Coursera - Practical Machine Learning Course Project

Human Activity Recognition - Weight Lifting Exercise Dataset

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. 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. In this project, our 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.

Six young healthy participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different ways: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

The main goals of this project are:

1. Predict the manner in which they did the exercise depicted by the classe variable.

2. Build a prediction model using different features and cross-validation technique.

3. Calculate the out of sample error.

4. Use the prediction model to predict 20 different test cases provided.

Data retrieval, processing and transformation

This section includes of the steps to get the required data for this project, clean and process the data.

Getting the required data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

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

The test data are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv
(https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

```
trainURL<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(trainURL, "pml-training.csv",  method="curl")
download.file(testURL, "pml-testing.csv",  method="curl")
```

Loading the data

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

Processing the data

We first check for the total number of NAs in the dataset and then total NAs in each training and testing datasets.

```
sum(is.na(training))
```

```
## [1] 1921600
```

```
na_train = sapply(training, function(x) {sum(is.na(x))})
table(na_train)
```

```
## na_train
##     0 19216
##    60   100
```

```
na_test = sapply(testing, function(x) {sum(is.na(x))})
table(na_test)
```

```
## na_test
##   0  20
##  60 100
```

Looking at the above values it is clear that 60 variables have 0 NA values while the rest have NA values for almost all the rows of the dataset, so we are going to ignore them using the following code .

```
# for training dataset
columnNACounts <- colSums(is.na(training))        # getting NA counts for all columns
badColumns <- columnNACounts >= 19000             # ignoring columns with majority NA values
cleanTrainingdata <- training[!badColumns]        # getting clean data
sum(is.na(cleanTrainingdata))                     # checking for NA values
```

```
## [1] 0
```

```
cleanTrainingdata <- cleanTrainingdata[, c(7:60)] # removing unnecessary columns
```

```
# for testing dataset
columnNACounts <- colSums(is.na(testing))         # getting NA counts for all columns
badColumns <- columnNACounts >= 20                # ignoring columns with majority NA values
cleanTestingdata <- testing[!badColumns]        # getting clean data
sum(is.na(cleanTestingdata))                     # checking for NA values
```

```
## [1] 0
```

```
cleanTestingdata <- cleanTestingdata[, c(7:60)] # removing unnecessary columns
```

Now the dataset don't have any NA values. Therefore, data can be now used for some exploratory analysis and prediction model.

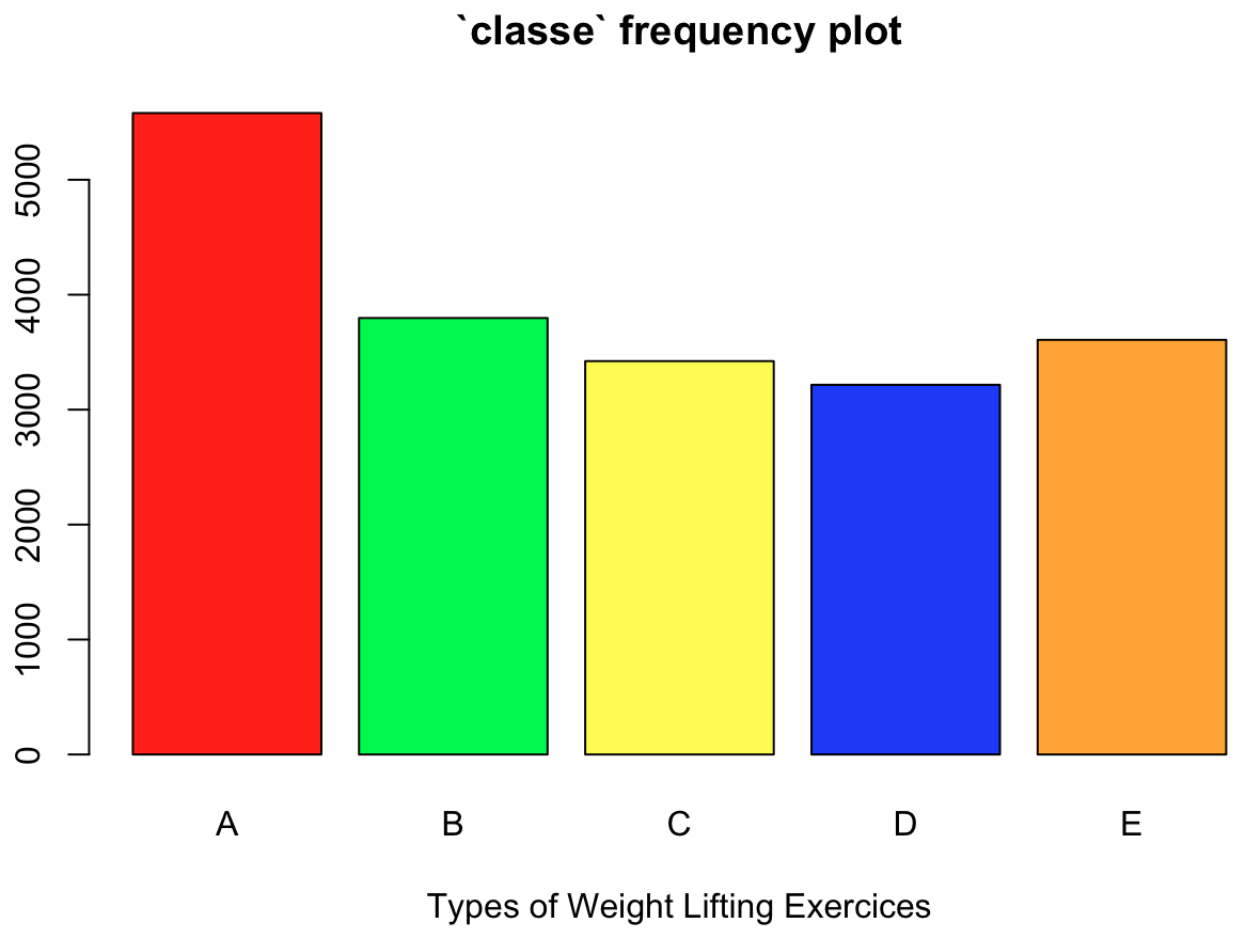
Exploratory Data Analysis

We look at some summary statistics and frequency plot for the "classe" variable.

```
summary(cleanTrainingdata$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
plot(cleanTrainingdata$classe,col=c("red", "green", "yellow", "blue", "orange"),main = "`classe
` frequency plot", xlab = "Types of Weight Lifting Exercices")
```



Model Building

In this section, we will build a machine learning model for predicting the classe value based on the other features of the dataset.

Data partitioning

We first partition the cleanTrainingdata dataset into training and testing data sets for building model

```
library (caret)
```

```
## Warning: package 'caret' was built under R version 3.1.1
```

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

```
inTrain <- createDataPartition(y=cleanTrainingdata$classe, p=0.6, list=FALSE)
trainingdata <- cleanTrainingdata[inTrain,]
testingdata <- cleanTrainingdata[-inTrain,]
dim(trainingdata)
```

```
## [1] 11776    54
```

Model building

Next, we use the features in the trainingdata dataset, we will build our model using the Random Forest machine learning technique.

```
model <- train(trainingdata$classe ~., data = trainingdata, method = "rf", prox = TRUE,
               trControl = trainControl(method = "cv", number = 4, allowParallel = TRUE))
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.1.1
```

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

```
print (model)
```

```
## Random Forest
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 8831, 8833, 8832, 8832
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      0.002        0.003
##   27    1         1      0.002        0.002
##   53    1         1      0.003        0.004
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

We build the model using 4-fold cross validation.

In sample accuracy

Now we calculate the “in sample”" accuracy which is the prediction accuracy of our model on the training data set.

```
training_pred <- predict(model, trainingdata)
confusionMatrix(training_pred, trainingdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.194    0.174    0.164    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

Thus, from the above statistics we see that the in sample accuracy value is 1 which is 100%.

Out of sample accuracy

We also calculate the “out of sample” accuracy which is the prediction accuracy of our model on the testing data set.

```
testing_pred <- predict(model, testingdata)
confusionMatrix(testing_pred, testingdata$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    3    0    0    0
##          B    0 1510    4    0    0
##          C    0    4 1364    3    0
##          D    0    1    0 1282   14
##          E    0    0    0    1 1428
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.995, 0.997)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.995    0.997    0.997    0.990
## Specificity             0.999    0.999    0.999    0.998    1.000
## Pos Pred Value          0.999    0.997    0.995    0.988    0.999
## Neg Pred Value          1.000    0.999    0.999    0.999    0.998
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.174    0.163    0.182
## Detection Prevalence    0.285    0.193    0.175    0.165    0.182
## Balanced Accuracy       1.000    0.997    0.998    0.997    0.995
```

Thus, from the above statistics we see that the out of sample accuracy value is 0.996 which is 99.6%.

Prediction Assignment

In this section, we apply the above machine learning algorithm to each of the 20 test cases in the testing data set provided.

```
answers <- predict(model, cleanTestingdata)
answers <- as.character(answers)
answers
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

Finally, we write the answers to files as specified by the course instructor using the following code segment.

```
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE, row.names = FALSE,
            col.names = FALSE)
    }
}

pml_write_files(answers)
```

On submission, a score of 20/20 can be obtained if all the predicted values are correct.

Conclusions

For this project, we chose Random Forest as our machine learning algorithm to build our model as,

1. It builds a highly accurate classifier.

2. It can handle thousands of variables.

3. It balances bias and variance trade-offs by settling for a balanced model.

4. It uses k-fold cross validation to build a robust model.

We also obtained a really good accuracy based on the statistics we obtained above.