Coursera Practical Machine Learning Assignment

Task

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

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, 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. 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

Data Pre-Processing and Cleaning

First, load the necessary libraries and datasets to be used for the analysis.

```
library(ggplot2)
library(caret)
```

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

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

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

```
## randomForest 4.7-1

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

##
## Attaching package: 'randomForest'

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

```
set.seed(10)
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
head(testing)
```

The dataset was split into training and testing partitions. 60% of the data was used for the training partition while 40% was used for the testing partition.

```
partition <- createDataPartition(y = training$classe, p = 0.6, list = FALSE)
training_partition <- training[partition,]
testing_partition <- training[-partition,]
dim(training_partition) #60% of the data is in the training_partition
```

```
## [1] 11776   160
```

```
dim(testing_partition) #40% of the data is in the testing_partition
```

```
## [1] 7846  160
```

After splitting the data into training and testing partitions, it is now important to clean the datasets. This was done by removing the near zero variance columns, columns with NA data comprising more than 80%, and columns which are just timestamps or identifiers. This is a neccessary step in order to ensure that the models will not be fed useless data. The number of columns decreased from 160 to 54 after the pre-processing steps were implemented.

```
nearlyZero <- nearZeroVar(training_partition)
filtered_training_partition <- training_partition[, -nearlyZero]
filtered_testing_partition  <- testing_partition[, -nearlyZero]

filtered_training_partition <- filtered_training_partition[, colMeans(is.na(filtered_training_partition
filtered_testing_partition <- filtered_testing_partition[, colMeans(is.na(filtered_testing_partition))

filtered_training_partition <- filtered_training_partition[, -(1:5)]
dim(filtered_training_partition)
```

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

```r
filtered_testing_partition <- filtered_testing_partition[, -(1:5)]
dim(filtered_testing_partition)
```

```
## [1] 7846   54
```

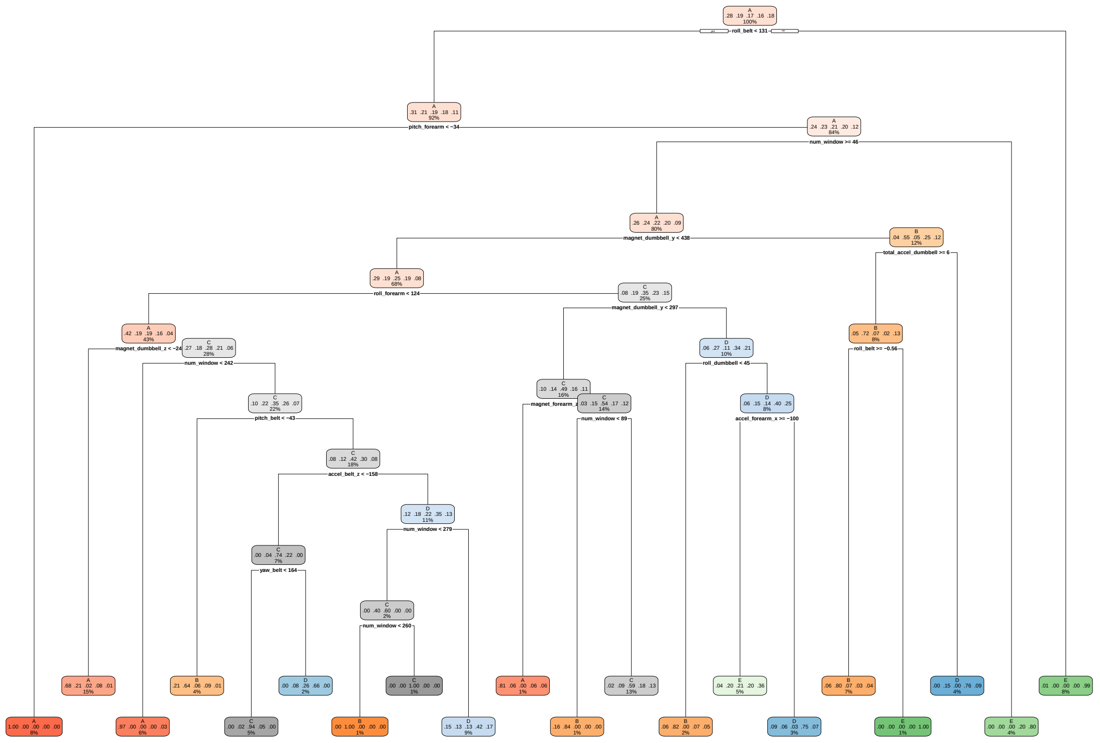
Machine Learning Algorithms

Decision Tree

The first machine learning algorithm that was implemented was the decision tree. This was done by using the rpart() function then plotting the result. After the prediction, a confusion matrix was outputted.

```r
decisiontree <- rpart(classe~., data=filtered_training_partition, method='class')
rpart.plot(decisiontree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```r
predict_decision_tree <- predict(decisiontree, testing_partition, type = "class")
confusionMatrix(predict_decision_tree, as.factor(testing_partition$classe))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1936  270   25   99   40
##          B  132  908   70   51   30
##          C   26   93 1050  210  133
##          D  104  165  148  812  168
##          E   34   82   75  114 1071
##
## Overall Statistics
##
##                Accuracy : 0.7363
##                  95% CI : (0.7264, 0.746)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6661
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8674   0.5982   0.7675   0.6314   0.7427
## Specificity            0.9227   0.9553   0.9287   0.9108   0.9524
## Pos Pred Value         0.8169   0.7624   0.6944   0.5812   0.7783
## Neg Pred Value         0.9459   0.9083   0.9498   0.9265   0.9427
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2467   0.1157   0.1338   0.1035   0.1365
## Detection Prevalence   0.3021   0.1518   0.1927   0.1781   0.1754
## Balanced Accuracy      0.8950   0.7767   0.8481   0.7711   0.8475
```

Random Forest

The next model implemented was the random forest algorithm. It will be used to compare with the performance of the decision tree algorithm.

```
filtered_training_partition$classe = factor(filtered_training_partition$classe)
randomforest <- randomForest(classe ~. , data=filtered_training_partition)
predict_random_forest <- predict(randomforest, testing_partition)
confusionMatrix(predict_random_forest, as.factor(testing_partition$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    7    0    0    0
##          B    0 1511    1    0    0
##          C    0    0 1367   12    0
##          D    0    0    0 1273    0
##          E    1    0    0    1 1442
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9972
##                  95% CI : (0.9958, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9965
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9954   0.9993   0.9899   1.0000
## Specificity            0.9988   0.9998   0.9981   1.0000   0.9997
## Pos Pred Value         0.9969   0.9993   0.9913   1.0000   0.9986
## Neg Pred Value         0.9998   0.9989   0.9998   0.9980   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1926   0.1742   0.1622   0.1838
## Detection Prevalence   0.2852   0.1927   0.1758   0.1622   0.1840
## Balanced Accuracy      0.9992   0.9976   0.9987   0.9949   0.9998
```

The output of the confusion matrix shows that the accuracy of the model is 0.9972, which is much higher compared to the 0.7363 accuracy of the decision tree algorithm implemented earlier. This is to be expected since random forests are known to be more accurate despite having problems with lower speed and overfitting. Since the random forest model has an accuracy of 0.9972, this model will be used to predict the 20 test cases.

Applying to Test Data

The random forest model was applied to the testing data and the output is shown in the results.

```r
prediction_testing <- predict(randomforest, newdata=testing)
prediction_testing
```

```
##  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
```