Machine_Learning_Project

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/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 (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

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

Analysis

Loading libraries.

```
# install.packages("randomForest")
# install.packages("caret")
# install.packages("corrplot")
# install.packages("rpart")
# install.packages('e1071', dependencies=TRUE)

library(kernlab)
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.5.3
```

```
## corrplot 0.84 loaded
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

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

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## randomForest 4.6-14
```

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

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

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

Loading data.

```
url1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

destfile1 <- "./ML_proj/pml-training.csv"
destfile2 <- "./ML_proj/pml-testing.csv"

if (!file.exists("ML_proj")) {dir.create("ML_proj")}
download.file(url1 , destfile = destfile1)
download.file(url2 , destfile = destfile2)
```

Read data

```
train <- read.csv("./ML_proj/pml-training.csv", na.strings = c("NA", " ", ""))
test <- read.csv("./ML_proj/pml-testing.csv",na.strings = c("NA", " ", "") )
```

The dataset contains lot of missing values.

The following code is to clean the data from missing values columns.

```
# keep columns with no NAs:
train_NA_ct <- apply(train, 2, function(x){sum(is.na(x))})
train_clean <- train[, which(train_NA_ct==0)]

test_NA_ct <- apply(test, 2, function(x){sum(is.na(x))})
test_clean <- test[, which(test_NA_ct==0)]
```

Remove columns that do not contribute to the experiment

```
train_regex <- grepl("user_name|timestamp|window|^X", names(train_clean))
train_clean1 <- train_clean[, !train_regex]

test_regex <- grepl("user_name|timestamp|window|^X", names(test_clean))
test_clean1 <- test_clean[, !test_regex]

dim(train_clean1)
```

```
## [1] 19622    53
```

```
dim(test_clean1)
```

```
## [1] 20 53
```

Model development

Partitioning the data into a 70:30 training and cross-validation datasets, to check for model over-fitting.

```
set.seed(23456)
inTrain <- createDataPartition(train_clean1$classe, p=0.70, list = FALSE)
traing <- train_clean1[inTrain, ]
val <- train_clean1[-inTrain, ]
dim(traing)[1]
```

```
## [1] 13737
```

```
dim(val)[1]
```

```
## [1] 5885
```

Fit the model with a Random Forest algorithm

```
model <- randomForest(classe ~ ., data = traing)
model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = traing)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3903    3    0    0    0 0.0007680492
## B   14 2638    6    0    0 0.0075244545
## C    0   12 2382    2    0 0.0058430718
## D    0    0   25 2225    2 0.0119893428
## E    0    0    3    5 2517 0.0031683168
```

The OOB estimate of error rate is 0.52%.

Performance analysis

```
predictVal <- predict(model, val)
confusionMatrix(val$classe, predictVal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    3    0    0    1
##          B    3 1136    0    0    0
##          C    0    4 1022    0    0
##          D    0    0    5  959    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9971
##                  95% CI : (0.9954, 0.9983)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9963
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9939   0.9951   0.9990   0.9991
## Specificity            0.9991   0.9994   0.9992   0.9990   0.9998
## Pos Pred Value         0.9976   0.9974   0.9961   0.9948   0.9991
## Neg Pred Value         0.9993   0.9985   0.9990   0.9998   0.9998
## Prevalence             0.2843   0.1942   0.1745   0.1631   0.1839
## Detection Rate         0.2838   0.1930   0.1737   0.1630   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9986   0.9966   0.9972   0.9990   0.9994
```

The model accuracy is 99.71%, which is quiet robust.

Apply the random Forest model to the test data.

```
predictTest <- predict(model, test_clean1)
predictTest
```

```
##  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
```

Conclusion

A random forest model would accurately predict the mannerin which a person did the exercise.

Appendix

Correlation matrix of the variables in the training dataset:

```
corr<- cor(traing[ , -length(names(traing))])
corrplot(corr, method = "color")
```