Qualitative Activity Recognition

Synopsis

The goal of your project is to predict the manner. The training data for this project are pml-training.csv available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv and the test data are pml-testing.csv available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv. The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Data Preprocessing

1. Set up the working directory.

2. Load the training and testing data, and perform elementary data analysis.

3. Preprocess the data, and extract the useful feature. For this purpose, select the columns which have most of the entry are NA and blank, and filter out the training and test data set and build the validate data set that use to train the model.At this phase, perform the elementary data analysis on newly build training data set.

4. For implementing Machine Learning algorithm load the “caret” package.

5. Partition the tidy training data using “createDataPartition()” function and perform basic operations.

6. Now find out the less useful or useless predictor from the tidy training data set, and update the training data set.

7. Fit a model on the training data set i.e apply “train()” function where method is random forest algorithm (mehtod = “rf”). In order to speed up the execution trControl parameter of the “train” function is used.

8. Print the fitted model and check out the accuracy of the model.

9. Predict the classe of each instance of the reshaped test data set by using “prediction” function of the caret package.

10. estimate out of sample error appropriately with cross-validation

11. write up the predicted character vector to the “.txt” files

```
# setup directory
setwd("F:/Data Science/8. Pratical Machine
Learning/Week3/Assessment/Project")
getwd()
```

```
## [1] "F:/Data Science/8. Pratical Machine
Learning/Week3/Assessment/Project"
```

Load the data set and check the dimension

```
# load the data set and check the dimension
trainRawData <- read.csv("pml-training.csv", na.strings =
c("NA", ""))
dim(trainRawData)
```

```
## [1] 19622   160
```

To eliminate the columns where most of the entry are NA values find out the NAs, build a new data set , and check out the dimension.

```
# discard NA
NAs2 <- apply(trainRawData, 2, function(x) {
    sum(is.na(x))
})

# build new training data set
validData2 <- trainRawData[, which(NAs2 == 0)]
dim(validData2)
```

```
## [1] 19622    60
```

Load the “caret” package, partition the training data set, get the training data, and check the dimension and View the data set.

```
# make trianing set
library(caret)
training <- createDataPartition(y = validData2$classe, p =
0.7, list = FALSE)
trainData <- validData2[training, ]
dim(trainData)
```

```
## [1] 13737    60
```

```
testValidateData <- validData2[-training, ]
dim(testValidateData)
```

```
## [1] 5885   60
```

Take a manual look up over the data set and discard the useless predictors. Then, we are ready to use the training data to train the model.

```
# discard useless predictors
removeIndex <- grep("timestamp|X|user_name|new_window",
names(trainData))

trainData <- trainData[, -removeIndex]
dim(trainData)
```

```
## [1] 13737    54
```

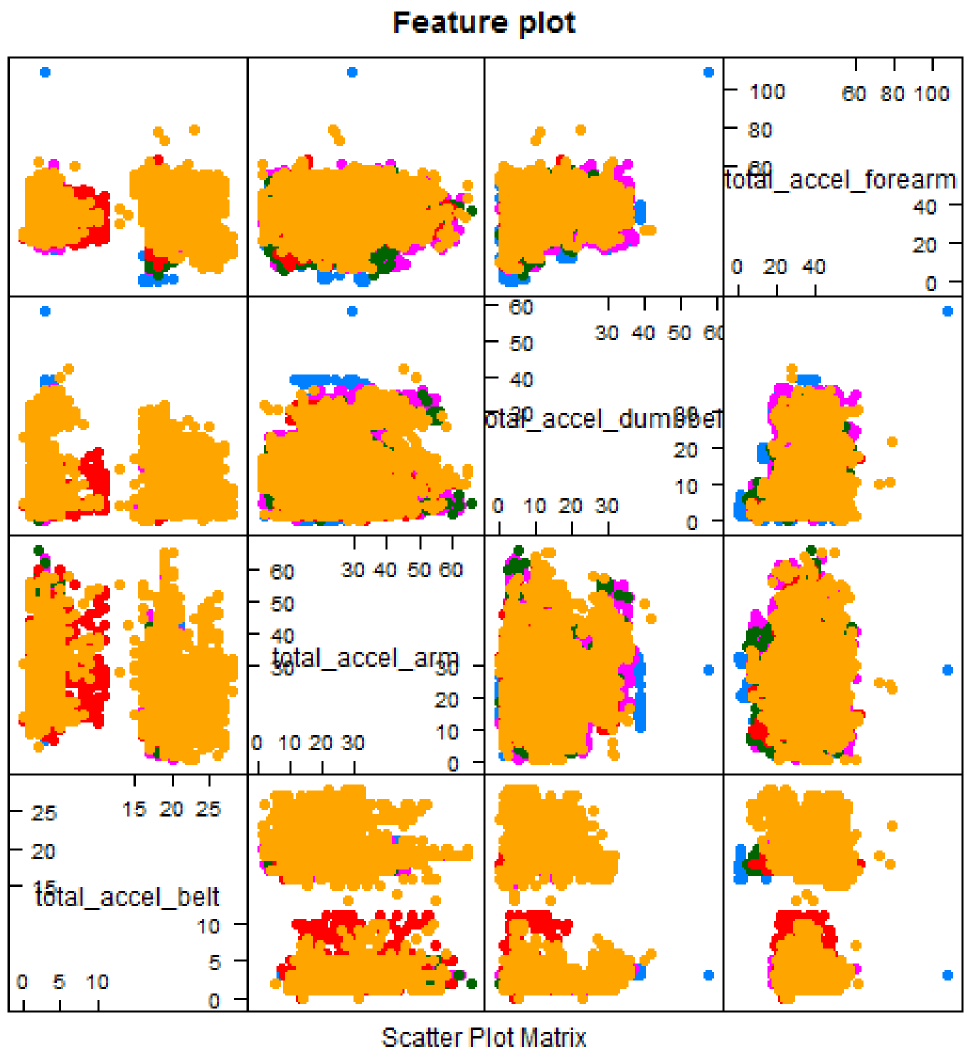
```
testValidateData <- testValidateData[, -removeIndex]
dim(testValidateData)
```

```
## [1] 5885   54
```

```
# plot features
total <- which(grepl("^total", colnames(trainData),
ignore.case = F))

totalAccel <- trainData[, total]

featurePlot(x = totalAccel, y = trainData$classe, pch = 19,
main = "Feature plot",
    plot = "pairs")
```



Build the prediction model using the training data where “classe” is the outcome and other features are as predictors, method is random forest (“rf”), and the remaining parameter is “trControl”.

```
# train control
trControl <- trainControl(method = "cv", number = 4)

# build model
modelFit.rf <- train(trainData$classe ~ ., method = "rf",
trControl = trControl,
    trainData)

summary(modelFit.rf)
```

```
##                   Length Class      Mode
## call                 4  -none-     call
## type                 1  -none-     character
## predicted         3927  factor     numeric
## err.rate          3000  -none-     numeric
## confusion           30  -none-     numeric
## votes            19635  matrix     numeric
## oob.times         3927  -none-     numeric
## classes              5  -none-     character
## importance          53  -none-     numeric
## importanceSD         0  -none-     NULL
## localImportance      0  -none-     NULL
## proximity            0  -none-     NULL
## ntree                1  -none-     numeric
## mtry                 1  -none-     numeric
## forest              14  -none-     list
## y                 3927  factor     numeric
## test                 0  -none-     NULL
## inbag                0  -none-     NULL
## xNames              53  -none-     character
## problemType          1  -none-     character
## tuneValue            1  data.frame list
## obsLevels            5  -none-     character
```

```
modelFit.rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 1.12%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1115   1   0   0   0   0.0008961
## B    6 746   8   0   0   0.0184211
## C    0   7 678   0   0   0.0102190
## D    0   0   9 635   0   0.0139752
## E    0   1   2  10 709   0.0180055
```

Reshape the test data set i.e discarding the column with where most of the entry are NA and useless predictors.

```
# load the testing data set
testRawData <- read.csv("pml-testing.csv", na.strings =
c("NA", ""))
dim(testRawData)
```

```
## [1]  20 160
```

```
# discard NA
NAs <- apply(testRawData, 2, function(x) {
    sum(is.na(x))
})

validDataT <- testRawData[, which(NAs == 0)]
dim(validDataT)
```

```
## [1] 20 60
```

```
# discard useless predictors
removeIndex <- grep("timestamp|X|user_name|new_window",
names(validDataT))

testData <- validDataT[, -removeIndex]
dim(testData)
```

```
## [1] 20 54
```

Apply the machine learning model to the test data set, and get the predictions.

```
model.predict <- predict(modelFit.rf, testData)
model.predict
```

```
##  [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
```

```
summary(model.predict)
```

```
## A B C D E
## 7 8 1 1 3
```

```
# create a character vector of the predictions and check #the
length of the
# vector
model.predict <- c(as.character(model.predict))
# length of the predicted vector
length(model.predict)
```

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

```
model.predict
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B"
"A" "E" "E" "A"
## [18] "B" "B" "B"
```

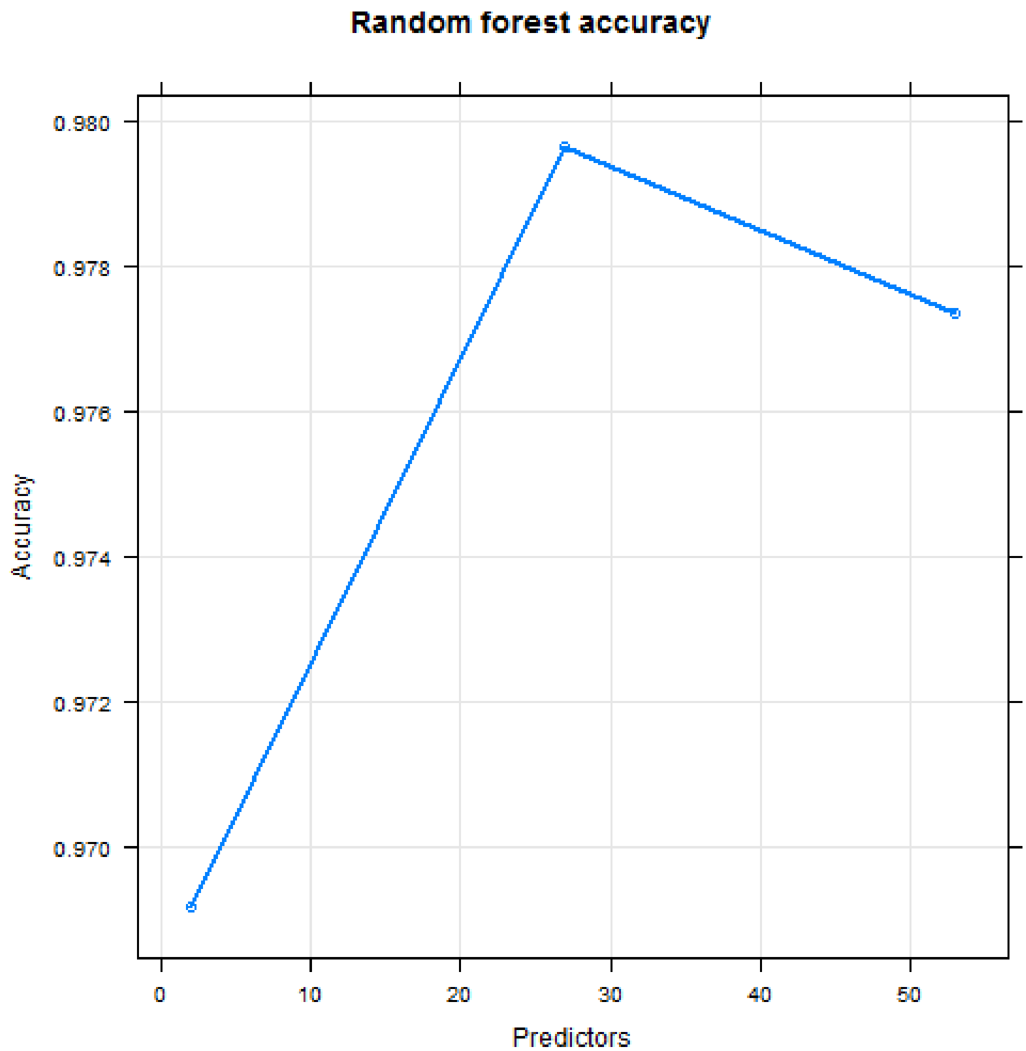
Results

Designed machine learning algorithm:

```
library(caret)
# Result of the random forest model
print(modelFit.rf, digits = 3)
```

```
## Random Forest
##
## 3927 samples
##   53 predictors
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
##
## Summary of sample sizes: 2946, 2945, 2945, 2945
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##   2     0.969     0.961  0.00493      0.00624
##   27    0.98      0.974  0.00221      0.00279
##   53    0.977     0.971  0.00567      0.00716
##
## Accuracy was used to select the optimal model using  the
largest value.
## The final value used for the model was mtry = 27.
```

```
# plot the random forest model
plot(modelFit.rf, log = "y", lwd = 2, main = "Random forest
accuracy", xlab = "Predictors",
    ylab = "Accuracy")
```



Predictions:

```
# Result of the prediction
library(caret)
model.predict
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B"
"A" "E" "E" "A"
## [18] "B" "B" "B"
```

Out of sample error

In sample error rate is 2% (1 - .98 = .2 * 100).

Estimate out of sample error

```
dim(testValidateData)
```

```
## [1] 5885   54
```

```
# predict on testValidateData
predictions <- predict(modelFit.rf, testValidateData)
# length of the predictions
length(predictions)
```

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

```
# true accuracy of the predicted model
outOfSampleError.accuracy <- sum(predictions ==
testValidateData$classe)/length(predictions)

outOfSampleError.accuracy
```

```
## [1] 0.9864
```

```
# out of sample error and percentage of out of sample error
outOfSampleError <- 1 - outOfSampleError.accuracy
outOfSampleError
```

```
## [1] 0.01359
```

```
e <- outOfSampleError * 100
paste0("Out of sample error estimation: ", round(e, digits =
2), "%")
```

```
## [1] "Out of sample error estimation: 1.36%"
```

Write up

Write up the predicted character to the “.txt” files

```
# write up
pml_write_files = function(x) {
    n = length(x)
    for (i in 1:n) {
        filename = paste0("problem_id_", i, ".txt")
        write.table(x[i], file = filename, quote = FALSE,
row.names = FALSE,
            col.names = FALSE)
    }
}

pml_write_files(model.predict)
```