Practical Machine Learning Quantified Self

Philip Murphy

Sunday, March 15, 2015

Background

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. These type 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. The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har, and the associated paper is:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#wle_paper_section#ixzz3UVCiIjfV

Model Building

The goal of this project is to predict the manner in which exercises were performed, as captured in the "classe" variable in the training set. In order to develop a reasonable predictor for classe (discrete, 5-valued), first the data set had to be analyzed to separate irrelevant variables from potentially relevant ones. This analysis revealed that many of the variables had few, if any, values filled in, and variables with fewer than 50% of the values filled in were discarded. Further, after that step, it was noted that approximately 1% of the remaining cases were not complete, and so incomplete cases were also discarded. Finally, clearly irrelevant varibles such as names and sequence numbers were discarded, along time-based variables since no time sequence analysis was planned. This process produced the clean data. The first step is to read in the data sets, and set the seed for reproducibility:

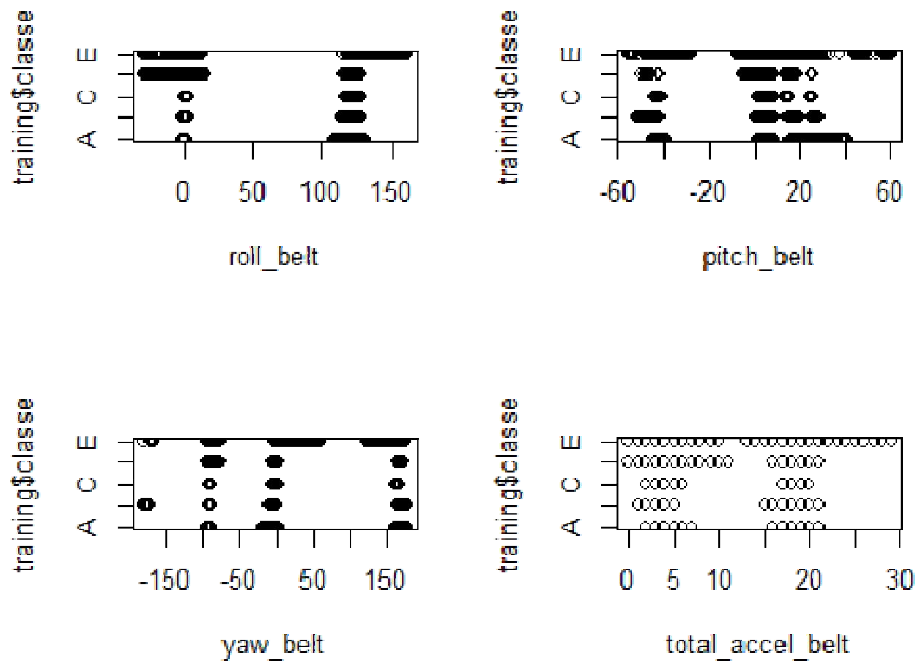
```
setwd("C:/Users/Phil/Documents/Practical Machine Learning")
library(caret)

## Warning: package 'caret' was built under R version 3.1.3

## Loading required package: lattice
## Loading required package: ggplot2

library(randomForest)

## Warning: package 'randomForest' was built under R version 3.1.3

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.

library(rpart)

## Warning: package 'rpart' was built under R version 3.1.3

training_data_set <- read.table("pml-training.csv", header=T,
sep=",",fill=T,stringsAsFactors=F,na.strings=c("","NA"))
testing_data_set <- read.table("pml-testing.csv", header=T,
sep=",",fill=T,stringsAsFactors=F,na.strings=c("","NA"))
set.seed(100)   # make this reproducible
```

Next, eliminate unwanted variables ,and then split the provided training set into training and testing parts so the training result can be checked later against the testing subset

```
clean_training <- training_data_set[,8:dim(training_data_set)[2]]  #eliminate
names, time info, other unrelated
b0 <- colSums(is.na(clean_training))/dim(clean_training)[1]<.5
clean_training <- clean_training[,b0]  # eliminate variables that are
(mostly) missing
clean_training <- clean_training[complete.cases(clean_training),]
clean_training$classe <- as.factor(clean_training$classe)
inTrain <- createDataPartition(y=clean_training$classe, p=3/4, list=F)
training <- clean_training[inTrain,]
testing <- clean_training[-inTrain,]
clean_testing <- testing_data_set[,8:dim(testing_data_set)[2]]  #eliminate
names, time info, other unrelated
clean_testing <- clean_testing[,b0]  # eliminate variables that are (mostly)
missing
clean_testing <- clean_testing[complete.cases(clean_testing),]
remove(training_data_set, testing_data_set, clean_training, inTrain)
```

Variable Selection and Cross Validation Approach

The remaining data set still has a large number of variables (52), so each variable was plotted to attempt to see if additional variables could be discarded. Below is a plot showing the relationship of four of the remaining variables to the outcome:



After carefully reviewing each of the variables it could be seen that each remaing varaible was potentially valuable and so all of them were used in the subsequent analysis. Potential model types and cross validation approaches were next evaluated. Multiple model types capable of predicting discrete outcomes were constructed, including recursive partitioning and regression trees, generalized boosted regression models, and random forests. The results of each of these trials is shown below.

The first model evaluated was a regression tree:

```
modelFitRpart <- rpart(classe~., data=training,  method="class")
predRpart <- predict(modelFitRpart, newdata=testing, type="class")
confusionMatrix(predRpart, testing$classe)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1270  213   31   99   54
##          B   28  490   45   20   46
##          C   27   69  658  112   98
##          D   18   65   55  475   38
##          E   24   92   49   80  646
##
## Overall Statistics
##
##                Accuracy : 0.737
##                  95% CI : (0.7243, 0.7494)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6647
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9290   0.5274   0.7852  0.60433   0.7324
## Specificity            0.8844   0.9641   0.9228  0.95618   0.9375
## Pos Pred Value         0.7618   0.7790   0.6826  0.72965   0.7250
## Neg Pred Value         0.9691   0.8948   0.9531  0.92508   0.9397
## Prevalence             0.2847   0.1935   0.1745  0.16368   0.1837
## Detection Rate         0.2645   0.1020   0.1370  0.09892   0.1345
## Detection Prevalence   0.3471   0.1310   0.2007  0.13557   0.1855
## Balanced Accuracy      0.9067   0.7458   0.8540  0.78025   0.8350
```

```
remove(modelFitRpart, predRpart)
```

This model demonstrated relatively poor predictive capability of less than 75% on the withheld part of the training data.

The second model evaluated was a generalized boosted regression model. The code and partial resulting confusiion matrix are included below:

fitControl <- trainControl(method="cv", number=6) # K-Fold Cross Validation, K=6, single pass modelFitGBM <- train(classe~., data=training, method="gbm", trControl=fitControl, verbose=F) # generalized boosted regression predGBM <- predict(modelFitGBM, newdata=testing) confusionMatrix(predGBM, testing$classe) remove(fitControl, modelFitGBM, predGBM)

Confusion Matrix and Statistics

Overall Statistics

```
           Accuracy : 0.9596
             95% CI : (0.9536, 0.965)
No Information Rate : 0.2847
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.9489
```

Mcnemars Test P-Value : 3.015e-06

The cross validation approach utilized was K-Fold cross validation, with K=6 folds. This was able to generate an accuracy on the witheld part of the training data of nearly 96%.

The third model evaluated was a random forest model with internal cross-validation:

```r
modelFitRF <- randomForest(classe~., data=training, method="class")
predRF <- predict(modelFitRF, newdata=testing, type="class")
confusionMatrix(predRF, testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1364    4    0    0    0
##          B    2  920    8    0    0
##          C    0    5  830   12    0
##          D    0    0    0  773    0
##          E    1    0    0    1  882
## 
## Overall Statistics
## 
##                Accuracy : 0.9931
##                  95% CI : (0.9904, 0.9953)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9913
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9903   0.9905   0.9835   1.0000
## Specificity            0.9988   0.9974   0.9957   1.0000   0.9995
## Pos Pred Value         0.9971   0.9892   0.9799   1.0000   0.9977
## Neg Pred Value         0.9991   0.9977   0.9980   0.9968   1.0000
## Prevalence             0.2847   0.1935   0.1745   0.1637   0.1837
## Detection Rate         0.2840   0.1916   0.1728   0.1610   0.1837
## Detection Prevalence   0.2849   0.1937   0.1764   0.1610   0.1841
## Balanced Accuracy      0.9983   0.9939   0.9931   0.9917   0.9997
```

The random forest model demonstrated an accuracy of over 99% and is thus the model chosen for predicting the project test data set outcome.

Expected out-of-sample error

Validation techniques are useful for both model selection and performance estimation. Striking a balance between sample data and test data can optimize the relationship between the model bias and the the model variance. The predicted error agrees very well with accuracy calculated for the held-out testing data as seen in the random forest confusion matrix.

Discussion

The goal of this project was to develop a predictor effective in identifying the manner in which exercises were performed based on a large data set. The subtasks required to create this predictor were:

- clean the data set to eliminate irrelevant variables
- review the remaining variables to identify those that were likely to assist in prediction
- fit various models capable of predicting one of multiple descrete outcomes
- select the best model for this task over several discrete factor prediction models
- build a model using cross validation
- predict out-of-sample error
- test out-of-sample error on held-out data set
- develop prediction for project testing data

After individually evaluating the variables for which there was sufficient data, the conclusion reaches was that all of the remaining 52 variables may be important contributors, so all were retained for model building. These variables were then used to fit varous discrete pretiction models including recursive partitioning and regression trees, generalized boosted regression model, and random forest. The analysis revealed that the random forest approach had the best predictive capability and was thus adopted. A part of the training data was withheld to validate the out-of-sample error prediction, and the remaining data was used to train the model. In order to balance the error bias and error variance, cross validation approach was utilized. The accuracy for out-of-sample data sets was calculated as the average of calculated error from each of the data set analyses, and yielded 99%. This was then tested against the set of data withheld, and the accuracy was very close to the prediciton. Finally, this model was used to predict the outcome for the 20 points of the project test set, with the expectation that zero or one out of the 20 predictions should be incorrect. The predictions are in the next section.

Prediction on Test Cases

The code below prints a data frame that correlates the test case to the predicted value.

```
predict(modelFitRF, newdata=clean_testing)

##  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
## Levels: A B C D E
```