Coursera - Human Activity Recognition

Synopsis

This project applies human activity recognition to the weight lifting domain. Six participants taught by a professional and equiped with measurement captors were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The goal is to use machine learning to build a model from a dataset of measurments provisioned by the set of captors installed on the participants, and to use it to predict or classify the correctness of the same exercises executed by anyone without the support of a professional of the domain. The measure of correctness or the outcome of the predictions is given by the class category variable which values belong to the enumeration A,B,C,D and E.

Data processing

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)

- The training data can be downloaded from here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- The testing data can be downloaded from here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Loading

The training and testing data from the given CSV text files:

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

The training set has 160 variables and 19622 observations and the testing set has 160 variables and 20 observations.

Missing data diagnosis

Missing data can be detected in several predictors of the training and the testing datasets and includes all their observations.

```
range(colSums(is.na(training)))
```

```
## [1]     0 19216
```

```
hist(colSums(is.na(training)), plot = F)$counts
```

```
##  [1]  60   0   0   0   0   0   0   0   0 100
```

```
range(colSums(is.na(testing)))
```

```
## [1]  0 20
```

```
hist(colSums(is.na(testing)), plot = F)$counts
```

```
##  [1]  60   0   0   0   0   0   0   0   0 100
```

Consequently a removal of these predictors is preferred to any data imputation. Only the sane predictors are considered.

```
training <- training[,colSums(is.na(training)) == 0]
testing <- testing[,colSums(is.na(testing)) == 0]
```

Cleaning

The predicting power of predictors based on identity and timing features is very low. Only the predictors having a significant predicting power on the outcome are considered.

```
names(training)[1:7]
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"
```

```
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
```

The training set has now 53 variables and 19622 observations and the testing set has now 53 variables and 20 observations.

Training and testing have the same 53 variables but the last one, respectively classe and problem_id. The size of the training data is significantly larger than the testing data one.

Splitting

A split of the cleaned training set into a training dataset and a validation dataset allows to compute out-of-sample errors during model building evaluation. A 70% / 30% partition is considered and a fix seed is required for reproduceability.

```
set.seed(33833)
inTrain <- createDataPartition(training$classe, p=0.7, list = F)
training_data <- training[inTrain,]
validation_data <- training[-inTrain,]
```

Prediction

The outcome to be predicted is the **classe** variable of the training dataset. A model based on the other variables of the training dataset aims in modelling the behaviour of the data and in predicting the outcome when applied first of all to the validation dataset and finally to the testing data.

Non-linear model buiding

Classification trees and Random forests, both with K-fold cross validation, are candidate techniques for this classification prediction problem.

Random forests

The Random forests algorithm applied to the training dataset using a 5-fold cross validation:

```
control <- trainControl(method = "cv", number = 5)
model_fit_rf <- train(classe ~ ., data = training_data, method = "rf", trControl = co
ntrol)
model_fit_rf$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9907553 0.9883039 0.002249986 0.002846600
## 2   27 0.9906093 0.9881209 0.001244782 0.001573247
## 3   52 0.9838390 0.9795558 0.001227610 0.001554533
```

The prediction applied to the validation dataset followed by its evalution using the confusion matrix:

```
prediction_rf <- predict(model_fit_rf, validation_data)
confusion_rf <- confusionMatrix(validation_data$classe, prediction_rf)
confusion_rf$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    1 1137    1    0    0
##          C    0   18 1008    0    0
##          D    0    0   25  937    2
##          E    0    0    4    3 1075
```

```
confusion_rf$overall[1]
```

```
##  Accuracy
## 0.9904843
```

The accuracy of the prediction and the out-of-sample error (0.0095157) are pretty good, although the computation of the algorithm is not efficient and takes a long time.

Classification trees

Classification tree algorithm applied to the training dataset using a 5-fold cross validation:

```
model_fit_rpart <- train(classe ~ ., data = training_data, method = "rpart", trContro
l = control)
model_fit_rpart$results
```

```
##           cp  Accuracy      Kappa AccuracySD    KappaSD
## 1 0.03722917 0.5128475 0.36415374 0.02031529 0.02753661
## 2 0.06133659 0.4435392 0.25428003 0.06697676 0.11252072
## 3 0.11484081 0.3319451 0.07275564 0.04353741 0.06651522
```

Prediction applied to the validation dataset followed by its evalution using the confusion matrix:

```
prediction_rpart <- predict(model_fit_rpart, validation_data)
confusion_rpart <- confusionMatrix(validation_data$classe, prediction_rpart)
confusion_rpart$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1517   26  128    0    3
##          B  484  369  286    0    0
##          C  494   28  504    0    0
##          D  435  172  357    0    0
##          E  162  135  294    0  491
```

```
confusion_rpart$overall[1]
```

```
##  Accuracy
## 0.4895497
```

The accuracy of the prediction based on a classification tree is very low compared to the one obtained using the Random forests method.

Testing set

The resulting prediction model based on the Random forests algorithm applied to the testing dataset:

```
predict(model_fit_rf, testing)
```

```
##  [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
```

Conclusion

The Random forests method applied to this dataset returns far better prediction results than the classification tree method, but is computationaly intensive. A usual 10-fold cross validation is not necessary and would make the execution time even longer.