Prediction of Weight Lifting Exercises

Aditya

October 10, 2017

Synopsis

This Report captures the Analysis to Predict the Activity Recognition of Weight Lifting Exercises.

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. The quality of executing an activity, the "how (well)", has only received little attention so far, even though it potentially provides useful information for a large variety of applications.

The Data in this project is collected from Sensors mounted on user's glove, armband, lumbar belt and dumbbell when participants performed a set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

- (Class A) Exactly according to the specification.
- (Class B) Throwing the elbows to the front.
- (Class c) Lifting the dumbbell only halfway.
- (Class D) Lowering the dumbbell only halfway.
- (Class E) Throwing the hips to the front.

The goal of the project is to predict the manner in which they did the exercise

Report Section

```
library("markdown")
library("rmarkdown")
library("knitr")
library("ggplot2")
library("caret")
library("corrplot")
library("doParallel")
```

Setting Work Directory and downloading the files from Source

```
setwd("G:/Data Science Project/Practical Machine Learning/Wk4/Project")

#download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfi
#download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfil
```

Reading the files into R by interpreting the strings ('#DIV/0!','','NA') as NA values

```
training <- read.csv(file = "training.csv",header = TRUE,sep = "," ,
                     na.strings = c('#DIV/0!','','NA'), stringsAsFactors = FALSE)
testing  <- read.csv(file = "testing.csv", header = TRUE,sep = "," ,
                     na.strings = c('#DIV/0!','','NA'), stringsAsFactors = FALSE)
```

To check the structure of the table

```
dim(training)
```

[1] 19622 160

```
#kable(str(training))
sum(is.na.data.frame(training))
```

[1] 1925102

Fill Ratio of the variables in the Training and Testing Tables

```
fillratio_train <- data.frame(ratio= round(colSums(!is.na(training))/nrow(training), digits=2))
table(fillratio_train)
```

fillratio_train 0 0.02 1 6 94 60

```
fillratio_test <- data.frame(ratio= round(colSums(!is.na(testing))/nrow(testing), digits=2))
table(fillratio_test)
```

fillratio_test 0 1 100 60

```
### So 60 variables in both training and testing dataset are having 100% fill ratio and nearly 100 vari
```

checking if the same variables are missing in both datasets

```
# outputs Testing set variable name which mismatches with training set variable list
names(testing)[names(training)!=names(testing)]
```

[1] "problem_id"

```
# outputs Training set variable name which mismatches with testing set variable list
names(training)[names(training)!=names(testing)]
```

[1] "classe"

```
which(colnames(training)=="classe") # COlumn Index of the mismatched variable
```

[1] 160

```
kable(confusionMatrix(round(fillratio_train[-160,],0),fillratio_test[-160,])$table)
```

	0	1
0	100	0

	0	1
1	0	59

```
# The Confusion Matrix confirms 100 variables in both testing and training datasets are missing

    training1 <- subset(training, select = (fillratio_train==1))
    testing1  <- subset(testing, select = (fillratio_test==1))
```

Checking the distribution of classe variable

```
    table(training1$classe)
```

A B C D E 5580 3797 3422 3216 3607

```
    table(training1$user_name,training1$classe)
```

```
          A    B    C    D    E
```

adelmo 1165 776 750 515 686 carlitos 834 690 493 486 609 charles 899 745 539 642 711 eurico 865 592 489 582 542 jeremy 1177 489 652 522 562 pedro 640 505 499 469 497

```
# Identification of Near Zero Variance Predictors
    nearZeroVar(training1, names = TRUE)
```

[1] "new_window"

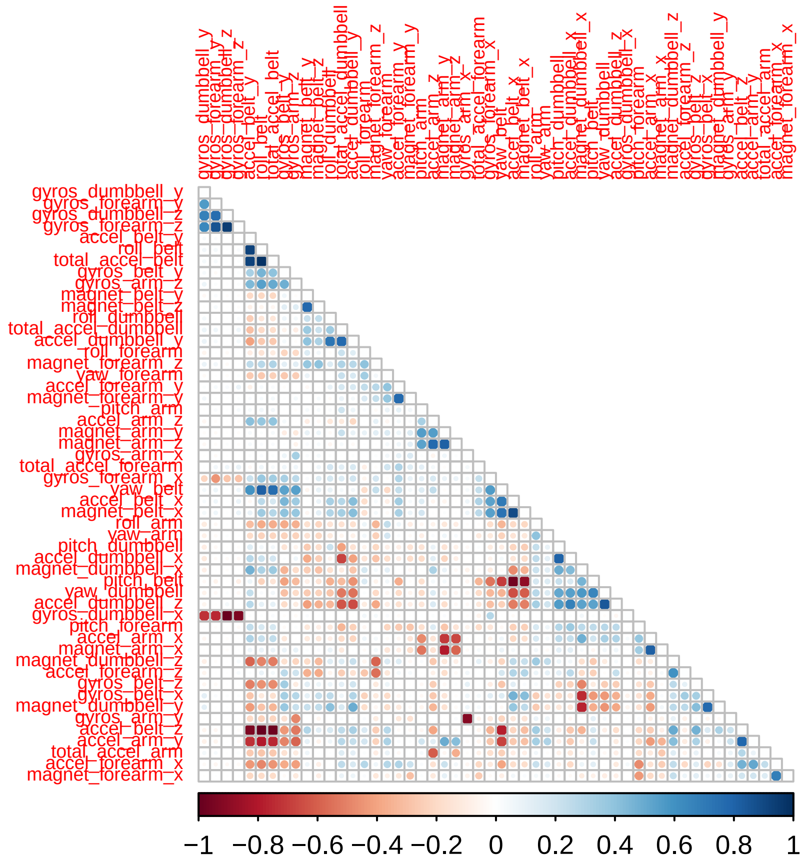
```
# Removing the user_name, Timestamp and window variables
    var_drop <- grep(pattern="^X$|user|timestamp|window", names(training1))
    training2 <- subset.data.frame(training1, select=-c(var_drop))

# Slicing the training2 dataset into train and test datasets
    inTrain <- createDataPartition(training2$classe,p = .75,list = FALSE)
    train_data <- training2[inTrain,]
    test_data  <- training2[-inTrain,]
```

Generating Correlation Matrix excluding the Classe categorical variable

```
fa_cor <- cor(x = train_data[,-53])
    diag(fa_cor) <- 0  # Setting the Diagonal values to 0 as its correlation of same variables

# Correlation plot of the correlation Matrix
    corrplot(fa_cor, tl.pos = "lt", order="hclust", hclust.method="complete",type = "lower",tl.cex = .6)
```




```r
# To generate the variables which are 90% highly correlated with eachother
    kable(which(fa_cor>0.9, arr.ind = TRUE))
```

	row	col
total_accel_belt	4	1
accel_belt_y	9	1
roll_belt	1	4
accel_belt_y	9	4
roll_belt	1	9
total_accel_belt	4	9
gyros_forearm_z	46	33
gyros_dumbbell_z	33	46

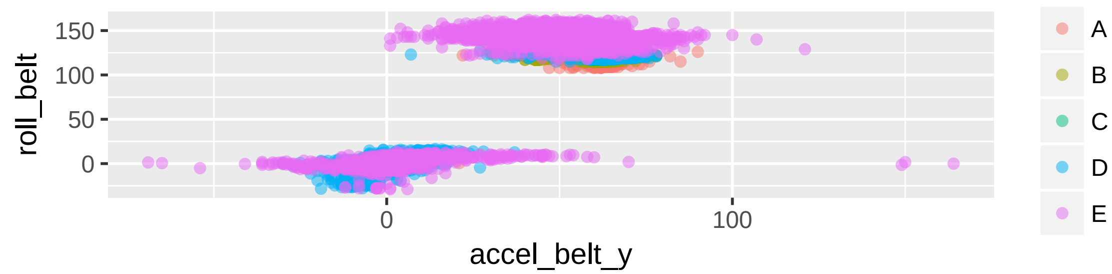
```r
    names(train_data)[c(1,4,9,33,46)]
```

[1] "roll_belt" "total_accel_belt" "accel_belt_y"
[4] "gyros_dumbbell_z" "gyros_forearm_z"

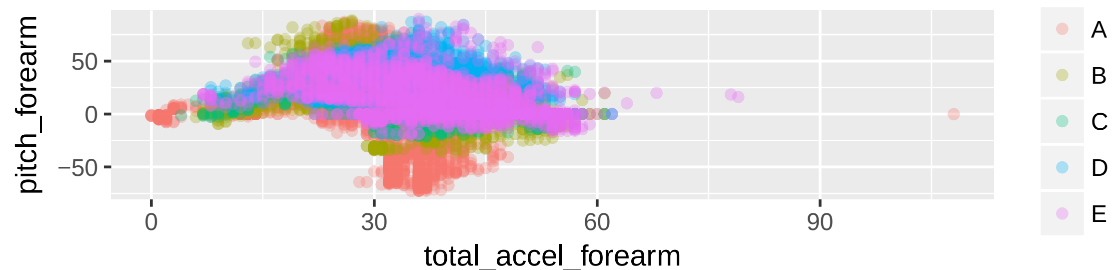
Below plots are generated as part of EDA process.

Few of the plots are selected out of various built where there's a clarity of distinction in classe variables.

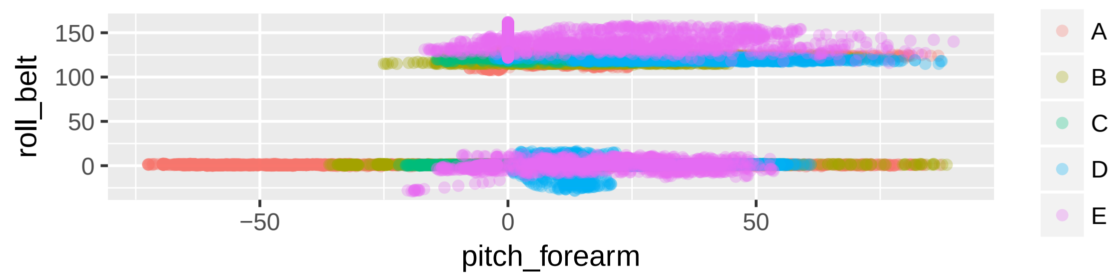
```
ggplot(data = train_data, mapping = aes(x=accel_belt_y , y=roll_belt, col=classe))+geom_point(alpha
```




```
ggplot(data = train_data, mapping = aes(x=total_accel_forearm , y=pitch_forearm, col=classe))+geom_
```




```
ggplot(data = train_data, mapping = aes(x=pitch_forearm , y=roll_belt, col=classe))+geom_point(alpha
```



Configuring Parallel Processing

```
    cluster <- makeCluster(detectCores() - 1)
    registerDoParallel(cluster)

## Configuring Train COntrol Object and Developing Train Model
    fitControl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
    system.time(rf_model <- train(classe ~ ., method="rf", data = train_data, trControl=fitControl, ntre
```

user system elapsed 33.92 0.28 281.67

```
## De-registering parallel processing cluster
    stopCluster(cluster)
```

```r
registerDoSEQ()
```

Generating Confusion matrix for Train model built

```r
kable(round(confusionMatrix.train(rf_model)$table,1))
```

	A	B	C	D	E
A	28.4	0.1	0.0	0.0	0.0
B	0.0	19.2	0.1	0.0	0.0
C	0.0	0.1	17.3	0.2	0.0
D	0.0	0.0	0.0	16.1	0.1
E	0.0	0.0	0.0	0.0	18.3

```r
## Printing the Final MOdel built
    rf_model$finalModel
```

Call: randomForest(x = x, y = y, ntree = 100, mtry = param$mtry) Type of random forest: classification Number of trees: 100 No. of variables tried at each split: 27

```
OOB estimate of  error rate: 0.65%
```

Confusion matrix: A B C D E class.error A 4178 4 2 0 1 0.001672640 B 18 2822 8 0 0 0.009129213 C 0 12 2545 10 0 0.008570316 D 0 1 26 2384 1 0.011608624 E 0 1 4 8 2693 0.004804139

Predicting the results for Test_data and generating the confusion matrix

```r
test_results <- predict(object =rf_model,newdata = test_data  )
confusionMatrix(test_results, test_data$classe)
```

Confusion Matrix and Statistics

```
Reference
```

Prediction A B C D E A 1394 10 0 0 0 B 0 937 5 0 1 C 1 2 849 5 2 D 0 0 1 799 3 E 0 0 0 0 895

Overall Statistics

```
            Accuracy : 0.9939
              95% CI : (0.9913, 0.9959)
No Information Rate : 0.2845
P-Value [Acc > NIR] : < 2.2e-16

               Kappa : 0.9923
```

Mcnemar's Test P-Value : NA

Statistics by Class:

```
                 Class: A Class: B Class: C Class: D Class: E
```

Sensitivity 0.9993 0.9874 0.9930 0.9938 0.9933 Specificity 0.9972 0.9985 0.9975 0.9990 1.0000 Pos Pred Value 0.9929 0.9936 0.9884 0.9950 1.0000 Neg Pred Value 0.9997 0.9970 0.9985 0.9988 0.9985 Prevalence 0.2845 0.1935 0.1743 0.1639 0.1837 Detection Rate 0.2843 0.1911 0.1731 0.1629 0.1825 Detection Prevalence 0.2863 0.1923 0.1752 0.1637 0.1825 Balanced Accuracy 0.9982 0.9929 0.9953 0.9964 0.9967

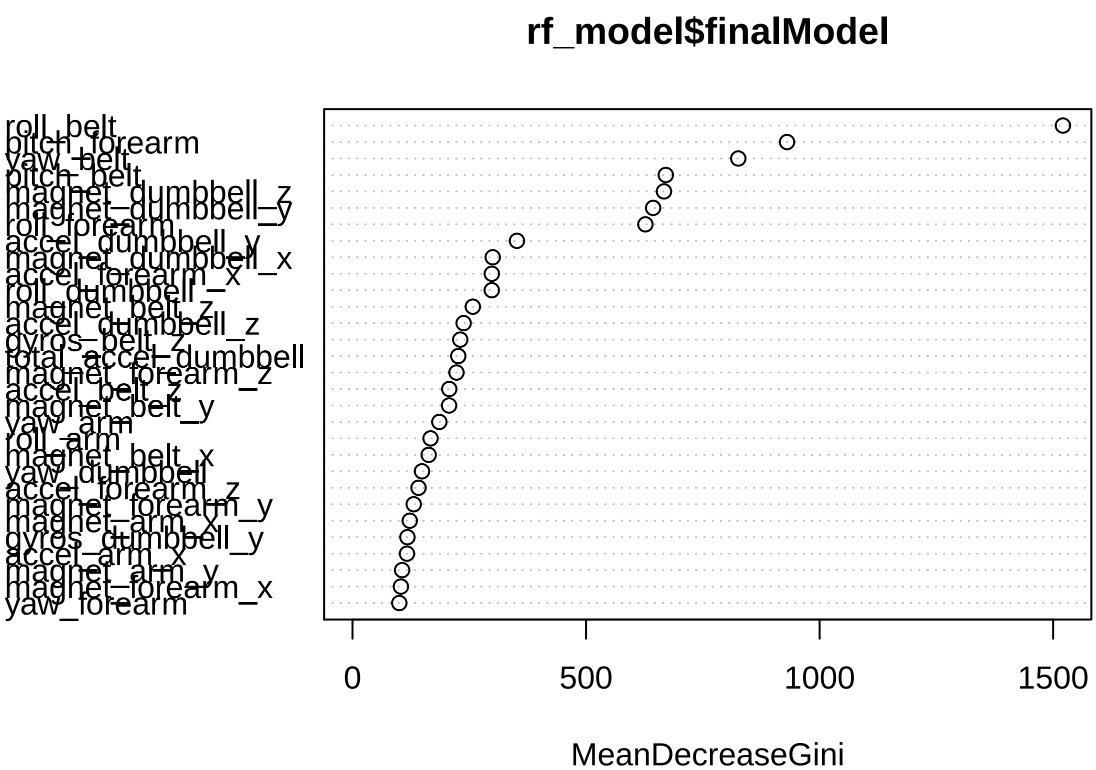
```
## Error rate
    #### Train dataset:
    sum(rf_model$finalModel$predicted!=train_data$classe)/length(train_data$classe)
```

[1] 0.006522625

```
    #### Test dataset:
    sum(test_results!=test_data$classe)/length(test_data$classe)
```

[1] 0.006117455

```
## Generating the Variable Importance plot for the final model
    varImpPlot(rf_model$finalModel)
```



The Test Data set is predicted at a 99.6% Accuracy and Out of Bound Error rate is 0.6% ## The roll_belt, pitch_forearm and Yaw_belt are the Top 3 Important Predictors of the Random Forest Model Built

Predicting the results for Final Test Dataset

```
    final_test_results <- predict(object=rf_model, testing)
    final_test_results
```

[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