MachineLearningProject

Sypnosis

For this project we use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who quantify how well they do an exercise. 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).

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. Other variables will be use to predict with.

Data Processing

First we install libraries and download data files from URL.

```
#install.packages("caret")
#install.packages("randomForest")
library(caret)
```

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

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
setwd("C:/DataScientistGit/machine-learning")

if(!file.exists("data")){
  dir.create("data")
}

fileurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#download.file(fileurl, destfile = "./data/pml-training.csv")
fileurl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#download.file(fileurl, destfile = "./data/pml-testing.csv")
```

We upload data and prepare them to keep just columns with no missing data. We also remove time columns because this information is no relevant for this study.

```
training <- read.table("./data/pml-training.csv", sep=",", header=TRUE)
testing <- read.table("./data/pml-testing.csv", sep=",", header=TRUE)

training <- training[, c("user_name", "new_window",  "num_window", "roll_belt", "pitch_belt",
"yaw_belt", "total_accel_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z",
"accel_belt_x", "accel_belt_y", "accel_belt_z", "magnet_belt_x", "magnet_belt_y", "magnet_belt_z",
"roll_arm", "pitch_arm", "yaw_arm", "total_accel_arm", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z",
"accel_arm_x", "accel_arm_y", "accel_arm_z", "magnet_arm_x", "magnet_arm_y", "magnet_arm_z",
```

```r
"roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell", "roll_forearm", "pitch_forearm",
"yaw_forearm", "classe")]

dim(training)
```

```
## [1] 19622    36
```

We can see we have a larger sample and therefore we could use crossValidation and create a partition of data like this:
60% for training data
40% for validation data

We will improve accuracy and avoid overfitting in our model

```r
set.seed(33833);

trainingIndex = createDataPartition(training$classe, p = 0.60,list=FALSE)
trainingSet = training[trainingIndex, ]
validationSet = training[-trainingIndex,]
dim(trainingSet)
```

```
## [1] 11776    36
```

```r
dim(validationSet)
```

```
## [1] 7846   36
```

We use randomForest model with training data because it has high accuracy
We can see error of model is not significant.

```r
modelFit <- randomForest(classe~.,data=trainingSet)
print(modelFit)
```

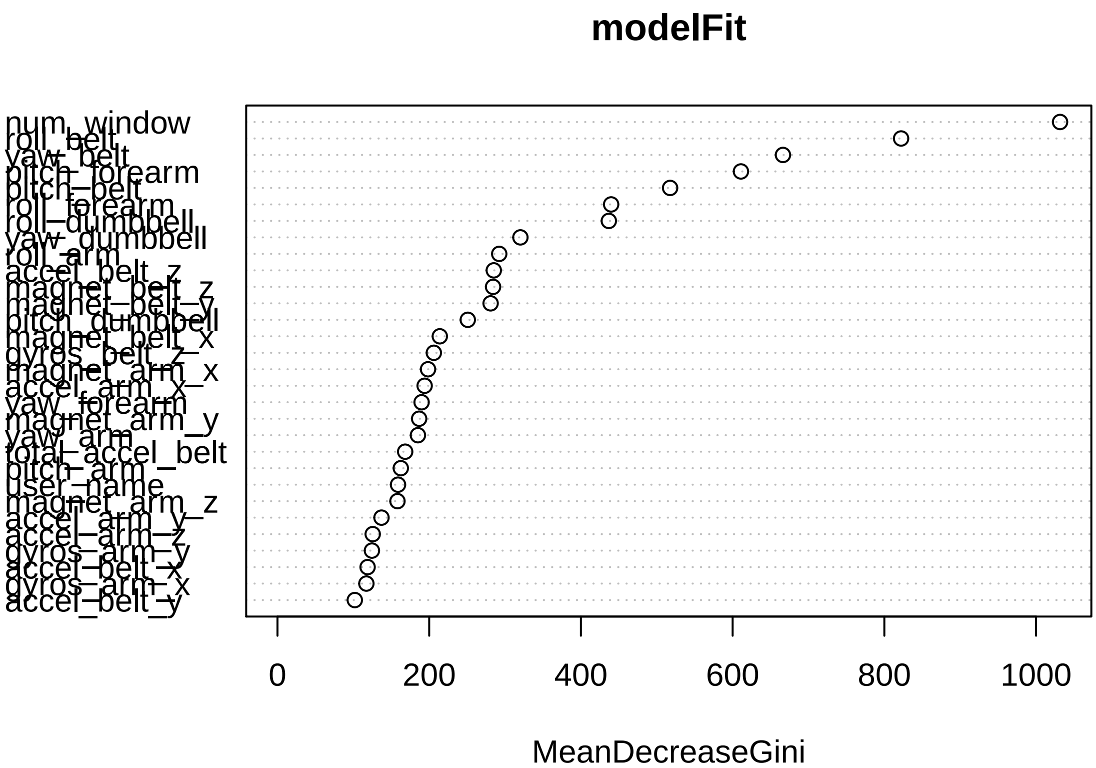
```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainingSet)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.36%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    0    0    2    0 0.0005973716
## B    9 2270    0    0    0 0.0039491005
## C    0   10 2038    6    0 0.0077896787
## D    0    0   11 1918    1 0.0062176166
## E    0    0    0    3 2162 0.0013856813
```

Calculate and order variable importance to see which variables have more influence in data.

```r
vimp<-varImp(modelFit, scale=FALSE)
vimpOrder <- vimp[ order(vimp$Overall, decreasing=TRUE), ,drop=FALSE]
print(vimpOrder)
```

```
##                     Overall
## num_window      1031.631248
## roll_belt        822.060491
## yaw_belt         666.331116
## pitch_forearm    610.950807
## pitch_belt       517.583732
## roll_forearm     439.854013
## roll_dumbbell    436.809591
## yaw_dumbbell     320.216277
## roll_arm         292.314402
## accel_belt_z     285.154136
## magnet_belt_z    284.211766
## magnet_belt_y    280.953359
## pitch_dumbbell   250.877880
## magnet_belt_x    213.987962
## gyros_belt_z     206.119524
## magnet_arm_x     198.457526
## accel_arm_x      193.980948
## yaw_forearm      190.001522
## magnet_arm_y     186.741500
## yaw_arm          185.129217
## total_accel_belt 168.334889
## pitch_arm        162.575545
## user_name        158.909586
## magnet_arm_z     158.185839
## accel_arm_y      137.106866
## accel_arm_z      125.561943
## gyros_arm_y      124.512095
## accel_belt_x     118.795386
## gyros_arm_x      117.174454
## accel_belt_y     101.930023
## total_accel_arm   87.351857
## gyros_belt_x      86.527479
## gyros_belt_y      86.078339
## gyros_arm_z       62.002658
## new_window         0.746865
```

```r
varImpPlot(modelFit, sort=TRUE)
```



Calculate confusion matrix comparing predictions of validation data with their real values. We can see accuracy is very high about 0.9976. It's a good method to predict our data.

```
#Validate data
confusionMatrix(predict(modelFit,newdata=validationSet),validationSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    4    0    0    0
##          B    0 1514    0    0    0
##          C    0    0 1368    9    0
##          D    1    0    0 1276    4
##          E    0    0    0    1 1438
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.9962, 0.9985)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9969
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9974   1.0000   0.9922   0.9972
## Specificity            0.9993   1.0000   0.9986   0.9992   0.9998
## Pos Pred Value         0.9982   1.0000   0.9935   0.9961   0.9993
## Neg Pred Value         0.9998   0.9994   1.0000   0.9985   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1930   0.1744   0.1626   0.1833
## Detection Prevalence   0.2849   0.1930   0.1755   0.1633   0.1834
## Balanced Accuracy      0.9994   0.9987   0.9993   0.9957   0.9985
```

Predictions Results

We prepare testing data and predict them with model.

```r
testing <- testing[, c("user_name", "new_window",  "num_window", "roll_belt", "pitch_belt",
"yaw_belt", "total_accel_belt", "gyros_belt_x", "gyros_belt_y", "gyros_belt_z",
"accel_belt_x", "accel_belt_y", "accel_belt_z", "magnet_belt_x", "magnet_belt_y", "magnet_belt_z",
"roll_arm", "pitch_arm", "yaw_arm", "total_accel_arm", "gyros_arm_x", "gyros_arm_y", "gyros_arm_z",
"accel_arm_x", "accel_arm_y", "accel_arm_z", "magnet_arm_x", "magnet_arm_y", "magnet_arm_z",
"roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell", "roll_forearm", "pitch_forearm",
"yaw_forearm")]
classe<- factor(x="A",levels=c("A", "B", "C", "D", "E"))
testing <- cbind(testing, classe)
testing <- rbind(validationSet[1,],testing)

# Predict data
pred <- predict(modelFit,newdata=testing[-1,])
print(pred)
```

```
## 22  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

Annex I:

Program help you to submit predictions in assigment.
Generating Answers Files to Submit Assignment

answers = c("B","A","B","A","A","E","D","B","A","A","B","C","B","A","E","E","A","B","B","B")

then you can load this function by copying and pasting it into R:
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)
}
}
then create a folder where you want the files to be written. Set that to be your working directory and run:
pml_write_files(answers)