PML_assignment

Aditya

June 19, 2016

This project aims to predict the quality of exercises performed by 6 individuals who performed these exercises correctly and incorrectly. The information was recorded from accelerometers placed on the belt, forearm, arm and dumbell. The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Loading the necessary packages needed for the analysis.

```
rm(list=ls())
library(AppliedPredictiveModeling)
library(randomForest)
```

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

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

```
library(caret)
```

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

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

```
##
## Attaching package: 'ggplot2'
```

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

Read in the data files while stating the different kinds of NA values in the data.

```
training <- read.csv("/export/faraday/ak.kasinadhuni/PML_coursera/pml-training.csv",
header=T, as.is=T, stringsAsFactors =F, sep=",", na.strings=c("NA", "#DIV/0!",""))

testing <- read.csv("/export/faraday/ak.kasinadhuni/PML_coursera/pml-testing.csv", he
ader=T, as.is=T, stringsAsFactors =F, sep=",", na.strings=c("NA", "#DIV/0!",""))
```

Cleaning the training and testing datasets.

```
na_training <- apply(training, 2, function(x){sum(is.na(x))})
clean_train <- training[,which(na_training ==0)]

na_testing <- apply(testing, 2, function(x){sum(is.na(x))})
clean_test <- testing[,which(na_testing ==0)]
```

We will now process the data and include columns which possess "numeric" values. The other data will not be useful in our prediction algorithm.

```
features <- which(lapply(clean_train,class) %in% "numeric")
objects = preProcess(clean_train[,features],method=c('knnImpute', 'center', 'scale'))
train <- predict(objects, clean_train[,features])
train$classe <- clean_train$classe

test <- predict(objects, clean_test[,features])
```

We will now perform cross validation by separating the training data into two parts one of which will be our train control.

```
set.seed(1234)

df_train <- createDataPartition(train$classe, p=0.75, list=F)
training <- train[df_train,]
validation <- train[-df_train,]
```

The train model will now be created using RandomForest algorithm. We have already loaded the randomforest libary into R.We will then save the prediction to a file.

```
prediction <- train(classe ~. , method="rf", data = training, trControl=trainControl(
method="cv"), number=5, allowParallel=T)
save(prediction,file="/export/faraday/ak.kasinadhuni/PML_coursera/fit.R")
```

We will now perform the predictions on the training and validation datasets to understand the level of accuracy in our model.

```
predict_training <- predict(prediction, training)
confusionMatrix(predict_training, training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
predict_cv <- predict(prediction, validation)
confusionMatrix(predict_cv, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    3    0    0    0
##          B    0  944   12    0    0
##          C    0    2  836    6    1
##          D    0    0    7  798    0
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.991, 0.9957)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9778   0.9925   0.9989
## Specificity            0.9991   0.9970   0.9978   0.9983   1.0000
## Pos Pred Value         0.9979   0.9874   0.9893   0.9913   1.0000
## Neg Pred Value         1.0000   0.9987   0.9953   0.9985   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1705   0.1627   0.1835
## Detection Prevalence   0.2851   0.1949   0.1723   0.1642   0.1835
## Balanced Accuracy      0.9996   0.9958   0.9878   0.9954   0.9994
```

We now estimate the out of sample error:

```
oos_error = 1 - 0.9937
oos_error
```

```
## [1] 0.0063
```

Predict on the testing data.

```
predict_test = predict(prediction, test)
predict_test
```

```
##  [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
```