08 Practical Machine Learning Project

Assignment

In this project, your goal will be 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.

Loading data and libraries

First we load libraries and data as required.

```r
suppressMessages(library(caret))
suppressMessages(library(dplyr))

training <- read.csv("./pml-training.csv",
                     header=T, na.strings=c("", "NA"))
testing <- read.csv("./pml-testing.csv",
                    header=T, na.strings=c("", "NA"))
```

Data cleaning

First we note that of the 160 columns of data, 100 columns have no data in the testing dataset. Since we will not be able to use these predictors to predict the testing outcomes, we will discard them in both the testing and training data. This leaves 60 predictors.

```r
nodata <- sapply(testing, function(x) {all(is.na(x))})
training <- training[, !nodata]
testing <- testing[, !nodata]
```

Pre-processing data

Next we further constrain the number of predictors. Here we pre-process using PCA, which compresses the data down to 25 principal components which account for 95% of the variance in the data. We ignore the first 6 columns of data because these are factor variables which did not come from the accelerometer data and will not be used for model building.

```r
pp <- preProcess(training[, 7:60], method = c('pca', 'center', 'scale'),
                 outcome = training$classe)
training.pp <- predict(pp, training)
testing.pp <- predict(pp, testing)
```

Now we split the training set into two datasets: `train1` and `validation`. We will use 75% of the data (`train1`) to train the model and the other 25% (`validation`) to validate the model.

```r
inT <- createDataPartition(training.pp$classe, p = 0.75, list=F)
t1 <- training.pp[ inT, ]
validation <- training.pp[-inT, ]
train1 <- t1[, -(1:6)]
```

Model fitting

Now we fit a random forests model using repeated cross-validation. We expect this to provide a high degree of accuracy for this type of high-dimensional data.

```
set.seed(1234)
rfm <- train( classe ~ ., data=train1 , method='rf',
              trControl=trainControl(method='repeatedcv', number=5, repeats=5))
```

To check the model, we use the model to predict the outcomes in the `train1` dataset. We find that the model has a 100% accuracy in this dataset (which was used to train the model).

```
t1rfm <- predict(rfm, train1)
confusionMatrix(t1rfm, train1$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We further check the model by using it to predict outcomes in the `validation` dataset. The model still has a >97% accuracy on an independent, out-of-sample dataset, so we predict an out-of-sample error rate of about 2-3%.

```
validationrfm <- predict(rfm, validation)
confusionMatrix(validationrfm, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1389    9    4    0    0
##          B    3  929   11    1    2
##          C    2   11  835   27    1
##          D    1    0    4  774    4
##          E    0    0    1    2  894
##
## Overall Statistics
##
##                Accuracy : 0.9831
##                  95% CI : (0.9791, 0.9865)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9786
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9957   0.9789   0.9766   0.9627   0.9922
## Specificity            0.9963   0.9957   0.9899   0.9978   0.9993
## Pos Pred Value         0.9907   0.9820   0.9532   0.9885   0.9967
## Neg Pred Value         0.9983   0.9949   0.9950   0.9927   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2832   0.1894   0.1703   0.1578   0.1823
## Detection Prevalence   0.2859   0.1929   0.1786   0.1597   0.1829
## Balanced Accuracy      0.9960   0.9873   0.9832   0.9802   0.9957
```

Applying the model to the testing dataset

Now we use this model on the `testing` dataset.

```
testingrfm <- predict(rfm, testing.pp)
print(testingrfm)
```

```
##  [1] B A C A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```