Practical Machine Learning Course_Project

Chinmoy Das

September 3, 2017

Introduction

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. The goal of this project is to predict the manner in which they did the exercise. ## Load the necessary library

```
library (caret);
```

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

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

```
library (rpart);
library (rattle);
```

```
## Warning: package 'rattle' was built under R version 3.4.1
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library (randomForest);
```

```
## Warning: package 'randomForest' was built under R version 3.4.1
```

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

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

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

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

Load the data

We also clean all the columns that have NA's

```
pml.train <- read.csv ("D:/R/JHU/Practical Machine Learning/Course_Project/pml-training.csv",
 na.strings = c ("NA", "#DIV/0!", ""));
pml.train <- pml.train[ , (colSums(is.na (pml.train)) == 0)];
pml.test <- read.csv ("D:/R/JHU/Practical Machine Learning/Course_Project/pml-testing.csv", n
a.strings = c ("NA", "#DIV/0!", ""));
pml.test <- pml.test[ , (colSums(is.na (pml.test)) == 0)];
#all the data has been read, and the NA's have also been taken care of
```

Actual datasets contain 160 columns, which have been reduced to 60 !! ## Some pre-processing of data is needed

```
#now for the pre-processing
num_col <- which(lapply(pml.train, class) %in% "numeric");
pre_process_model <- preProcess (pml.train[ , num_col], method = c ('knnImpute', 'center', 's
cale'));
pre_process_train <- predict (pre_process_model, pml.train[ , num_col]);
pre_process_train$classe <- pml.train$classe;
pre_process_test <- predict (pre_process_model, pml.test[ , num_col]);
```

Removing the NEAR-ZERO VALUES

```
#removing the near-zero variables
#training_data
nzv <- nearZeroVar (pre_process_train, saveMetrics = T);
pre_process_train <- pre_process_train[ , (nzv$nzv == F)]
#testing_data
nzv <- nearZeroVar (pre_process_test, saveMetrics = T);
pre_process_test <- pre_process_test[ , (nzv$nzv== F)];
rm (nzv);
```

Creating the data_sets required

```
#creating the training & testing dataset
set.seed (10000007);
inTrain <- createDataPartition (pre_process_train$classe, p = 0.7, list = F)
training <- pre_process_train[inTrain, ];
testing <- pre_process_train[-inTrain, ];
```

seed is also set, so as to keep up with reproducibility ##Model Fitting using Random_Forest

```
#model fitting using random_foresting
model_rf <- train (classe ~., method = "rf", data = training,
                    trControl = trainControl (method = 'cv'), number = 5, allowParallel = T,
importance = T);

#prediction on testing_data
res_test <- predict (model_rf, testing);
print (confusionMatrix (res_test, testing$classe));
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1132    3    1    3
##          C    0    5 1014    6    2
##          D    0    0    9  956    2
##          E    0    0    0    1 1075
##
## Overall Statistics
##
##                Accuracy : 0.9942
##                  95% CI : (0.9919, 0.996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9927
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9939   0.9883   0.9917   0.9935
## Specificity            0.9995   0.9985   0.9973   0.9978   0.9998
## Pos Pred Value         0.9988   0.9939   0.9873   0.9886   0.9991
## Neg Pred Value         1.0000   0.9985   0.9975   0.9984   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1924   0.1723   0.1624   0.1827
## Detection Prevalence   0.2848   0.1935   0.1745   0.1643   0.1828
## Balanced Accuracy      0.9998   0.9962   0.9928   0.9947   0.9967
```

Accuracy Checking && Out_of_sample Errors checking

```
resample_accuracy <- postResample (res_test, testing$classe)
accuracy <- resample_accuracy[[1]];
out_of_sample_error <- 1 - accuracy;
print (paste ("Accuracy = ", accuracy));
```

```
## [1] "Accuracy =  0.994222599830076"
```

```
print (paste ("Out of sample error = ", out_of_sample_error));
```

```
## [1] "Out of sample error =  0.00577740016992356"
```

```
#rf -- accuracy ~ 99.42%
#out_of_sample_error -- 0.57%
```

I’ve even mentioned the results, that I got in the initial run !! ##Working on the final model !!

```
#final prediction
res_final <- predict (model_rf, pml.test);
print (res_final);
```

```
##  [1] E B B A A E E B B E B B B B E E E B E E
## Levels: A B C D E
```

```
#E B B A A E E B B E B B B B E E E B E E
#Levels: A B C D E
```

I’ve also mentioned the final results, that I got in the initial run !!

Submitted on Github

Thank you for your patience !!