PREDICTION ASSIGNMENT WRITEUP

OVERVIEW

The goal of this report is to predict the manner in which 6 participants performed some exercises described in the Background section. This is the “classe” variable in the training set. I used the other variables to achieve the best prediction model according to the training test. After that, I applied the best prediction model to the test set.

BACKGROUND

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. 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 (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

DATA LOADING AND EXPLORATORY ANALYSIS

First, we load the libraries and the training Dataset from the link. Aftewards, we clean the input data by eliminanting variables with a high amount of NA or missing data.

```
#LOAD LIBRARIES
library(ISLR)
library(caret)
library(e1071)
library(RANN)
library(corrplot)

#Download and clean the datasets
path_training<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training_data<-read.csv(path_training,na.strings=c("NA","","#DIV/0!"))

#Delete columns wiht 90% values are missing values
filter_variables<-sapply(training_data,function(x) mean(is.na(x)))>0.90
training_data_2<-training_data[,filter_variables==FALSE]
```

Then, we removed variables with near zero variance predictors :

```
# Check zero variance predictors and remove them
nsv<-nearZeroVar(training_data_2,saveMetrics=TRUE)
training_filter<-training_data_2[,nsv$nzv==FALSE]
```

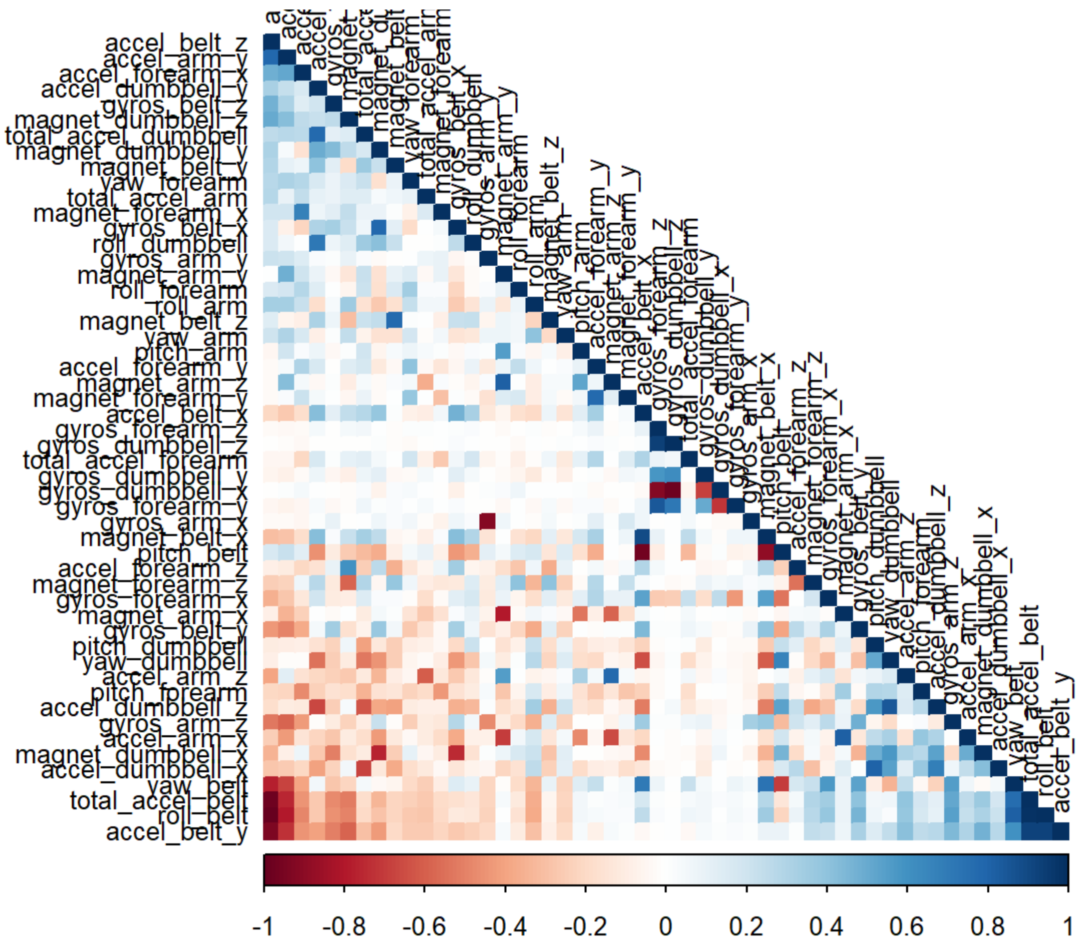
Finally, I filter out the identification variables

```
# Remove Identification variables
training_filter_2<-training_filter[,-(1:6)]
```

After cleaning the number of variables for the analysis has been reduced from 160 to 53.

CORRELATION ANALYSIS

```
CORRELATION<-cor(training_filter_2[,-53])
corrplot(CORRELATION,order="FPC",method="color",type="lower",tl.cex=0.8,tl.col=rgb(0,0,0))
```



```
corr<-abs(cor(training_filter_2[-53]))
diag(corr)<-0
which(corr>0.9,arr.ind=TRUE)
```

```
##                   row col
## total_accel_belt   4   1
## accel_belt_y       9   1
## accel_belt_z      10   1
## accel_belt_x       8   2
## roll_belt          1   4
## accel_belt_y       9   4
## accel_belt_z      10   4
## pitch_belt         2   8
## roll_belt          1   9
## total_accel_belt   4   9
## accel_belt_z      10   9
## roll_belt          1  10
## total_accel_belt   4  10
## accel_belt_y       9  10
## gyros_arm_y       19  18
## gyros_arm_x       18  19
## gyros_dumbbell_z  33  31
## gyros_forearm_z   46  31
## gyros_dumbbell_x  31  33
## gyros_forearm_z   46  33
## gyros_dumbbell_x  31  46
## gyros_dumbbell_z  33  46
```

As can be seen, There are few variables correlated (>0.9 out of 1). I will filter out the correlated variables :

- roll_belt correlated with total_accel_belt,accel_belt_y and accel_belt_z (1 -> 9,4,10)
- pitch_belt correlated with accel_belt_x (2 -> 8)
- gyros_arm_x correlated with gyros_arm_y (18 -> 19)
- gyros_dumbbell_x correlated with gyros_dumbbell_z and gyroes_forearm_z (31 -> 33 and 46)

```
training_filter_3<-training_filter_2[,-c(4,9,10,8,19,33,46)]
```

Finally, The number of variables is 46.

PREDICTION MODEL BUILDING

First, I split the training Dataset into training set and testing set, 70%-30%

```
# CREATE A PARTITION TRAINING DATASET 70% and TEST DATASET 30%
inTrain<-createDataPartition(training_filter_3$classe,p=0.7,list=FALSE)
training<-training_filter_3[inTrain,]
testing<-training_filter_3[-inTrain,]
```

Then, I calculate three prediction models : BOOSTING, LINEAR DISCRIMINANT ANALYSIS AND RANDOM FOREST.

mod_gbm<-train(classe ~ .,data=training,method="gbm") mod_lda<-train(classe ~ .,data=training,method="lda") mod_rf<-train(classe ~ .,data=training,method="rf")

FINALLY, I WILL COMPARE THE DIFFERENT MODELS WITH THE TESTING SET

```
# BOOSTING
pred_gbm<-predict(mod_gbm,newdata=testing)
confusionMatrix(pred_gbm,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1650   21    0    0    0
##          B   15 1093   32    4    9
##          C    5   20  984   32    7
##          D    3    3    9  924   13
##          E    1    2    1    4 1053
##
## Overall Statistics
##
##                Accuracy : 0.9692
##                  95% CI : (0.9645, 0.9735)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9611
##
##  Mcnemar's Test P-Value : 2.047e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9596   0.9591   0.9585   0.9732
## Specificity            0.9950   0.9874   0.9868   0.9943   0.9983
## Pos Pred Value         0.9874   0.9480   0.9389   0.9706   0.9925
## Neg Pred Value         0.9943   0.9903   0.9913   0.9919   0.9940
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2804   0.1857   0.1672   0.1570   0.1789
## Detection Prevalence   0.2839   0.1959   0.1781   0.1618   0.1803
## Balanced Accuracy      0.9903   0.9735   0.9729   0.9764   0.9858
```

```
# LINEAR DISCRIMINANT ANALYSIS
pred_lda<-predict(mod_lda,newdata=testing)
confusionMatrix(pred_lda,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1372  199   97   80   45
##          B   35  689   93   51  171
##          C  132  123  677  105  115
##          D  124   44  114  682   95
##          E   11   84   45   46  656
## 
## Overall Statistics
## 
##                Accuracy : 0.6926
##                  95% CI : (0.6806, 0.7044)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6105
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8196   0.6049   0.6598   0.7075   0.6063
## Specificity            0.9000   0.9263   0.9022   0.9234   0.9613
## Pos Pred Value         0.7652   0.6631   0.5877   0.6440   0.7791
## Neg Pred Value         0.9262   0.9071   0.9263   0.9416   0.9155
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2331   0.1171   0.1150   0.1159   0.1115
## Detection Prevalence   0.3047   0.1766   0.1958   0.1799   0.1431
## Balanced Accuracy      0.8598   0.7656   0.7810   0.8154   0.7838
```

```
# RANDOM FOREST
pred_rf<-predict(mod_rf,newdata=testing)
confusionMatrix(pred_rf,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1138    2    0    0
##          C    0    1 1024    3    0
##          D    0    0    0  960    0
##          E    0    0    0    1 1082
## 
## Overall Statistics
## 
##                Accuracy : 0.9988
##                  95% CI : (0.9976, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9985
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9981   0.9959   1.0000
## Specificity            1.0000   0.9996   0.9992   1.0000   0.9998
## Pos Pred Value         1.0000   0.9982   0.9961   1.0000   0.9991
## Neg Pred Value         1.0000   0.9998   0.9996   0.9992   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1740   0.1631   0.1839
## Detection Prevalence   0.2845   0.1937   0.1747   0.1631   0.1840
## Balanced Accuracy      1.0000   0.9994   0.9986   0.9979   0.9999
```

The best prediction model is Random Forest with a accuracy of 99.44%
LDA Accuracy is about 68%
GBM Accuracy is about 96%

APPLYING THE BEST MODEL TO THE TEST DATASET

Giving the results from the PREDICTION MODEL BUILDING section. We are going to use Random Forest Model because it fits in quite well on the test dataset.

First, I have to load the data

```
#Download and clean the datasets
path_test="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
test_data<-read.csv(path_test)
```

Afterwards, I will clean the data as I did with the training dataset

```
#CLEAN AND PREPROCESS TEST DATA
test_data_2<-test_data[,filter_variables==FALSE]
test_filter<-test_data_2[,nsv$nzv==FALSE]
test_filter_2<-test_filter[,-(1:6)]
testData<-test_filter_2[,-c(4,9,10,8,19,33,46)]
```

Finally, I apply the best prediction model on the test Dataset.

```
#APPLY THE PREDICTION MODEL RANDOM FOREST
pred_final<-predict(mod_rf,newdata=testData)
pred_final
```

```
##  [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
```

The results will be applied in the Quiz of the coursera Course : Practical Machine Learning.