PML_Project

Amirrudin Bin Dahlan

Saturday, January 24, 2015

Executive Summary

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. 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, the goal will be to use data from accelerometers on the belt, forearm, arm, dumbell of the participants and predict the manner in which they did the exercise.

Load required packages

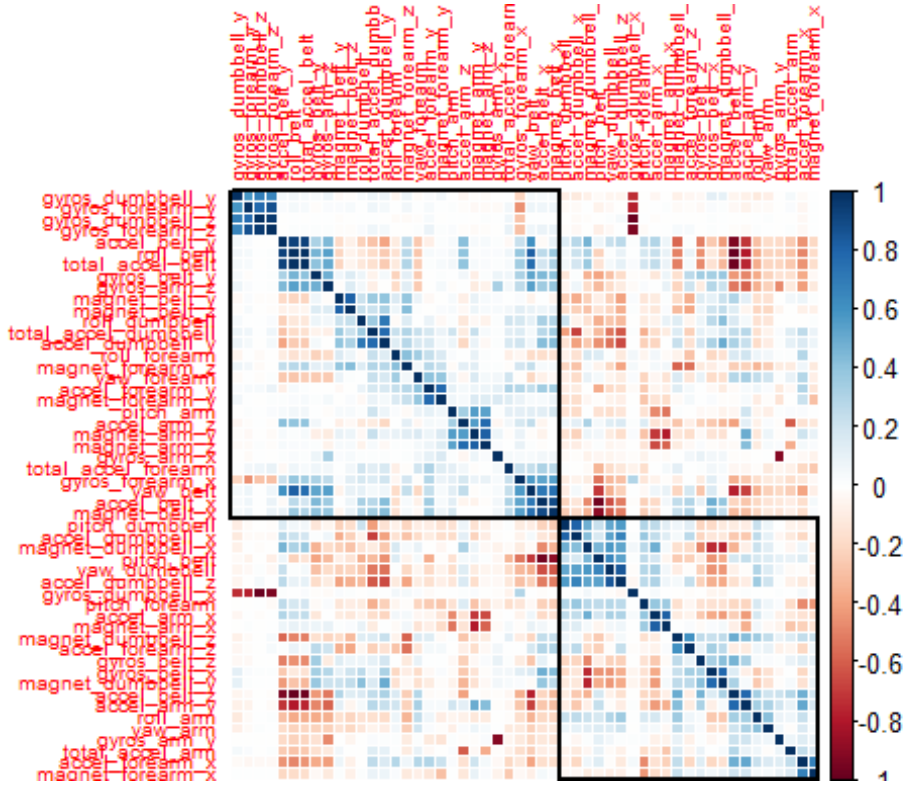
```
library(kernlab)
library(corrplot)
library(randomForest)
library(caret)
```

Get and clean datasets from invalid values

```
# download datasets
download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv", destfile = "pml-training.csv")
download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv", destfile = "pml-testing.csv")

# read data to local variable
dt_train <- read.csv("pml-training.csv", na.strings=c("NA",""))

# get desired columns and remove unwanted fields
dt_train <- dt_train[8:length(dt_train)]
dt_train<-dt_train[,colSums(is.na(dt_train)) == 0]
```

Set data partitions into training and validation

```
# split data set
split_dt_train <- createDataPartition(y = dt_train$classe, p = 0.7, list =
FALSE)
training <- dt_train[split_dt_train,]
crossval <- dt_train[-split_dt_train,]

# correlation matrix shows the degree of correlation between different
variables with those in the highlighted boxes as the more prevalent ones.
However this does not really indicate any assurance of model's accuracy as
yet.
correlMatrix <- cor(training[, -length(training)])
corrplot(correlMatrix, order = "hclust", addrect = 2, method = "color",
tl.cex = 0.6)
```



Random forest function - Model Evaluation

```
# in random forest function, error rates is calculated within function.
# create training model with training data set.
model <- randomForest(classe ~ ., data = training)

# model shows high correlation between the factors as we can see below
based on the diagonal five 4-digit figures (etc 3904,2648 and so on)
spanning from the top left to bottom right part of the table matrix. There
are indications of outliers, but the percentage figures are small as shown
in column 'class.error'
print(model)

##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.47%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B   13 2638    7    0    0 0.0075244545
## C    0    8 2386    2    0 0.0041736227
## D    0    0   24 2227    1 0.0111012433
## E    0    0    3    5 2517 0.0031683168
```

Confusion matrix - Model Evaluation

```
# overall model accuracy shows a high value of 0.99 with small amounts of
outliers in the table matrix.
confusionMatrix(crossval$classe, predict(model, crossval))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    2 1135    2    0    0
##          C    0    5 1021    0    0
##          D    0    0   13  950    1
##          E    0    0    1    6 1075
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9947   0.9846   0.9937   0.9991
## Specificity            0.9998   0.9992   0.9990   0.9972   0.9985
## Pos Pred Value         0.9994   0.9965   0.9951   0.9855   0.9935
## Neg Pred Value         0.9995   0.9987   0.9967   0.9988   0.9998
## Prevalence             0.2846   0.1939   0.1762   0.1624   0.1828
## Detection Rate         0.2843   0.1929   0.1735   0.1614   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9993   0.9969   0.9918   0.9954   0.9988
```

Apply test dataset to model

```
# read data to local variable
dt_test <- read.csv("pml-testing.csv", na.strings=c("NA",""))

# get desired columns and remove unwanted fields
dt_test <- dt_test[8:length(dt_test)]
dt_test<-dt_test[,colSums(is.na(dt_test)) == 0]

# predict the classes of the test set
outcome <- predict(model, dt_test)

# these were codes provided by MOOC
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
)
  }
}

# write output
pml_write_files(outcome)
```