Practical Machine Learning Project

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. In this project, 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://web.archive.org/web/20161224072740/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

Project Setup

The goal of this project is to predict how well actions are performed using the following classifications. The classifications are found in the 'classe' variable in the provided dataset.

- 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)
- throwing the hips to the front (Class E)

Prediction Methodology

Cross validation will be performed by subsampling our training data set with a 70:30 ratio. Our models will be fitted on the training data set, and tested on the validation data. Using different modelling algorithms, the most effective method will be determined. This method will then be applied on the supplied testing data set.

Project Setup

The required libraries and datasets are imported. The random number seed is set for reproducibility.

```
library(caret)
```

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

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

```
library(rpart)
library(e1071)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1)

train.raw <- read.csv('pml-training.csv', na.strings=c("NA","#DIV/0!",""))
test.raw <- read.csv('pml-testing.csv', na.strings=c("NA","#DIV/0!",""))
```

Cleaning Training Data Set

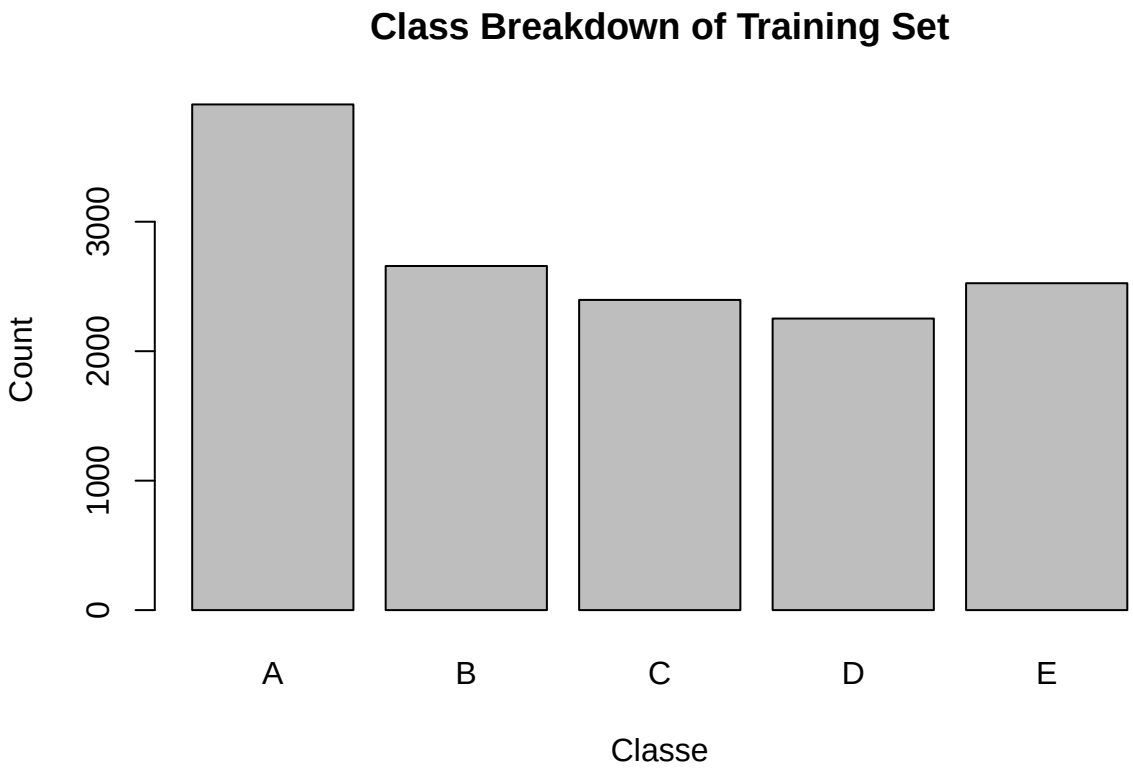
The first 7 columns contain non essential information, so they are removed. Then, columns that contain NAs are also removed.

```
train.clean <- train.raw[, 8:ncol(train.raw)]
train.clean <- train.clean[, colSums(is.na(train.clean)) == 0]
```

The training data is then randomly partioned into a training set and validation set, with a 70:30 ratio.

```
train <- createDataPartition(train.clean$classe, p=0.70, list=F)
train.clean <- train.clean[train, ]
validate <- train.clean[-train, ]
```

The distribution of the 'classe' variable is displayed in a simple bar plot.



Model 1: Decision Tree

The first model was created using the Decisioning Tree algorithm. The model is then use to predict on the validation dataset.

```
dt.model <- rpart(classe ~ ., data = train.clean, method = "class")
dt.predict <- predict(dt.model, validate, type = "class")
confusionMatrix(validate$classe, dt.predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1070   37   20   42   20
##          B  111  480   80   51   57
##          C   12   39  584   47   42
##          D   34   43  102  449   41
##          E   11   40  111   45  544
##
## Overall Statistics
##
##                Accuracy : 0.7605
##                  95% CI : (0.7471, 0.7734)
##     No Information Rate : 0.3011
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6963
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8643   0.7512   0.6511   0.7082   0.7727
## Specificity            0.9586   0.9139   0.9565   0.9367   0.9393
## Pos Pred Value         0.8999   0.6162   0.8066   0.6712   0.7244
## Neg Pred Value         0.9425   0.9523   0.9076   0.9463   0.9524
## Prevalence             0.3011   0.1554   0.2181   0.1542   0.1712
## Detection Rate         0.2602   0.1167   0.1420   0.1092   0.1323
## Detection Prevalence   0.2892   0.1894   0.1761   0.1627   0.1826
## Balanced Accuracy      0.9114   0.8325   0.8038   0.8225   0.8560
```

As seen from the confusion matrix, the decision tree model has fairly low accuracy.

Model 2: Random Forest

The second model was created using the Random Forest algorithm. The same process is applied to the validation dataset.

```
rf.model <- randomForest(classe ~ ., data = train.clean, method = "class")
rf.predict <- predict(rf.model, validate, type = "class")
confusionMatrix(validate$classe, rf.predict)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1189    0    0    0    0
##          B    0  779    0    0    0
##          C    0    0  724    0    0
##          D    0    0    0  669    0
##          E    0    0    0    0  751
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2892
##     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             0.2892   0.1894   0.1761   0.1627   0.1826
## Detection Rate         0.2892   0.1894   0.1761   0.1627   0.1826
## Detection Prevalence   0.2892   0.1894   0.1761   0.1627   0.1826
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

This model created very accurate predictions. In fact, it is possible that the model overfits the training dataset. However, for the purpose of this project, the model should provide good predictions on the testing dataset. The random forest model is still signficantly more accurate than the decision tree model. Therefore, the model will be applied to the testing dataset to measure its performance.

Final Prediction Performance

The provided testing dataset is cleaned and wrangled using the same method as the training set. Then, the random forest model is applied.

```
test.clean <- test.raw[, 8:ncol(test.raw)]
results <- predict(rf.model, test.clean[, -length(names(test.clean))])
results
```

```
##  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
```

The distribution of the predictions are plotted, as a simple comparison to the distribution of the training set.