Human Activity Recognition

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. In general, there are many potential applications for Human Activity Recognition, like elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

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. The goal of the project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

References

For more details see http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har)

Research question

The research question for this project can be expressed like:

- Is it possible to predict the manner in which a person perform barbell from his/her movements?

In more details:

- Is it possible to predict the classe variable from accelerometers measures variables?

1. Data preprocessing

Libraries

For this analysis, we will use the caret package (Classification And REgression Training) that contains a complete set of functions to create predictive models like data splitting, pre-processing, feature selection, training and testing models.

```
library(caret)
```

Data loading

The training and test data used for the project are available at

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

respectively.

After downloading the two files, we read them and load two R data sets.

```
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")
```

Data cleaning

In the data set there are 67 variables contain 19216 NA values and other 33 variables contain 19216 empty strings.

```
t <- table(colSums(is.na(train)))
t
```

```
##
##     0 19216
##    93    67
```

```
table(colSums(train == ""))
```

```
##
##     0 19216
##    60    33
```

It is possible to ignore these features because the 0.9793% of their values are not significant. Other features in the dataset are not measurements, like X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp and new_window and num_window, and so they can be ignored too. All these features have so been removed from the dataset.

```
index <- which(colSums(is.na(train)) == 0 & colSums(train == "") == 0)
train <- train[,index] # remove features with NA and "" values
train <- train[, 7:60] # remove non significant features
```

2. Model fitting

Cross Validation

Cross validation technique is used to limit problems like overfitting and obtain a high out of sample accuracy and consequently a low out of sample error (we will use accuracy as error measure). Therefore, first the training dataset is partitioned in a training and a test set using the rule of thumb of 70/30%. Then the model is built on this training set and finally evaluated on the test set.

```
set.seed(8503)
partition <- createDataPartition(y = train$classe, p = 0.6, list = FALSE)
training <- train[partition, ]
testing <- train[-partition, ]
```

Model training

The model for this analysis is built using the random forest algorithm because it can manage very well data with a great amount of features reaching at the same time a very high accuracy. The trControl function is used with cross validation and parallel processing.

```
set.seed(4567)
trControl = trainControl(method = "cv", number = 4, allowParallel = TRUE)
model <- train(classe ~ ., data = training, method = "rf", prox = TRUE,
               trControl = trControl)
model
```

```
## Random Forest
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 8833, 8832, 8831, 8832
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    1         1      0.002        0.003
##   27    1         1      0.001        0.002
##   53    1         1      0.001        0.002
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

It is possible to see that the accuracy of the model chosen is 100%, a very accurate model. This is the in sample accuracy, the accuracy obtained on the same data the model is built, and it is always optimistic. More important is the out of sample accuracy, the accuracy on a new data set, a test set to predict, that is expected to be generally lower than the in sample accuracy due to overfitting. Cross validation is used to avoid overfitting and obtain a good out of sample accuracy too.

Model evaluation

Prediction and evaluation are made on the test set giving an accuracy of 99.7% that is an extremely good level of accuracy as we expected from cross-validation.

```
pred <- predict(model, testing)
confusionMatrix(pred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    5    0    0    0
```

```
##          B    1 1512    7    0    2
##          C    0    0 1361    9    0
##          D    0    1    0 1277    2
##          E    0    0    0    0 1438
##
## Overall Statistics
##
##                Accuracy : 0.997
##                  95% CI : (0.995, 0.998)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.996
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.996    0.995    0.993    0.997
## Specificity             0.999    0.998    0.999    1.000    1.000
## Pos Pred Value          0.998    0.993    0.993    0.998    1.000
## Neg Pred Value          1.000    0.999    0.999    0.999    0.999
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.173    0.163    0.183
## Detection Prevalence    0.285    0.194    0.175    0.163    0.183
## Balanced Accuracy       0.999    0.997    0.997    0.996    0.999
```