Prediction of User Behaviour Based on Accelerometer Data, by Charles Park

Introduction

This report describes the process of developing a model with machine learning, in order to predict user behaviour based on data from an accelerometer.

Setting Up the Environment

We will first load all the relevant libraries needed to conduct the analysis.

```
library(downloader)
library(dplyr)
library(ggplot2)
library(caret)
library(e1071)
```

Data Extraction and Exploration

We first load the training and test data for the study. Any attribute that has empty values will be encoded with NA values, using the na.strings argument, to facilitate pre-processing later on.

```
#urltrain = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#urltest = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#download(urltrain, dest = "./train.csv", mode = "wb")
#download(urltest, dest = "./test.csv", mode = "wb")
training = read.csv("train.csv", na.strings= c("NA","","# "))
testing = read.csv("test.csv", na.strings= c("NA","","# "))
```

To better understand how the data is structured, we will profile the training set dimensions, identify the names of all its attributes, and examine the class and value each attribute takes.

```
dim(training)
```

```
## [1] 19622   160
```

As there are 19,622 observations each having 160 variables, it is not practical to report all the names and to profile the class and value of each attribute. Using the name command below, the analyst will notice that the last variable, classe, is the outcome variable we are trying to model. We also notice that the first 7 attributes comprise identifiers for each observation, rather than variables that can help predict classe. These first 7 attributes to be removed are:

1. X (the entry number)
2. user_name
3. raw_timestamp_part_1

4. raw_timestamp_part_2
5. cvtd_timestamp
6. new_window
7. num_window

```
names(training)
```

From the str command below, the analyst will notice that there are many attributes with NA values that will not serve as suitable predictors for the classe outcome; later on, these attributes will also be removed from the formulation of the model.

```
str(training)
```

Observe that the outcome we are predicting is a categorical attribute taking one of 5 values: A, B, C, D, or E. The machine learning algorithm we employ will classify feature vectors, consisting of relevant attributes from the remaining 159 other columns to predict the letter value of classe.

```
train_results <- training$classe
table(train_results)
```

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

Data Pre-processing

As noted earlier, there are many attributes with NA. We will remove any attributes that have NA values in them, since they can skew the modelling process otherwise. We will implement code to flag which attributes have NA values and which do not.

```
NA_attr <- apply(training, 2, function(x){sum(is.na(x))})
NA_attr <- data.frame(NA_attr)
head(NA_attr)
```

```
##                       NA_attr
## X                           0
## user_name                   0
## raw_timestamp_part_1        0
## raw_timestamp_part_2        0
## cvtd_timestamp              0
## new_window                  0
```

Now we will remove all the NA attributes.As seen in the NA_attr output, the attributes with a 0 value are the ones with no NA values so we first filter for such columns. We then remove the first 7 columns that act as identification columns than predictors.

```
training <- training[, which(NA_attr == 0)]
training <- training[, -(1:7)]
```

Modeling

With the training data now prepared, we will apportion 70% of these observations as the model training set and the remaining 30% as the cross validation set. A support vector machine (SVM) was used due to its computational efficiency compared to other methods, like the random forest. SVMs are also known for good generalization performance.

```
inTrain <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
train_data <- training[inTrain, ]
cv_data <- training[-inTrain, ]
modFit <- svm(classe~., data = train_data)
```

Cross Validation and Estimation of Out-of-Sample Error

We need to estimate how effective our SVM model will be in predicting outcomes for observations that were outside the sample by applying our model on the cross validation data.

```
pred <- predict(modFit, cv_data)
accuracy <- sum(pred == cv_data$classe)/length(pred)
error <- 1-accuracy
```

The expected out of sample error is 0.0613424 based on the cross validation results.

Prediction on the Test Set

We replicate the analysis done above for the training set on the test set and then generate the predictions.

```
NA_attr_test <- apply(testing, 2, function(x){sum(is.na(x))})
NA_attr_test <- data.frame(NA_attr_test)
testing <- testing[, which(NA_attr_test == 0)]
testing <- testing[, -(1:7)]
testpred <- predict(modFit, testing)
testpred
```

```
##  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
```