

# HAL\_Analysis

Abhishek Ajmera

07/07/2020

## Summary:

In this project I have performed analysis on Human activity recognition using the WLE dataset: <http://groupware.les.inf.puc-rio.br/har#dataset>. I first loaded the dataset into R. Then, I performed basic EDA on the dataset discovering the large number of missing values. I performed some cleansing and imputing operations. I then split the dataset into training and test sets. I trained 3 models - Random Forest (rf), Neural network and Gradient boosting machine (gbm) on the training set. After evaluating their prediction accuracy on the test set, I selected the gbm model as my final model.

## 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 dumbbell 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>

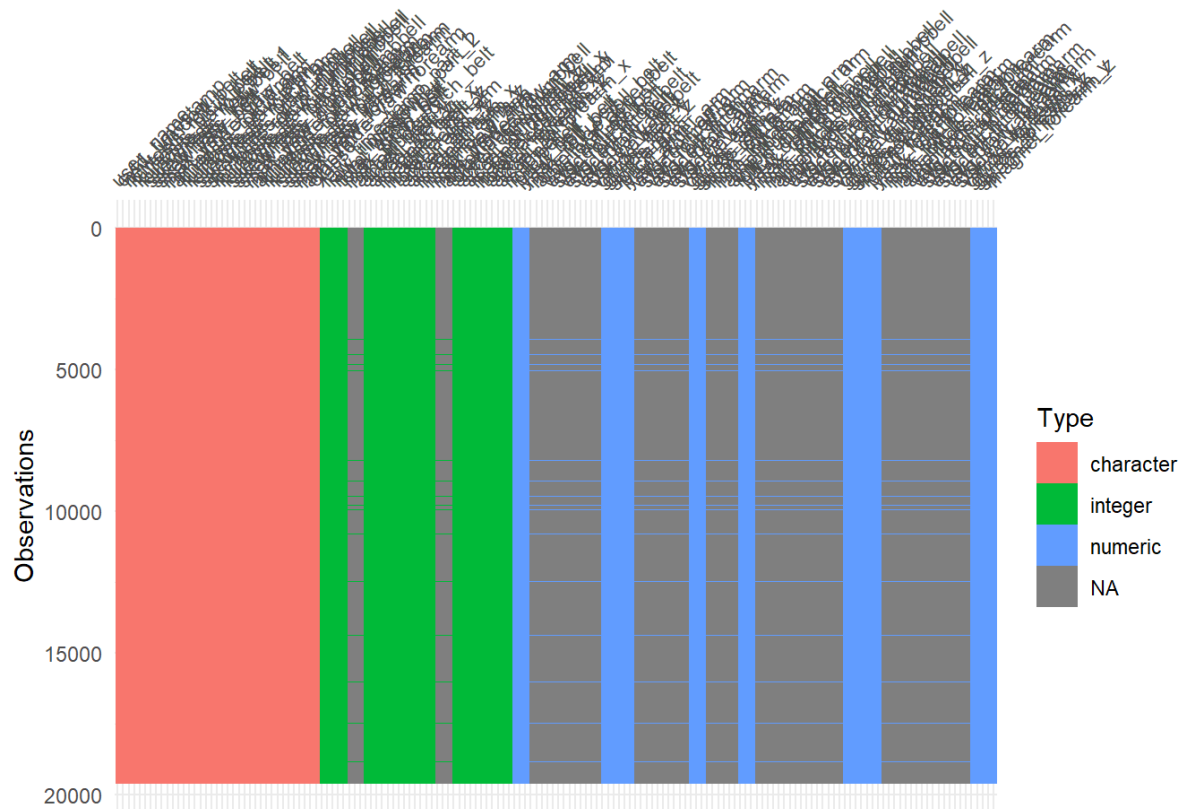
Goal: The goal of the project is to predict the manner in which they did the exercise - classe variable

## Loading the dataset

```
pml_training <- read.csv("C:/Users/Abhis/Downloads/pml-training.csv")
pml_testing <- read.csv("C:/Users/Abhis/Downloads/pml-testing.csv")
dim(pml_training)
## [1] 19622 160
```

## Looking at the distribution of missing data in the dataset:

```
library(visdat)
## Warning: package 'visdat' was built under R version 4.0.2
vis_dat(pml_training, warn_large_data = FALSE)
```



From the above plot, we can see that for some columns, most of their data is NA. Interestingly, in these columns, actual data entries (Non-NA values) are for the entire row ie. for these columns having non-NA vlaue, that entire row has Non-NA values (complete rows).

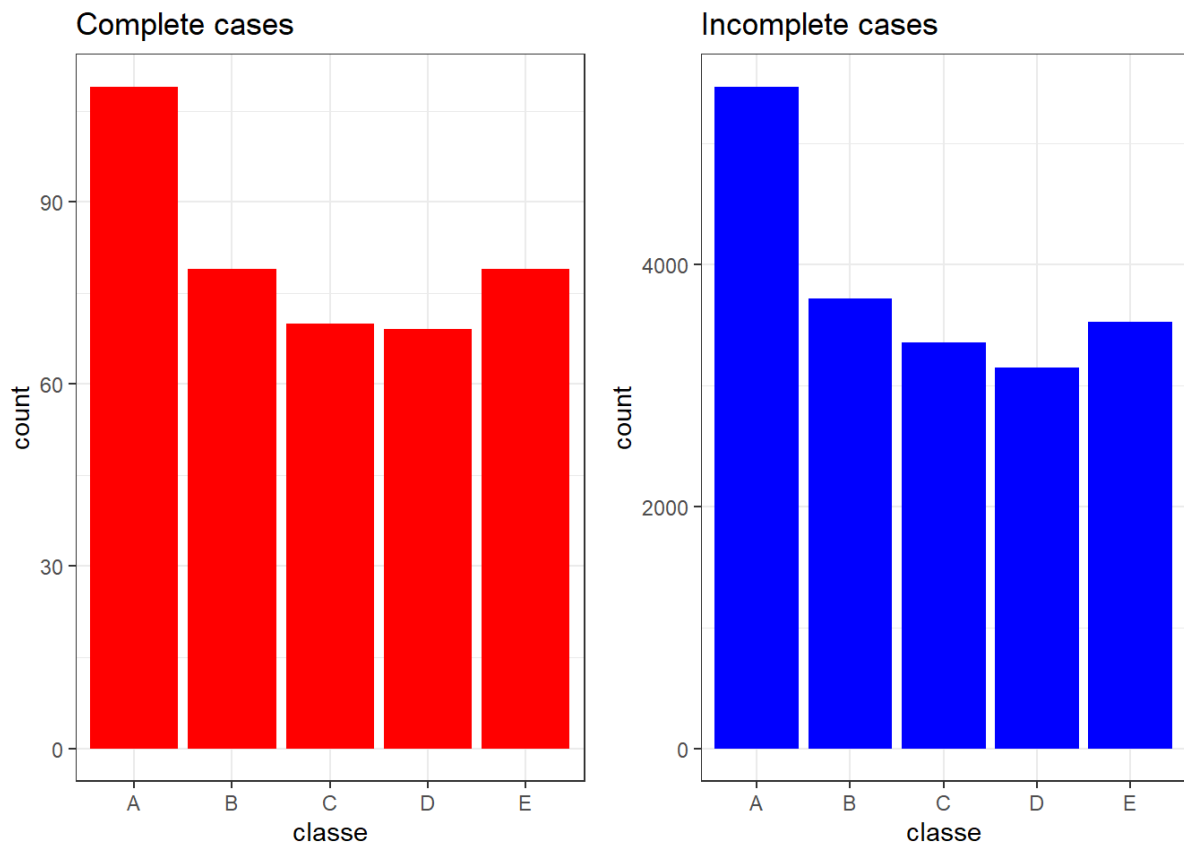
Now checking if there is a difference in the distribution of 'classe' comparing complete and incomplete cases

```
library(ggplot2)
library(gridExtra)

## Warning: package 'gridExtra' was built under R version 4.0.2

a <- pml_training[complete.cases(pml_training),]
b <- pml_training[!complete.cases(pml_training),]

x=ggplot(aes(x=classe), data=a)+geom_bar(fill="red")+theme_bw()+ggtitle("Complete cases")
y=ggplot(aes(x=classe), data=b)+geom_bar(fill="blue")+theme_bw()+ggtitle("Incomplete cases")
grid.arrange(x, y, nrow = 1)
```



From the above plot, the distribution of classe is similar.

Looking at the nature of the dataset and the vast number of NAs I have decided that it would be prudent to replace these values with 0 as there are far fewer complete rows to make a prediction on the incomplete ones

```
pml_training<-pml_training[,-c(1:5)]
pml_training[is.na(pml_training)] <- 0
library(caret)
## Warning: package 'caret' was built under R version 4.0.2
## Loading required package: lattice
head(nearZeroVar(pml_training,saveMetrics = TRUE))
```

##	freqRatio	percentUnique	zeroVar	nzv
## new_window	47.330049	0.01019264	FALSE	TRUE
## num_window	1.000000	4.37264295	FALSE	FALSE
## roll_belt	1.101904	6.77810621	FALSE	FALSE
## pitch_belt	1.036082	9.37722964	FALSE	FALSE
## yaw_belt	1.058480	9.97349913	FALSE	FALSE
## total_accel_belt	1.063160	0.14779329	FALSE	FALSE

## Removing variables having no variability and checking number of columns in new dataframe

```
Now<-pml_training[,-nearZeroVar(pml_training)]
a<-data.frame(cbind(length(Now),length(pml_training)))
colnames(a)=c("New", "Old")
a
##      New Old
## 1    54 155
```

100 columns were omitted reducing the dataset by a big margin

## Creating data Partition - 60% Training 40 % Testing

```
set.seed(123)
inTraining<-createDataPartition(Now$classe,p=0.6,list=FALSE)
Training=Now[inTraining,]
Testing=Now[-inTraining,]
```

## Training models:

### Random Forest Model (rf)

```
set.seed(123)
rftrain=train(classe~.,data= Training,method="rf",,verbose=FALSE)
```

### Gradient boosting Machine (gbm)

```
set.seed(123)
gbm_train<-train(classe~.,Training,verbose=FALSE,method="gbm")
```

## Neural Network

```
set.seed(123)
nnetTrain<- train(classe~., data = Training, method = "nnet",verbose=FALSE)
```

## Predictions on test set

```
set.seed(123)
Testing$classe=as.factor(Testing$classe)
rf_pred=predict(rftrain,Testing)
gbmpred=predict(gbm_train,Testing)
```

```
nnpred=predict(nnetTrain,Testing)
```

## Evaluating prediction accuracy

```
confusionMatrix(nnpred,Testing$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction      A      B      C      D      E
```

```
##           A 1147    45    23     8    42
```

```
##           B   83   371    83   128   228
```

```
##           C  411   288   609   236   237
```

```
##           D    0     0     0     0     0
```

```
##           E  591   814   653   914   935
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.3903
```

```
##           95% CI : (0.3794, 0.4012)
```

```
## No Information Rate : 0.2845
```

```
## P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##           Kappa : 0.2388
```

```
##
```

```
## McNemar's Test P-Value : < 2.2e-16
```

```
##
```

```
## Statistics by Class:
```

```
##
```

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

```
## Sensitivity          0.5139 0.24440 0.44518 0.0000 0.6484
```

```
## Specificity          0.9790 0.91751 0.81908 1.0000 0.5359
```

```
## Pos Pred Value       0.9067 0.41545 0.34194      NaN 0.2393
```

```
## Neg Pred Value       0.8351 0.83504 0.87486 0.8361 0.8713
```

```
## Prevalence           0.2845 0.19347 0.17436 0.1639 0.1838
```

```
## Detection Rate       0.1462 0.04729 0.07762 0.0000 0.1192
```

```
## Detection Prevalence 0.1612 0.11382 0.22699 0.0000 0.4980
```

```
## Balanced Accuracy     0.7464 0.58096 0.63213 0.5000 0.5922
```

```
confusionMatrix(rf_pred,Testing$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    A    B    C    D    E
```

```
##           A 2231    3    0    0    0
```

```
##           B    0 1513    4    0    0
```

```
##           C    0    2 1364    0    0
```

```
##           D    0    0    0 1285    2
```

```
##           E    1    0    0    1 1440
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.9983
```

```
##           95% CI : (0.9972, 0.9991)
```

```
## No Information Rate : 0.2845
```

```
## P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##           Kappa : 0.9979
```

```
##
```

```
## McNemar's Test P-Value : NA
```

```
##
```

```
## Statistics by Class:
```

```
##
```

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

```
## Sensitivity          0.9996  0.9967  0.9971  0.9992  0.9986
```

```
## Specificity          0.9995  0.9994  0.9997  0.9997  0.9997
```

```
## Pos Pred Value       0.9987  0.9974  0.9985  0.9984  0.9986
```

```
## Neg Pred Value       0.9998  0.9992  0.9994  0.9998  0.9997
```

```
## Prevalence           0.2845  0.1935  0.1744  0.1639  0.1838
```

```
## Detection Rate       0.2843  0.1928  0.1738  0.1638  0.1835
```

```
## Detection Prevalence 0.2847  0.1933  0.1741  0.1640  0.1838
```

```
## Balanced Accuracy     0.9995  0.9980  0.9984  0.9995  0.9992
```

```
confusionMatrix(gbmpred,Testing$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction    A    B    C    D    E
```

```
##           A 2226      9      0      4      1
##           B      6 1490     13      3      1
##           C      0   17 1350     14     10
##           D      0      2      5 1262     12
##           E      0      0      0      3 1418
##
## Overall Statistics
##
##           Accuracy : 0.9873
##           95% CI : (0.9845, 0.9896)
##           No Information Rate : 0.2845
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9839
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity          0.9973   0.9816   0.9868   0.9813   0.9834
## Specificity          0.9975   0.9964   0.9937   0.9971   0.9995
## Pos Pred Value       0.9938   0.9848   0.9705   0.9852   0.9979
## Neg Pred Value       0.9989   0.9956   0.9972   0.9963   0.9963
## Prevalence           0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate       0.2837   0.1899   0.1721   0.1608   0.1807
## Detection Prevalence 0.2855   0.1928   0.1773   0.1633   0.1811
## Balanced Accuracy    0.9974   0.9890   0.9903   0.9892   0.9914
```

Looking at the above tables , I choose the Random Forest model as the final model

Now predicting on pml\_testing

```
Now2<-pml_testing[,-nearZeroVar(pml_testing)]
predict(rftrain,Now2)
## [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
```

Conclusion: I have selected the Random Forest model as my final model and have obtained 100% accuracy on the pml\_testing dataset (Course project quiz)