Human Activity Recognition

Executive Summary

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 to predict the manner in which they did the exercise.

Data Loading and Processing

We obtain the training data from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and test data from https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
#Loading required packages
library(ggplot2)
library(caret)
```

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

```
fileUrl1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrl2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(fileUrl1, destfile = "training.csv")
```

```
## Error: unsupported URL scheme
```

```
download.file(fileUrl2, destfile = "testing.csv")
```

```
## Error: unsupported URL scheme
```

```
dateDownloaded <- date()
Train_data <- read.csv("training.csv", na.strings = c("NA", "", " "))
Test_data <- read.csv("testing.csv", na.strings = c("NA", "", " "))
```

Columns 1 to 7 contain unrelated predictor and remove the columns which contains NAs. Since there are a lot of NAs (more than half) on many variables, we will eliminate those variables.

```
set.seed(1234)
Train_data1 <- Train_data[,-(1:7)]
data_NAs <- apply(Train_data1, 2, function(x) {sum(is.na(x))})
Train_data2 <- Train_data1[,which(data_NAs == 0)]
```

Exploratory Analysis

After cleaning the data set, there still are a lot of variables for this data set as shown below:

```
dim(Train_data2)
```

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

We will skip on creating the plot because of too many predictors.

Model Selection and Creation

Next we will create training set and cross validation set in the ratio of 75:25.

```
inTrain <- createDataPartition(Train_data2$classe, p = 0.75,list = FALSE)
Training <- Train_data2[inTrain,]
CrossVal <- Train_data2[-inTrain,]
```

Since this is the classification prediction. We will use random Forest algorithm due to the accuracy eventhough it will take longer to produce the result. It also uses the bagging approach and less likely to overfit the model.

```
library(randomForest)
```

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

```
modFit <- randomForest(classe ~. , data = Training)
modFit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4183    2    0    0    0   0.0004779
## B   11 2833    4    0    0   0.0052669
## C    0   12 2553    2    0   0.0054538
## D    0    0   20 2390    2   0.0091211
## E    0    1    3    7 2695   0.0040650
```

From this training set, the OOB (Out-of-Bag) estimate of error rate is 0.43% which is very small error rate. Then we can proceed with the test set.

Cross Validation

In this section, we apply the same model to the cross validation set and create the confusion matrix to see how accurate the model is.

```
# crossvalidate the model using the remaining 25% of data
predictMod <- predict(modFit, CrossVal)
confusionMatrix(CrossVal$classe, predictMod)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    2  946    1    0    0
##          C    0    9  846    0    0
##          D    0    0    8  796    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.285
##     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             0.999    0.991    0.989    1.000    1.000
## Specificity             1.000    0.999    0.998    0.998    1.000
## Pos Pred Value          1.000    0.997    0.989    0.990    1.000
## Neg Pred Value          0.999    0.998    0.998    1.000    1.000
## Prevalence              0.285    0.195    0.174    0.162    0.184
## Detection Rate          0.284    0.193    0.173    0.162    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.999    0.995    0.994    0.999    1.000
```

It shows that the model has 99.6% accuracy

Prediction

Now we will test the testing data set which is in a separare file and we will do the same data cleaning as we did with Training data set.

```
Test_data1 <- Test_data[,-(1:7)]
Test_data2 <- Test_data1[,which(data_NAs == 0)]
```

Now we predict the testing data set.

```
predictMod2 <- predict(modFit, Test_data2)
predictMod2
```

```
##  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
```

Conclusion

The training data is large enough to be able to create the accurate prediction model where it shows 99.6% accuracy in this case.