Machine Learning Course Project

by Steve Friedman

Note to graders: A pdf output file has been included in this github repository in case you have trouble viewing the html page online.

Overview

In this project we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The goal of the project is to predict the manner in which they did the exercise.

```
library(lattice)
library(ggplot2)
```

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

```
library(caret)
```

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

Input Data

```
# load the training data
urlfile<-
'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv'
training<-read.csv(urlfile)


#load the testing data
urlfile<-
'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv'
testing<-read.csv(urlfile)
```

Cleaning Data

A quick visual inspection of the training and testing datasets shows that columns typically have a lot of NA values or hardly any NA values. We will filter out columns that have 95% or more of their values set to NA in the training data

```
testing <- testing[, colSums(is.na(training)) < .95*nrow
(training) ]
training <- training[, colSums(is.na(training)) < .95*nrow
(training) ]
dim(training); dim(testing)
```

```
## [1] 19622    93
```

```
## [1] 20 93
```

Further inspection shows that there are still a number of columns in the testing data set that have a lot of NA values. We will now filter out the columns that have 95% or more of their values set to NA in the reduced testing data set

```
training <- training[, colSums(is.na(testing)) < .95*nrow
(testing) ]
testing <- testing[, colSums(is.na(testing)) < .95*nrow
(testing) ]
dim(training); dim(testing)
```

```
## [1] 19622    60
```

```
## [1] 20 60
```

We have now filtered the training and testing datasets down to 60 columns that do not have NAs. The first 7 columns are not directly related to the motion data and will also be removed so the prediction algorithm will work.

```
training<-training[,-1:-7] # remove columns not related to
motion data
testing<-testing[,-1:-7] # remove columns not related to
motion data
dim(training);dim(testing)
```

```
## [1] 19622    53
```

```
## [1] 20 53
```

We have now filtered the training and testing datasets down to 53 columns of motion data with no NAs.

Create Validation Data

The reduced (no NAs) training dataset will be further partitioned into a second training and a validation dataset to be used for validation of the model. Note that the original testing dataset will remain untouched and will not be used until the final predictions are done. We will put 70% of this partitioned data into the second training dataset and the other 30% into the validation dataset.

```
set.seed(12345) # make the results reproducible
inTrain <- createDataPartition(training$classe, p=0.70,
list=F)
trainData <- training[inTrain, ]
validationData <- training[-inTrain, ]
dim(trainData); dim(validationData)
```

```
## [1] 13737    53
```

```
## [1] 5885   53
```

Create Prediction Model

A random forest model with 3 fold cross validation will be created. Random Forest was used because although it is computationally intensive it is generally very accurate. Validation was limited to 3 folds to decrease the computation time.

```
controlParams <- trainControl(method="cv",3)
model <- train(classe ~ ., data=trainData, method="rf",
trControl=controlParams)
```

```
## 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.
```

```
model
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
##
## Summary of sample sizes: 9158, 9159, 9157
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9891538  0.9862781  0.003304735  0.004182271
##   27    0.9882801  0.9851741  0.002073104  0.002624672
##   52    0.9813645  0.9764251  0.002268995  0.002875528
##
## Accuracy was used to select the optimal model using  the
largest value.
## The final value used for the model was mtry = 2.
```

Model Validation

Predict the outcomes for the validation dataset and then create the confusion matrix to get an idea of the accuracy of the predictions. The confusion matrix as shown directly below looks very accurate. Most points registered on the diagoal (eg A vs A). There were a small number (less than 100) of cases where the predicted value differed from the actual value in the validation dataset. The overall accuracy of the model as indicated by the confusion matrix is 98.84%. Conversly the out of sampe error rate for the validation data set = 100% - 98.84% = 1.16%

```
prediction <- predict(model, validationData)
confusionMatrix(validationData$classe, prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B   12 1122    5    0    0
##          C    0   18 1003    5    0
##          D    0    0   24  940    0
##          E    0    0    0    3 1079
##
## Overall Statistics
##
##                Accuracy : 0.9884
##                  95% CI : (0.9854, 0.991)
##     No Information Rate : 0.2863
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9854
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D
Class: E
## Sensitivity            0.9929   0.9833   0.9719   0.9916
1.0000
## Specificity            0.9998   0.9964   0.9953   0.9951
0.9994
## Pos Pred Value         0.9994   0.9851   0.9776   0.9751
0.9972
## Neg Pred Value         0.9972   0.9960   0.9940   0.9984
1.0000
## Prevalence             0.2863   0.1939   0.1754   0.1611
0.1833
## Detection Rate         0.2843   0.1907   0.1704   0.1597
0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638
0.1839
## Balanced Accuracy      0.9963   0.9899   0.9836   0.9933
0.9997
```

Evaluation

Use the model to predict a vector of outcomes for the original test dataset

```
finalPrediction <- predict(model, testing[, -length(names
(testing))]) #remove the problem_id field from the prediction
```

The final Predictions were subsequently compared to the original test dataset. They correctly predicted the outcome in all 20 cases.