11	954	3
E	1	4	0	5	1071