ML-project

Data Analyst

September 29, 2018

Background

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. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

Getting and cleaning the data

Use sapply() R function to clean our data by removing the variables that has missing values

```
miss_value <- sapply(pml_testing, function (x) any(is.na(x) | x == ""))
test_dataset<-pml_testing[!miss_value]
miss_value <- sapply(pml_training, function (x) any(is.na(x) | x == ""))
training_dataset<-pml_training[!miss_value]
```

Data Analysis

Split the data into training and testing

we will set aside a subset of our training data for cross validation (30%).

```
#Loading necessary libraries for data analysis
library(caret)
library(lattice)
library(ggplot2)
train <- createDataPartition(y=training_dataset$classe, p=0.7, list=FALSE)
train_data <- training_dataset[train, ]
test_data <- training_dataset[-train, ]
train_data$X<- NULL
test_data$X<- NULL
```

MAchine learning: Classification model

It seems we need to use Random forest model to classify our data ###Concept of our work The process works as follow: * Build (train) the model on the training data set *Apply the model to the test data set to predict the outcome of new unseen observations* Quantify the prediction error as the mean squared difference between the observed and the predicted outcome values. ###Definition Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. (Wikipidea)

```
#Load necessary libraries
library(AppliedPredictiveModeling)
library(rpart)
library(rattle)
library(randomForest)
set.seed(232323)
#model<- train(user_name~. , data=train_data, method = "rf")
model<-randomForest(classe~., data = train_data)
model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.12%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3906    0    0    0    0 0.0000000000
## B    2 2655    1    0    0 0.0011286682
## C    0    3 2389    4    0 0.0029215359
## D    0    0    3 2248    1 0.0017761989
## E    0    0    0    2 2523 0.0007920792
```

```
prediction <- predict(model, test_data, type="class")
confusionMatrix(prediction, test_data$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

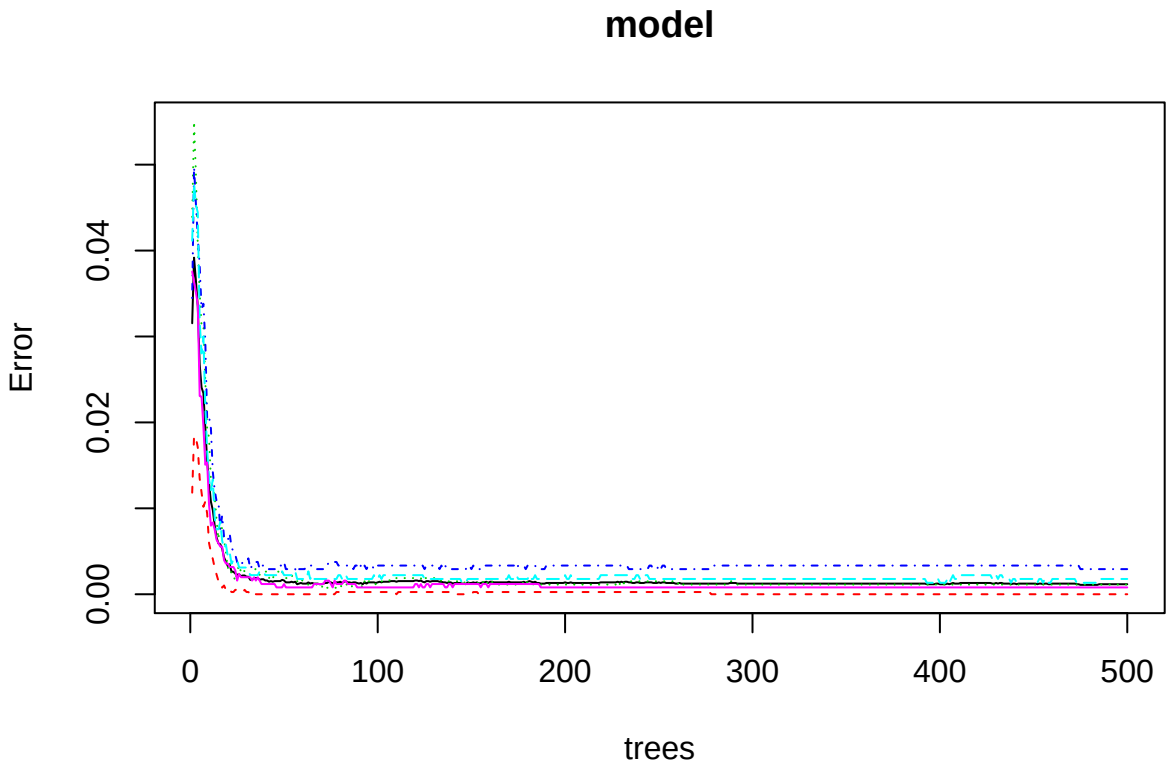
```
##          A 1672    0    0    0    0
##          B    2 1139    6    0    0
##          C    0    0 1016    0    0
##          D    0    0    4  962    0
##          E    0    0    0    2 1082
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   1.0000   0.9903   0.9979   1.0000
## Specificity            1.0000   0.9983   1.0000   0.9992   0.9996
## Pos Pred Value         1.0000   0.9930   1.0000   0.9959   0.9982
## Neg Pred Value         0.9995   1.0000   0.9979   0.9996   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1935   0.1726   0.1635   0.1839
## Detection Prevalence   0.2841   0.1949   0.1726   0.1641   0.1842
## Balanced Accuracy      0.9994   0.9992   0.9951   0.9986   0.9998
```

Summary

The random forest algorithm performs well by giving 0.992 accuracy which we can assume as our in sample error.

Plot the result

```
plot(model)
```



References

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har