Practical Machine Learning CP

Presented here is an analysis of a data set generated from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal of this analysis is to build a model/classifier that will take a set of observations as input and predict the manner (classe) in which the exercise was performed for each observation.

Loading the dataset

```
curls<-read.table("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", heade
r=T, sep=",", na.strings=c("NA", "", "#DIV/0!"))
```

Cleaning the dataset

```
#Remove any variable that contains NAs
nacount<-sapply(curls, is.na)
nacount<-as.data.frame(nacount)
totals<-sapply(nacount, sum)
ColumnsToRemove<-which(totals>0)
curls<-curls[,-ColumnsToRemove]
#Remove any variable related to time because this isn't likely to affect classe
curls<-curls[, -grep("time", names(curls))]
#Manual inspection of remaining variables suggests that the first 4 variables can also be rem
oved
curls<-curls[,-(1:4)]
```

Partition training data into a training and testing set

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

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

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

```
partition<-createDataPartition(curls$classe, p=.7, list=F)
curltrain<-curls[partition,]
curltest<-curls[-partition,]
```

A Random Forest Model

```
#We are trying to predict a categorical variable that has multiple levels.  In many cases lik
e this Random Forests are a good approach.  Therefore, the following code trains a random for
est model using 50 trees and 3-fold cross validation.  The choice of 50 trees and 3-fold cro
ss validation reduces computing time while still generating an accurate model

modFit<-train(classe~., data=curltrain, method="rf", ntree=50, number=3)
```

```
## Loading required package: 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.
```

```
#Let's take a look at the model
modFit
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9866070  0.9830475  0.002167557  0.002743705
##   27    0.9867919  0.9832823  0.001699605  0.002155581
##   52    0.9772386  0.9711891  0.004892467  0.006198734
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
#We can see that the final model chosen has an accuracy of 98.7% on the training data.  This
appears to be a very strong model.  Let's see how it performs on the test data and get an es
timation of the out of sample error rate
predictions<-predict(modFit, newdata=curltest)
confusionMatrix(predictions, curltest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671   10    0    1    0
##          B    3 1126    9    0    0
##          C    0    3 1012   10    4
##          D    0    0    5  953    6
##          E    0    0    0    0 1072
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9913
##                  95% CI : (0.9886, 0.9935)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.989
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9886   0.9864   0.9886   0.9908
## Specificity            0.9974   0.9975   0.9965   0.9978   1.0000
## Pos Pred Value         0.9935   0.9895   0.9835   0.9886   1.0000
## Neg Pred Value         0.9993   0.9973   0.9971   0.9978   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1913   0.1720   0.1619   0.1822
## Detection Prevalence   0.2858   0.1934   0.1749   0.1638   0.1822
## Balanced Accuracy      0.9978   0.9930   0.9914   0.9932   0.9954
```

```
#We can see the accuracy on the test set is very high (>99%).  This provides further evidence
for the strength of this model and suggests that the out of sample error rate will be very l
ow (<1%).  Further support for the validity of this model as a strong classifier comes from t
he 20/20 score that it acheived on the 20 test samples provided for the programming portion o
f this assigment.
```