MachineLearning_CourseProject

Overview

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. 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 Processing

Lets first download the training and test data set available at the following URLs.

```r
download_url <-  "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(download_url,"pml_training.csv",mode="wb")
test_download_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(test_download_url,"pml_testing.csv",mode="wb")
training_data <- read.csv("pml_training.csv")
testing_data <- read.csv("pml_testing.csv")
dim(training_data)
```

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

```r
dim(testing_data)
```

```
## [1]  20 160
```

Taking a look at the dimensions of the dataset we see that there are 160 variables. Lets do some cleanup to remove the columns having NA values.

```r
col.na <- colSums(sapply(training_data,is.na))
col.na.test <- colSums(sapply(testing_data, is.na))
train <- training_data[,col.na == 0 & col.na.test == 0]
test <- testing_data[,col.na == 0 & col.na.test == 0]
dim(train)
```

```
## [1] 19622    60
```

```r
dim(test)
```

```
## [1] 20 60
```

After cleanup we are left with 60 variables. Further removing first 5 variables which may not be related for this prediction.

```r
new_training_data <- train[,-c(1,2,3,4,5)]
new_training_data$new_window <- as.numeric(new_training_data$new_window)
```

Building the Training Model

Now lets split the training data set to 10 sets to apply cross validation.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4

## Loading required package: lattice

## Loading required package: ggplot2
```

```
set.seed(1234)
folds <- createFolds(y=new_training_data$classe,k=10,list=TRUE,returnTrain = TRUE)
sapply(folds,length)
```

```
## Fold01 Fold02 Fold03 Fold04 Fold05 Fold06 Fold07 Fold08 Fold09 Fold10
##  17661  17659  17660  17658  17660  17660  17660  17659  17660  17661
```

Using resampling method cross-validation in trainControl() function we will train the model for the first set of training data created using the above k-folds. Parallel library is used to achieve parallel processing for the execution.

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach

## Loading required package: iterators
```

```
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
model_control <- trainControl(method="cv",number=10,allowParallel = TRUE)
model_fit <- caret::train(classe~.,data=new_training_data[folds[[1]], ],method="rf",trControl=model_con
stopCluster(cluster)
registerDoSEQ()
```

Lets check the accuracy of the fitted model on the dataset using confusionMatrix().

```
confusionMatrix.train(model_fit)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.4  0.0  0.0  0.0  0.0
##          B  0.0 19.3  0.0  0.0  0.0
##          C  0.0  0.0 17.4  0.1  0.0
##          D  0.0  0.0  0.0 16.3  0.0
##          E  0.0  0.0  0.0  0.0 18.4
##
##  Accuracy (average) : 0.9981
```

The above model gives about 0.9982 accuracy which is good. Running the model in the second set of folds also gives the same accuracy.

Prediction using Test Data

Lets use the model to predict the accuracy in the test set. Before applying the model doing some cleaning up of the test data like removing the first few variables and doing the conversion of factor variables.

```r
new_test_data <- test[,-c(1,2,3,4,5)]
new_test_data$new_window <- as.numeric(new_test_data$new_window)
predict(model_fit,newdata = new_test_data)
```

```
##  [1] 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
```

Expected Out of Sample Error

Lets calculate the error rate by applying model to a new set of folds.

```r
predicted_values <- as.numeric(predict(model_fit,newdata = new_training_data[folds[[3]], ]))
actual_values <- as.numeric(new_training_data[folds[[3]], ]$classe)
(sum(predicted_values - actual_values)^2)/length(actual_values)
```

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

As you can see the out of sample error rate is relatively small since our model accuracy is 0.998.