IE 7275 Data Mining in Engineering

Weightlifting Performance Monitoring

Group No: 2

Student name: Karan Parikh, Kumaran Nehru Uthra

Executive Summary:

This case study determines the classification of different kinds of errors when an individual performs bicep curl in weightlifting using dumbbells and classify the errors with the help of data driven models. We obtained the dataset from UCI Machine learning website and it had 42896 and 53 columns with class being the errors made by the individual. We first performed PCA and built our train and test models with 70% training data and 30% being the validation data. We used four techniques to classify and predict the new dataset K-Nearest- Neighbours, Random forest, Support vector machine and Decision tree. We got the highest accuracy for random forest and recommend using that for our model based on the computation and accuracy and the lowest accuracy was obtained for KNN model.

Background and Introduction:

- Introduction

The approach we propose for the Weightlifting Exercises dataset is to investigate "how (well)" an activity was performed. The "how (well)" investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications, such as sports training. We first define quality of execution and investigate three aspects that pertain to qualitative activity recognition: the problem of specifying correct execution, the automatic and robust detection of execution mistakes, and how to provide feedback on the quality of execution to the user. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weightlifter to make sure the execution complied to the manner they were supposed to simulate.

It is significant because performing the exercise with the wrong posture leads to injuries. Also, this generation is moved towards virtual monitoring and training , so our model can be used to help athletes train and the coaches to monitor the perform of their athletes without any susceptibility to injuries and posture of their exercise for maximum efficiency and utilization of their energy in the most efficient way. It can be used by newbies/ beginners to learn and monitor their performance so that they are not prone to injuries with zero knowledge about weightlifting. It can be developed to a proper business model in the form of a virtual or a new physical product incorporating with mobile application, google assistant etc, to earn profit.

There are many erroneous positions while one performs any kind of weightlifting activity. Our aim is to predict and classify the kind of error an individual performs and eliminate the factor of lack of result due to improper form. Beginners and professional athletes usually struggle with injuries due to improper weightlifting postures and forms. they lose on their form during work out which results in incorrect muscle wear and tear and injuries. Thus, having an assisting sensor or system to analyze the fault and correct the user by giving feedback is important.

Beginners and professional athletes usually struggle with injuries due to improper weightlifting postures and forms. Either due to excessive weight or rigorous activity they lose on their form during work out which results in incorrect muscle wear and tear and injuries.

For beginners it is very important to have a proper form and technique as it builds the basis of their muscle and physique. Some of the posture injuries may result in long time disabilities. Everyone cannot afford or match time for trainers. Thus, having an assisting sensor or system to analyse the fault and correct the user by giving feedback is important.

- Data Origin:

In order to analyze the errors performed by an individual, a training database was required. A database was built by collecting sample data of exercises which were performed by six male participants aged between 20-28 years, with little weightlifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg). They were collected using sensors.

The dataset was obtained from UCI Machine learning website.

Class A – specified execution of the exercise

Class B-Throwing 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

Following is the snippet of how the data looks.

```
   user_name    raw_timestamp_part_1 raw_timestamp_part_2          cvtd_timestamp new_window    num_window      roll_belt        pitch_belt
 adelmo  : 311   Min.   :1.322e+09   Min.   :   297          2/12/2011 13:35 : 311   no :3936   Min.   : 1.00   Min.   :-28.90   Min.   :-56.20
 carlitos:1580   1st Qu.:1.323e+09   1st Qu.:244321          28/11/2011 14:15:  88   yes:  88   1st Qu.:24.00   1st Qu.:  1.38   1st Qu.:  6.22
 eurico  :  88   Median :1.323e+09   Median :492342          30/11/2011 17:12:   4              Median :46.00   Median :122.00   Median : 25.50
 jeremy  :   4   Mean   :1.323e+09   Mean   :490377          5/12/2011 11:23 : 337              Mean   :46.33   Mean   : 73.31   Mean   : 14.16
 pedro   :2041   3rd Qu.:1.323e+09   3rd Qu.:736278          5/12/2011 11:25 :1243              3rd Qu.:69.00   3rd Qu.:124.00   3rd Qu.: 26.40
                 Max.   :1.323e+09   Max.   :996453          5/12/2011 14:22 : 456              Max.   :91.00   Max.   :159.00   Max.   : 60.30
    yaw_belt          total_accel_belt kurtosis_roll_belt kurtosis_picth_belt skewness_roll_belt skewness_roll_belt.1 max_roll_belt     max_picth_belt
 Min.   :-179.000   Min.   : 0.00   Min.   :-3.333              :3936     Min.   :-3.032              :3936     Min.   :-94.40   Min.   : 3.00
 1st Qu.: -93.100   1st Qu.: 3.00   1st Qu.:-1.376     #DIV/0!   :   7     1st Qu.:-0.629     #DIV/0!  :   7     1st Qu.:-92.20   1st Qu.: 5.00
 Median :  -4.940   Median :19.00   Median :-1.036     -0.15095  :   2     Median : 0.005     -1.717824:   2     Median : -4.10   Median :20.00
 Mean   : -30.975   Mean   :12.77   Mean   :-0.624     -2.060105:   2     Mean   :-0.065     -0.045472:   1     Mean   :-28.15   Mean   :14.07
 3rd Qu.:  -2.695   3rd Qu.:20.00   3rd Qu.:-0.518     0.988872 :   2     3rd Qu.: 0.425     -0.04606 :   1     3rd Qu.: -1.45   3rd Qu.:21.00
 Max.   : 179.000   Max.   :26.00   Max.   : 7.515     -0.06016 :   1     Max.   : 2.713     -0.059758:   1     Max.   :179.00   Max.   :26.00
  max_yaw_belt     min_roll_belt      min_pitch_belt    min_yaw_belt     amplitude_roll_belt amplitude_pitch_belt amplitude_yaw_belt var_total_accel_belt
 Min.   :-3.300   Min.   :-179.00   Min.   : 0.00   Min.   :-3.300   Min.   :  0.000     Min.   : 0.000       Min.   :0          Min.   : 0.000
 1st Qu.:-1.400   1st Qu.: -93.30   1st Qu.: 2.00   1st Qu.:-1.400   1st Qu.:  0.650     1st Qu.: 1.000       1st Qu.:0          1st Qu.: 0.200
 Median :-1.000   Median :  -7.25   Median :18.00   Median :-1.000   Median :  1.345     Median : 2.000       Median :0          Median : 0.300
 Mean   :-0.619   Mean   : -34.13   Mean   :11.44   Mean   :-0.619   Mean   :  5.980     Mean   : 2.625       Mean   :0          Mean   : 0.978
 3rd Qu.:-0.500   3rd Qu.:  -3.60   3rd Qu.:19.00   3rd Qu.:-0.500   3rd Qu.:  2.683     3rd Qu.: 3.000       3rd Qu.:0          3rd Qu.: 0.925
 Max.   : 7.500   Max.   : 157.00   Max.   :20.00   Max.   : 7.500   Max.   :358.000     Max.   :21.000       Max.   :0          Max.   :18.200
 avg_roll_belt     stddev_roll_belt var_roll_belt     avg_pitch_belt    stddev_pitch_belt var_pitch_belt    avg_yaw_belt      stddev_yaw_belt    var_yaw_belt
 Min.   :-27.40   Min.   :0.000   Min.   : 0.000   Min.   :-49.40   Min.   :0.000   Min.   : 0.000   Min.   :-94.400   Min.   :  0.000   Min.   :    0.000
 1st Qu.:  1.35   1st Qu.:0.300   1st Qu.: 0.100   1st Qu.:  6.10   1st Qu.:0.200   1st Qu.: 0.000   1st Qu.:-92.900   1st Qu.:  0.200   1st Qu.:    0.030
 Median :121.90   Median :0.600   Median : 0.350   Median : 25.75   Median :0.350   Median : 0.100   Median : -4.950   Median :  0.400   Median :    0.170
 Mean   : 72.63   Mean   :1.774   Mean   : 8.907   Mean   : 13.99   Mean   :0.683   Mean   : 1.416   Mean   :-30.774   Mean   :  2.455   Mean   :  303.176
 3rd Qu.:123.75   3rd Qu.:1.625   3rd Qu.: 2.625   3rd Qu.: 26.32   3rd Qu.:0.725   3rd Qu.: 0.600   3rd Qu.: -2.525   3rd Qu.:  0.825   3rd Qu.:    0.698
 Max.   :154.50   Max.   :8.500   Max.   :71.800   Max.   : 59.70   Max.   :6.200   Max.   :39.000   Max.   :158.600   Max.   :163.100   Max.   :26610.320
  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
 Min.   :-0.7900   Min.   :-0.470000   Min.   :-0.7700   Min.   :-120.00   Min.   :-71.00   Min.   :-244.00   Min.   :-30.00   Min.   :428.0   Min.   :-513.0
 1st Qu.:-0.4300   1st Qu.:-0.030000   1st Qu.:-0.4600   1st Qu.: -42.00   1st Qu.:  4.00   1st Qu.:-176.00   1st Qu.: -3.00   1st Qu.:577.0   1st Qu.:-379.0
 Median :-0.2400   Median :-0.020000   Median :-0.4100   Median : -34.00   Median : 65.00   Median :-166.00   Median :  2.00   Median :585.0   Median :-366.0
 Mean   :-0.1823   Mean   :-0.008837   Mean   :-0.2464   Mean   : -24.36   Mean   : 39.84   Mean   : -94.73   Mean   : 24.65   Mean   :582.7   Mean   :-340.9
 3rd Qu.: 0.0200   3rd Qu.: 0.000000   3rd Qu.:-0.0200   3rd Qu.: -16.00   3rd Qu.: 70.00   3rd Qu.:  20.00   3rd Qu.:  8.00   3rd Qu.:601.0   3rd Qu.:-311.0
 Max.   : 2.0200   Max.   : 0.420000   Max.   : 0.8200   Max.   :  80.00   Max.   :164.00   Max.   :  77.00   Max.   :485.00   Max.   :652.0   Max.   : 293.0
    roll_arm          pitch_arm          yaw_arm          total_accel_arm var_accel_arm     avg_roll_arm      stddev_roll_arm   var_roll_arm
 Min.   :-180.00   Min.   :-87.100   Min.   :-180.000   Min.   : 1.00   Min.   :  0.00   Min.   :-169.69   Min.   :  0.000   Min.   :    0.00
 1st Qu.: -34.40   1st Qu.:-32.200   1st Qu.: -59.675   1st Qu.:15.00   1st Qu.: 22.41   1st Qu.: -44.13   1st Qu.:  4.588   1st Qu.:   21.06
 Median :  72.10   Median : -8.645   Median :  17.500   Median :25.00   Median : 65.10   Median :  76.22   Median : 16.104   Median :  259.36
 Mean   :  40.01   Mean   :-10.539   Mean   :   2.768   Mean   :24.89   Mean   : 73.66   Mean   :  37.31   Mean   : 21.820   Mean   : 1367.55
 3rd Qu.: 124.00   3rd Qu.: 14.600   3rd Qu.:  72.825   3rd Qu.:34.00   3rd Qu.:103.18   3rd Qu.: 118.52   3rd Qu.: 25.690   3rd Qu.:  660.05
 Max.   : 180.00   Max.   : 81.400   Max.   : 180.000   Max.   :59.00   Max.   :253.01   Max.   : 160.78   Max.   :161.964   Max.   :26232.21
 avg_pitch_arm     stddev_pitch_arm var_pitch_arm     avg_yaw_arm       stddev_yaw_arm    var_yaw_arm       gyros_arm_x       gyros_arm_y       gyros_arm_z
 Min.   :-57.29   Min.   : 0.000   Min.   :  0.00   Min.   :-164.64   Min.   :  0.00   Min.   :    0.0   Min.   :-5.2000   Min.   :-3.4400   Min.   :-2.17000
 1st Qu.:-29.37   1st Qu.: 5.688   1st Qu.: 32.35   1st Qu.: -42.61   1st Qu.: 15.58   1st Qu.:  242.8   1st Qu.:-2.0925   1st Qu.:-0.9200   1st Qu.:-0.20000
 Median :-10.17   Median :10.667   Median :113.80   Median :  19.06   Median : 35.88   Median : 1287.5   Median :-0.0200   Median :-0.0300   Median : 0.00000
 Mean   : -9.44   Mean   :12.279   Mean   :220.81   Mean   :   2.69   Mean   : 36.57   Mean   : 2188.3   Mean   :-0.1852   Mean   :-0.1818   Mean   : 0.04444
 3rd Qu.: 12.17   3rd Qu.:18.158   3rd Qu.:330.08   3rd Qu.:  54.38   3rd Qu.: 50.66   3rd Qu.: 2566.6   3rd Qu.: 1.7000   3rd Qu.: 0.5800   3rd Qu.: 0.28000
 Max.   : 54.60   Max.   :30.778   Max.   :947.27   Max.   : 148.45   Max.   :177.04   Max.   :31344.6   Max.   : 4.3400   Max.   : 2.4600   Max.   : 3.02000
```

```
  accel_arm_x       accel_arm_y       accel_arm_z       magnet_arm_x     magnet_arm_y     magnet_arm_z    kurtosis_roll_arm kurtosis_picth_arm kurtosis_yaw_a
 Min.    :-346.00   Min.    :-252.00   Min.    :-538.00   Min.    :-515.0   Min.    :-392.0   Min.    :-573.0          :3936            :3936            :3936
 1st Qu.: -88.00   1st Qu.: -21.00   1st Qu.:-124.00   1st Qu.:-332.0   1st Qu.: -13.0   1st Qu.:  -1.0   -0.11926:   1    #DIV/0! :   5    #DIV/0! :   8
 Median :  24.00   Median :  22.00   Median :   6.00   Median : 278.5   Median : 267.0   Median : 431.0   -0.19002:   1    -0.10176:   1    -0.06791:   1
 Mean   :  34.38   Mean   :  26.87   Mean   : -41.39   Mean   : 194.3   Mean   : 161.7   Mean   : 253.2   -0.20488:   1    -0.15381:   1    -0.12096:   1
 3rd Qu.: 136.00   3rd Qu.:  96.25   3rd Qu.:  76.00   3rd Qu.: 651.0   3rd Qu.: 348.0   3rd Qu.: 515.0   -0.34389:   1    -0.15426:   1    -0.1697 :   1
 Max.   : 434.00   Max.   : 229.00   Max.   : 209.00   Max.   : 782.0   Max.   : 482.0   Max.   : 647.0   -0.36227:   1    -0.16444:   1    -0.20332:   1
 skewness_roll_arm skewness_pitch_arm skewness_yaw_arm  max_roll_arm     max_picth_arm     max_yaw_arm      min_roll_arm     min_pitch_arm      min_yaw_arm
         :3936            :3936            :3936       Min.   :-36.300   Min.   :-164.00   Min.   : 3.00   Min.   :-87.100   Min.   :-180.00   Min.   : 1.00
 -0.00696:   1    #DIV/0! :   5    #DIV/0! :   8       1st Qu.: -4.475   1st Qu.:  37.12   1st Qu.:34.00   1st Qu.:-57.675   1st Qu.:-110.25   1st Qu.: 5.00
 -0.01884:   1    -0.01247:   1    -0.0046 :   1       Median :  8.450   Median :  77.25   Median :38.00   Median :-33.600   Median : -58.60   Median :10.00
 -0.03359:   1    -0.09627:   1    -0.008  :   1       Mean   :  9.725   Mean   :  56.34   Mean   :38.16   Mean   :-27.824   Mean   : -55.45   Mean   :12.83
 -0.08596:   1    -0.10319:   1    -0.00863:   1       3rd Qu.: 21.925   3rd Qu.: 118.25   3rd Qu.:46.00   3rd Qu.: -1.875   3rd Qu.:  -2.25   3rd Qu.:17.25
 -0.1009 :   1    -0.14513:   1    -0.05777:   1       Max.   : 81.400   Max.   : 180.00   Max.   :59.00   Max.   : 35.700   Max.   : 146.00   Max.   :34.00
 amplitude_roll_arm amplitude_pitch_arm amplitude_yaw_arm roll_dumbbell      pitch_dumbbell     yaw_dumbbell      kurtosis_roll_dumbbell kurtosis_picth_dumbbe
 Min.   : 0.00     Min.   :  0.0      Min.   : 0.00     Min.   :-152.782   Min.   :-134.73   Min.   :-129.33   Min.   :-2.089      Min.   :-2.089
 1st Qu.:21.77     1st Qu.: 59.8      1st Qu.:17.00     1st Qu.: -34.657   1st Qu.: -12.93   1st Qu.:  21.35   1st Qu.:-0.725      1st Qu.:-0.906
 Median :36.95     Median :121.5      Median :27.00     Median :  -2.295   Median :  14.48   Median :  72.49   Median :-0.096      Median :-0.442
 Mean   :37.55     Mean   :111.8      Mean   :25.33     Mean   :   3.500   Mean   :   5.18   Mean   :  55.66   Mean   : 0.328      Mean   : 0.057
 3rd Qu.:55.75     3rd Qu.:153.8      3rd Qu.:35.00     3rd Qu.:  58.014   3rd Qu.:  27.95   3rd Qu.: 122.01   3rd Qu.: 0.754      3rd Qu.: 0.335
 Max.   :90.00     Max.   :360.0      Max.   :52.00     Max.   : 139.729   Max.   :  97.28   Max.   : 152.92   Max.   : 7.563      Max.   :11.273
 skewness_roll_dumbbell skewness_pitch_dumbbell max_roll_dumbbell max_picth_dumbbell max_yaw_dumbbell min_roll_dumbbell min_pitch_dumbbell min_yaw_dumbbell
 Min.   :-2.611      Min.   :-2.050      Min.   :-70.90   Min.   :-84.50   Min.   :-2.100   Min.   :-134.70   Min.   :-129.30   Min.   :-2.100
 1st Qu.:-0.510      1st Qu.:-0.507      1st Qu.: 21.27   1st Qu.: 58.98   1st Qu.:-0.700   1st Qu.: -57.02   1st Qu.: -39.48   1st Qu.:-0.700
 Median : 0.082      Median :-0.216      Median : 41.85   Median :133.00   Median :-0.100   Median : -26.75   Median :  20.20   Median :-0.100
 Mean   : 0.015      Mean   :-0.090      Mean   : 34.27   Mean   : 88.11   Mean   : 0.331   Mean   : -27.52   Mean   :  15.46   Mean   : 0.331
 3rd Qu.: 0.464      3rd Qu.: 0.319      3rd Qu.: 56.42   3rd Qu.:139.30   3rd Qu.: 0.725   3rd Qu.:   6.05   3rd Qu.:  92.88   3rd Qu.: 0.725
 Max.   : 2.381      Max.   : 2.783      Max.   : 97.30   Max.   :152.90   Max.   : 7.600   Max.   :  26.80   Max.   : 122.90   Max.   : 7.600
 amplitude_roll_dumbbell amplitude_pitch_dumbbell amplitude_yaw_dumbbell total_accel_dumbbell var_accel_dumbbell avg_roll_dumbbell stddev_roll_dumbbell
 Min.   :  0.00      Min.   :  0.00      Min.   :0      Min.   : 1.00   Min.   :  0.000   Min.   :-110.933   Min.   :  0.00
 1st Qu.: 34.08      1st Qu.: 31.07      1st Qu.:0      1st Qu.: 6.00   1st Qu.:  0.656   1st Qu.: -35.513   1st Qu.: 10.51
 Median : 55.71      Median : 54.74      Median :0      Median : 9.00   Median :  2.416   Median :  -5.118   Median : 17.06
 Mean   : 61.79      Mean   : 72.65      Mean   :0      Mean   :12.02   Mean   :  9.482   Mean   :   2.662   Mean   : 26.32
 3rd Qu.: 87.70      3rd Qu.:110.91      3rd Qu.:0      3rd Qu.:14.00   3rd Qu.:  7.531   3rd Qu.:  52.649   3rd Qu.: 34.67
 Max.   :171.75      Max.   :217.33      Max.   :0      Max.   :37.00   Max.   :230.428   Max.   : 117.404   Max.   :103.12
 var_roll_dumbbell avg_pitch_dumbbell stddev_pitch_dumbbell var_pitch_dumbbell avg_yaw_dumbbell  stddev_yaw_dumbbell var_yaw_dumbbell  gyros_dumbbell_x
 Min.   :    0.0   Min.   :-70.916   Min.   : 0.000      Min.   :   0.00   Min.   :-105.65   Min.   : 0.000      Min.   :   0.00   Min.   :-1.4300
 1st Qu.:  110.5   1st Qu.:-10.014   1st Qu.: 7.475      1st Qu.:  55.88   1st Qu.:  25.94   1st Qu.: 6.987      1st Qu.:  48.85   1st Qu.:-0.0200
 Median :  291.0   Median : 13.931   Median :14.106      Median : 199.08   Median :  64.71   Median :13.575      Median : 184.56   Median : 0.3200
 Mean   : 1359.3   Mean   :  3.483   Mean   :15.197      Mean   : 342.30   Mean   :  51.08   Mean   :18.870      Mean   : 578.32   Mean   : 0.2487
 3rd Qu.: 1202.1   3rd Qu.: 25.301   3rd Qu.:22.125      3rd Qu.: 489.82   3rd Qu.: 118.01   3rd Qu.:30.262      3rd Qu.: 915.77   3rd Qu.: 0.5300
 Max.   :10634.5   Max.   : 57.453   Max.   :48.430      Max.   :2345.44   Max.   : 129.93   Max.   :71.060      Max.   :5049.47   Max.   : 1.4800
 gyros_dumbbell_y   gyros_dumbbell_z  accel_dumbbell_x   accel_dumbbell_y  accel_dumbbell_z  magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z
 Min.   :-2.04000   Min.   :-1.4600   Min.   :-237.000   Min.   :-163.00   Min.   :-273.00   Min.   :-638.00   Min.   :-730.0   Min.   :-262.00
 1st Qu.:-0.27000   1st Qu.:-0.3300   1st Qu.:  -6.000   1st Qu.: -28.00   1st Qu.:  12.00   1st Qu.:-515.00   1st Qu.:-544.0   1st Qu.:-101.00
 Median :-0.06000   Median :-0.1300   Median :  11.000   Median :  -2.00   Median :  51.00   Median : 107.50   Median :-486.0   Median : -59.00
 Mean   :-0.04674   Mean   :-0.1337   Mean   :  -7.091   Mean   :  12.83   Mean   :  16.63   Mean   :  10.55   Mean   :-115.7   Mean   : -41.12
 3rd Qu.: 0.14000   3rd Qu.: 0.0500   3rd Qu.:  23.000   3rd Qu.:  47.00   3rd Qu.:  79.00   3rd Qu.: 506.00   3rd Qu.: 304.0   3rd Qu.:   1.00
 Max.   : 4.37000   Max.   : 1.8900   Max.   : 217.000   Max.   : 281.00   Max.   : 122.00   Max.   : 579.00   Max.   : 618.0   Max.   : 300.00

  roll_forearm     pitch_forearm     yaw_forearm      max_roll_forearm max_picth_forearm min_roll_forearm min_pitch_forearm amplitude_roll_forearm
 Min.   :-180.0   Min.   :-64.00   Min.   :-180.00   Min.   :-63.900   Min.   :-152.0   Min.   :-64.000   Min.   :-180.00   Min.   : 0.00
 1st Qu.:-115.0   1st Qu.:  0.00   1st Qu.:-106.00   1st Qu.:  3.475   1st Qu.: 105.8   1st Qu.: -2.825   1st Qu.:-177.00   1st Qu.: 2.69
 Median :  89.5   Median : 19.70   Median :  83.50   Median : 49.600   Median : 168.0   Median :  4.650   Median :-168.50   Median :32.20
 Mean   :  36.1   Mean   : 18.57   Mean   :  17.79   Mean   : 33.587   Mean   : 111.8   Mean   :  3.006   Mean   : -81.85   Mean   :30.58
 3rd Qu.: 136.0   3rd Qu.: 43.90   3rd Qu.: 108.00   3rd Qu.: 59.975   3rd Qu.: 176.0   3rd Qu.: 16.200   3rd Qu.:  79.03   3rd Qu.:50.55
 Max.   : 180.0   Max.   : 86.90   Max.   : 180.00   Max.   : 86.900   Max.   : 180.0   Max.   : 47.500   Max.   : 125.00   Max.   :77.10
 amplitude_pitch_forearm total_accel_forearm var_accel_forearm avg_roll_forearm stddev_roll_forearm var_roll_forearm   avg_pitch_forearm stddev_pitch_forearm
 Min.   :  0.00      Min.   :10.00   Min.   :  0.000   Min.   :-145.14   Min.   :  0.000   Min.   :    0.000   Min.   :-63.900   Min.   : 0.000
 1st Qu.:  3.75      1st Qu.:30.00   1st Qu.:  4.027   1st Qu.:  -1.93   1st Qu.:  1.025   1st Qu.:    1.117   1st Qu.:  0.115   1st Qu.: 0.803
 Median :341.50      Median :35.00   Median : 14.077   Median :  27.86   Median : 45.163   Median : 2749.163   Median : 25.356   Median : 8.907
 Mean   :193.66      Mean   :34.38   Mean   : 30.004   Mean   :  40.28   Mean   : 67.767   Mean   : 9143.039   Mean   : 17.501   Mean   : 9.544
 3rd Qu.:352.00      3rd Qu.:37.00   3rd Qu.: 52.023   3rd Qu.:  91.12   3rd Qu.:140.739   3rd Qu.:19807.335   3rd Qu.: 40.072   3rd Qu.:15.718
 Max.   :359.00      Max.   :59.00   Max.   :124.178   Max.   : 151.25   Max.   :176.478   Max.   :31144.560   Max.   : 68.168   Max.   :26.729
 var_pitch_forearm avg_yaw_forearm   stddev_yaw_forearm var_yaw_forearm    gyros_forearm_x   gyros_forearm_y     gyros_forearm_z    accel_forearm_x
 Min.   :  0.000   Min.   :-152.33   Min.   :  0.000   Min.   :    0.00   Min.   :-1.8800   Min.   :-5.730000   Min.   :-2.58000   Min.   :-328.000
 1st Qu.:  0.651   1st Qu.: -26.37   1st Qu.:  1.026   1st Qu.:    1.07   1st Qu.:-0.1400   1st Qu.:-1.780000   1st Qu.:-0.31000   1st Qu.:-117.000
 Median : 79.335   Median :  17.09   Median : 74.276   Median : 5541.96   Median : 0.0600   Median :-0.020000   Median :-0.02000   Median :  -6.000
 Mean   :154.157   Mean   :  18.84   Mean   : 61.932   Mean   : 7187.42   Mean   : 0.1076   Mean   :-0.004108   Mean   : 0.09302   Mean   :  -6.445
 3rd Qu.:247.237   3rd Qu.:  89.45   3rd Qu.:116.847   3rd Qu.:13653.25   3rd Qu.: 0.4200   3rd Qu.: 1.830000   3rd Qu.: 0.48000   3rd Qu.: 113.000
 Max.   :714.453   Max.   : 132.59   Max.   :197.508   Max.   :39009.33   Max.   : 1.8100   Max.   : 5.170000   Max.   : 3.35000   Max.   : 279.000
 accel_forearm_y   accel_forearm_z magnet_forearm_x  magnet_forearm_y magnet_forearm_z classe
 Min.   :-467.00   Min.   :-366   Min.   :-1160.0   Min.   :-725.0   Min.   :-876.0   A:1365
 1st Qu.:  75.75   1st Qu.:-210   1st Qu.: -589.0   1st Qu.: -76.0   1st Qu.: 370.8   B: 901
 Median : 229.50   Median :-181   Median : -330.5   Median : 653.0   Median : 560.0   C: 112
 Mean   : 171.47   Mean   :-163   Mean   : -348.7   Mean   : 358.6   Mean   : 475.2   D: 276
 3rd Qu.: 297.00   3rd Qu.:-150   3rd Qu.: -152.0   3rd Qu.: 747.0   3rd Qu.: 670.0   E:1370
 Max.   : 575.00   Max.   : 239   Max.   :  413.0   Max.   :1440.0   Max.   :1040.0
```

- Data collection:
  - Sample is collected and stored using on-body sensing approach and ambient sensing approach, which is then logged.
  - If the database has too many data information, PCA might be used to reduce the number of features.
  - The output from the logged files are saved into a CSV file, containing 42896 rows and 53 columns
  - Following are some applications where the weightlifting monitoring can be helpful:
    - Help personal trainer and training assistants to correct and gauge the weightlifting performance of the athletes more accurately and seek for future injuries

- Develop a device for beginners to check whether they are training effectively or not
- Gauging one's performance in terms of correct posture and technique

The outcome of this model is Class A, Class B, Class C, Class D and Class E that tells us what kind of error the person deals with. The plan is to apply various statistical methods and machine learning algorithms to build the model. The performances among different methods is then compared for observing the factors that might affect the performance of the prediction

Data mining techniques and Implementation:

K-Nearest Neighbor

- Classification was done based on K-nearest neighbour algorithm.
- Converted the variable Classes into factors.
- Split the sample data into 70% training data and 30% validation data.
- Target variable was determined, and the predictors were chosen.
- Trained the model using K-nearest neighbour with predictors and target variable.
- We predicted this model on the validation data.
- Confusion matrix was made, and the accuracy of the model was determined with different K-values and the best value is chosen from that.

Decision Tree

- Classification was done based on Decision tree.
- Converted the variable Classes into factors.
- Split the sample data into 70% training data and 30% validation data.
- Target variable was determined, and the predictors were chosen.
- Trained the model using decision tree with predictors and target variable.
- We predicted this model on the validation data.
- Confusion matrix was made, and the accuracy of the model was determined.

Support Vector Machine

- Classification was done based on Support Vector Machine algorithm.
- Converted the variable Classes into factors.
- Split the sample data into 70% training data and 30% validation data.
- Target variable was determined, and the predictors were chosen.
- Trained the model using support vector machine with predictors and target variable.
- We predicted this model on the validation data.
- Confusion matrix was made, and the accuracy of the model was determined.

Random Forest

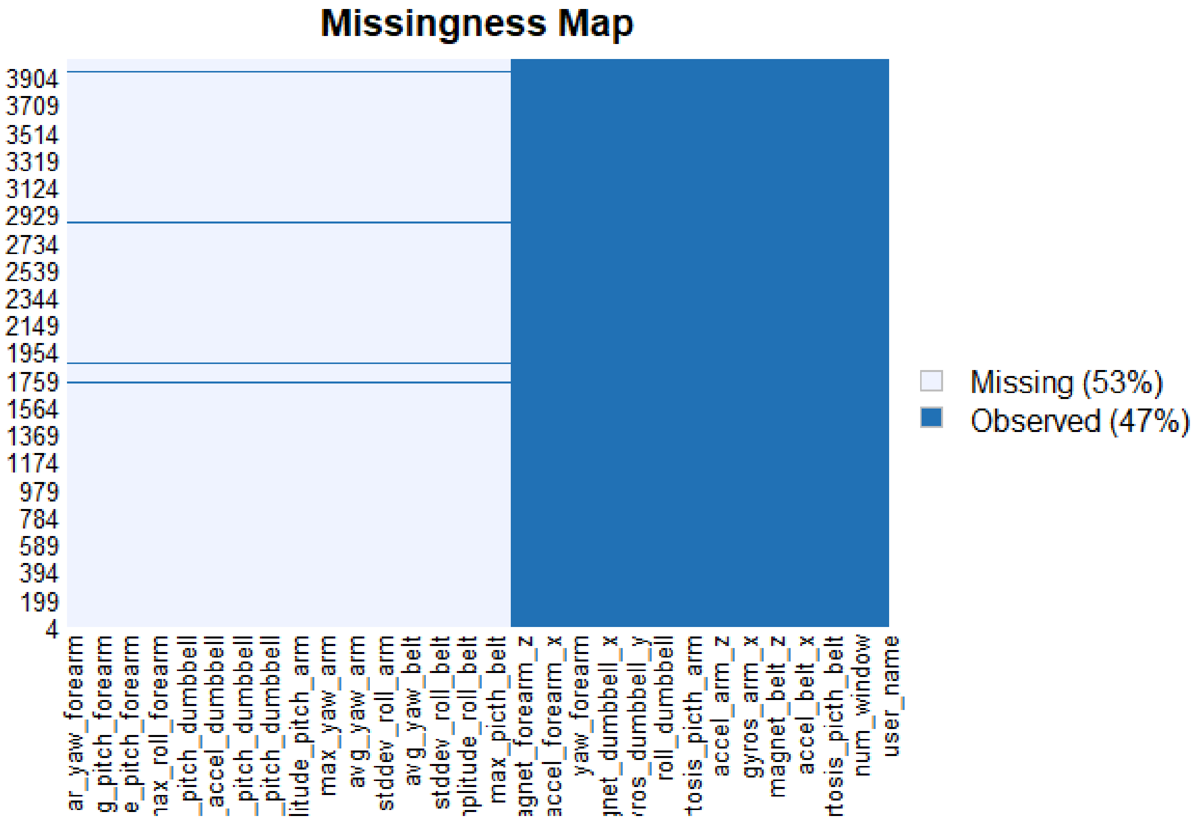
- Classification was done based on Random forest algorithm.
- Converted the variable Classes into factors.

- Split the sample data into 70% training data and 30% validation data.
- Target variable was determined, and the predictors were chosen.
- Trained the model using Random forest with predictors and target variable.
- We predicted this model on the validation data.
- Confusion matrix was made, and the accuracy of the model was determined.

Data Exploration and Visualization:

1. Missing map

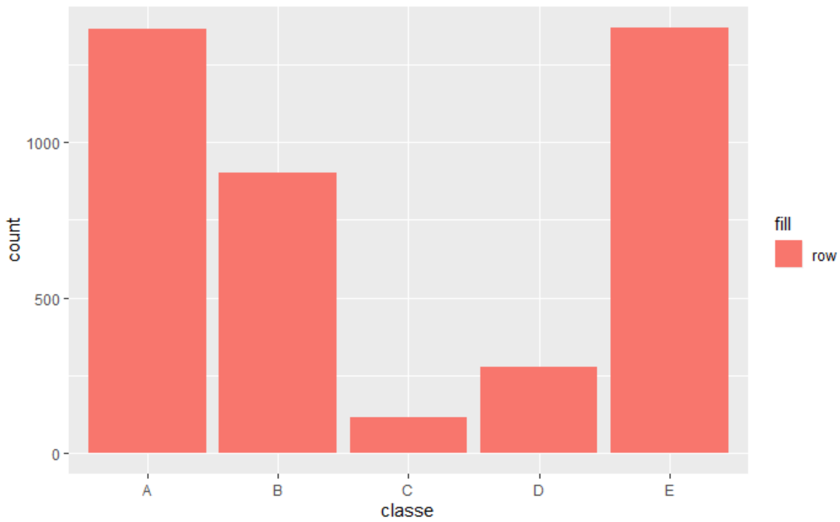
The first step in this is to eliminate all the missing values and changing all the character type to numerical type. All the character Yes/No values were changed to binary 1/0 values.



As you can see there are a lot of missing values which needs to be eliminated in the dataset. We eliminate them in the pre-processing stage.

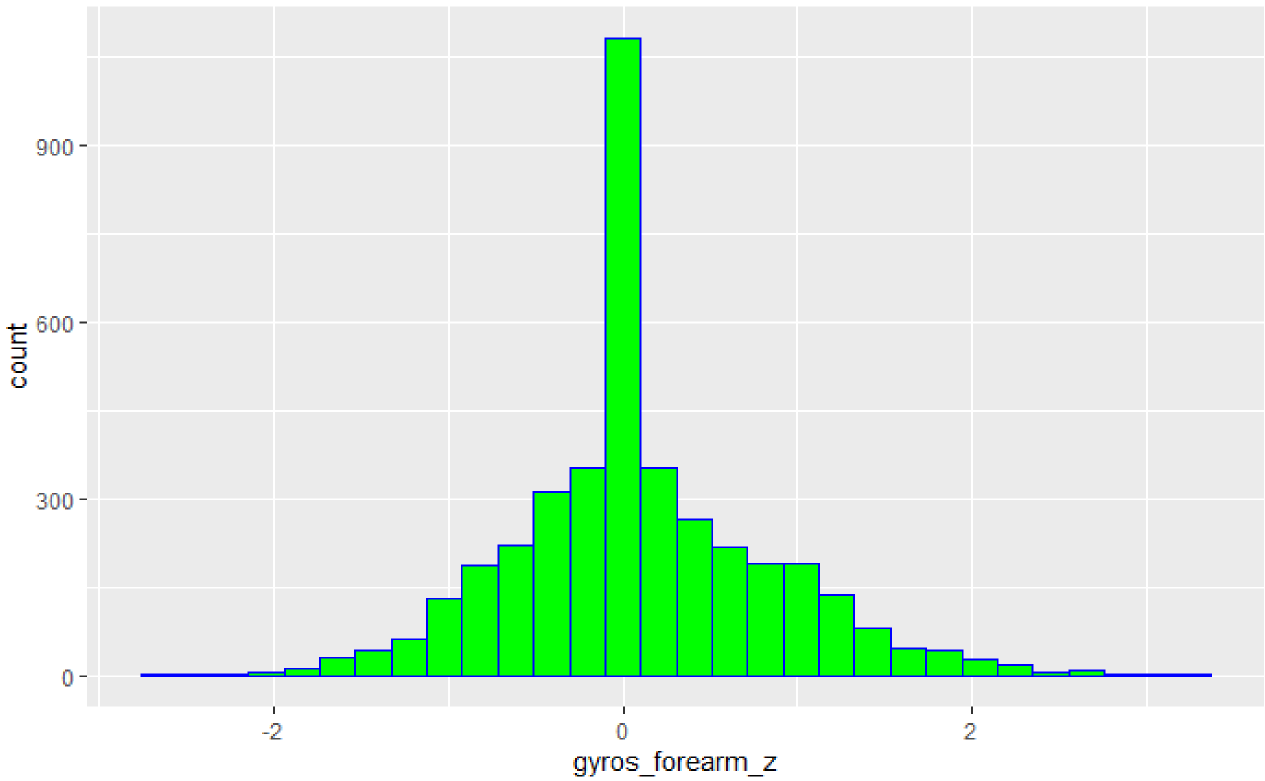
2. Bar chart

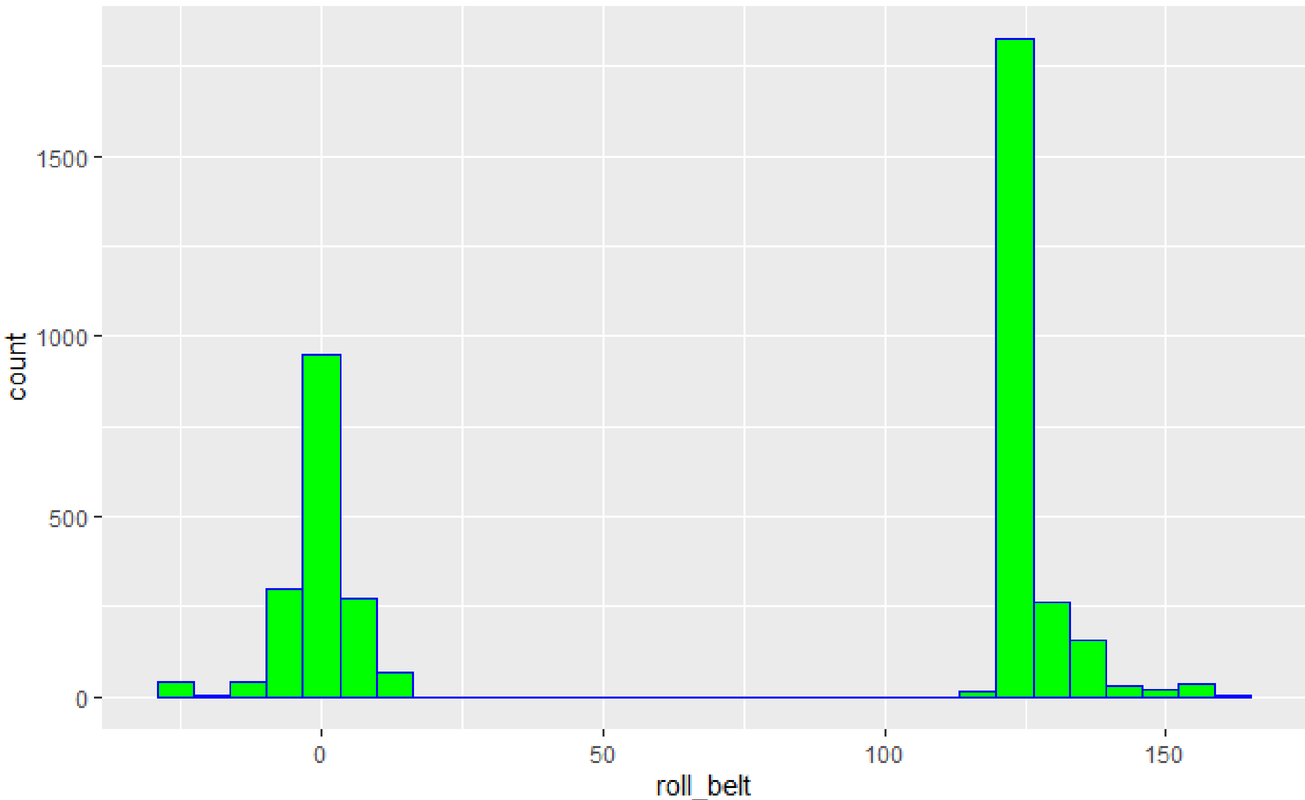
We plot a bar chart to know how the data of the classified variables which the required classes of weightlifting error are occurs. This helps us to identify the total distribution of all the results which we want to be able to predict. It is a good representation of what kind of errors are more common and how usually the errors are performed. This shows that most of the errors occur are of Class E and many perform them properly that is Class A.



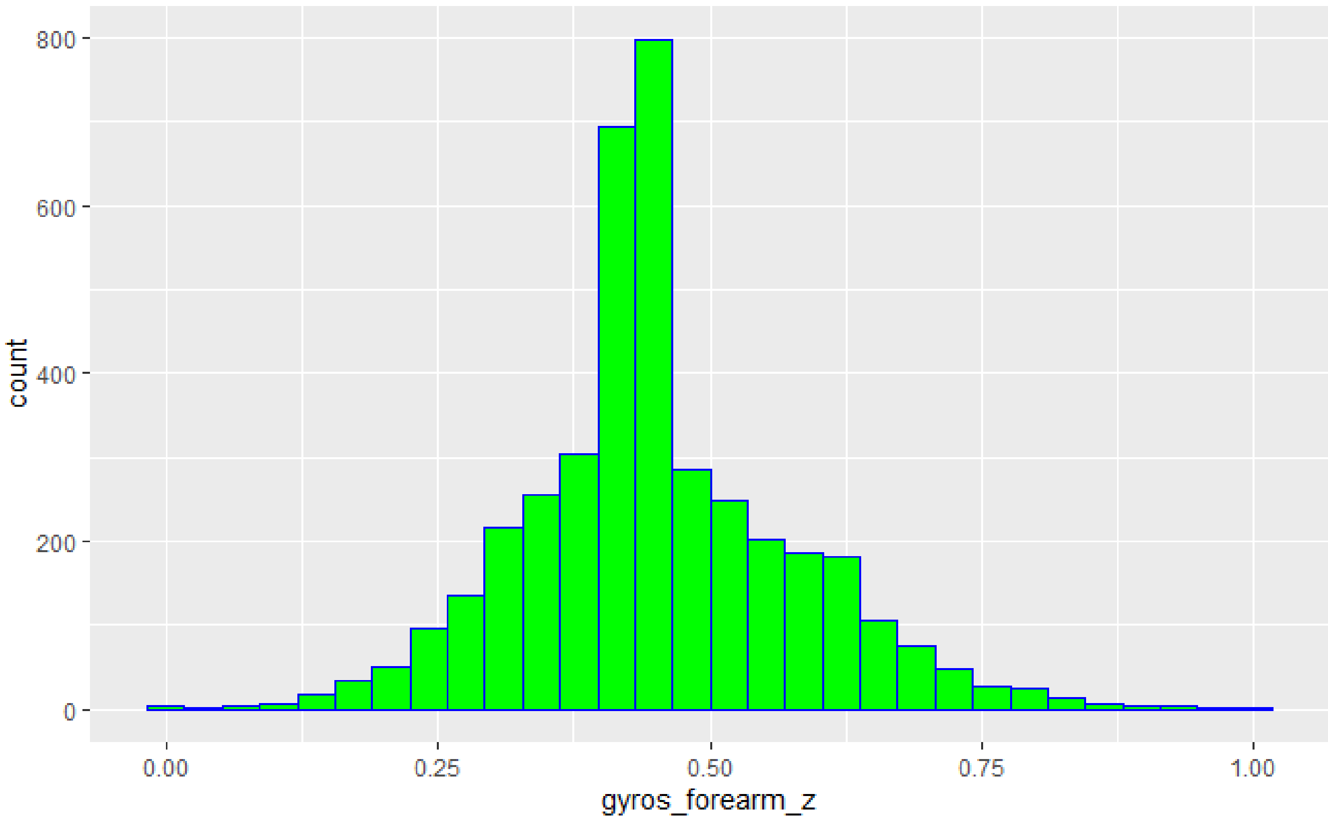
3. Normalize values

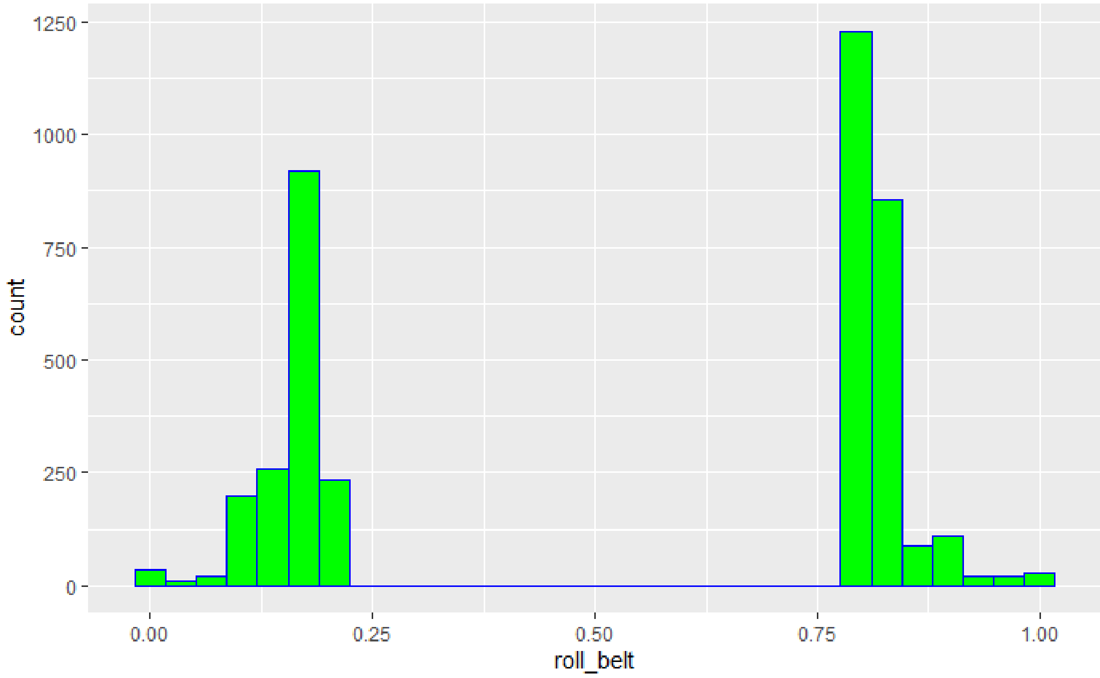
We plot a histogram to show the distribution of the data. It can be depicted from the histograms that the distribution does not seem unusual. The graph of certain data points like gyros_forearm_z does not need to be normalized that much but other data points like roll_belt requires to be normalized.





After normalising the data seems to be to be fit to be used for data preprocessing, ptediction and classifications.

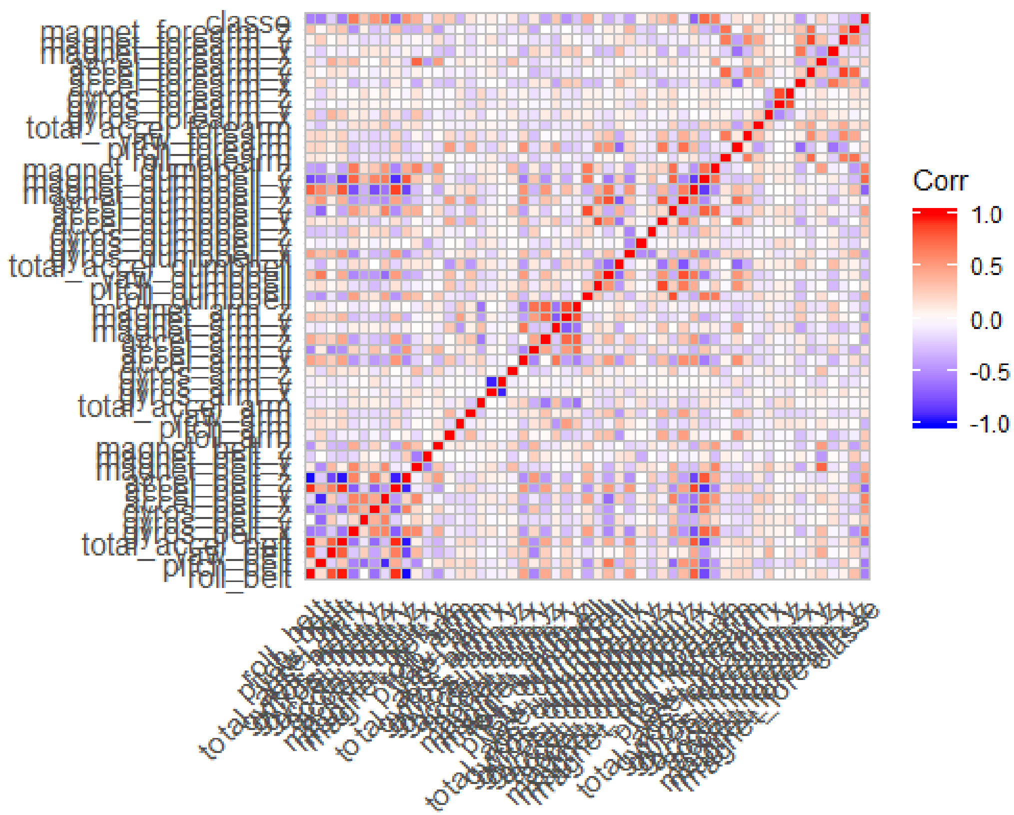




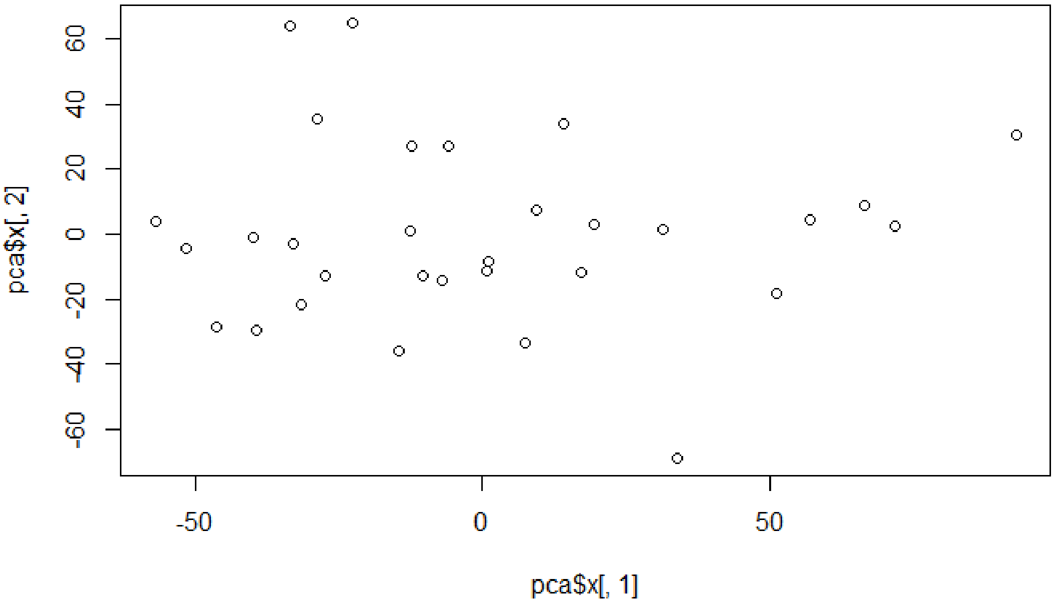
The between gap is due to the missing valus which we eliminate while preprocessing.

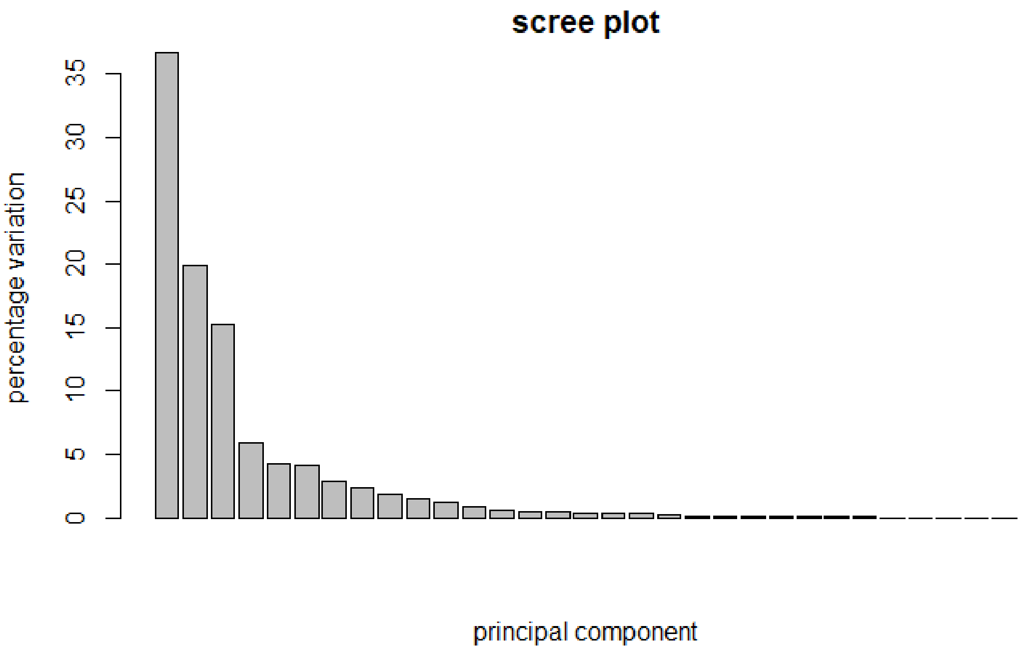
Data Pre-processing and preparation

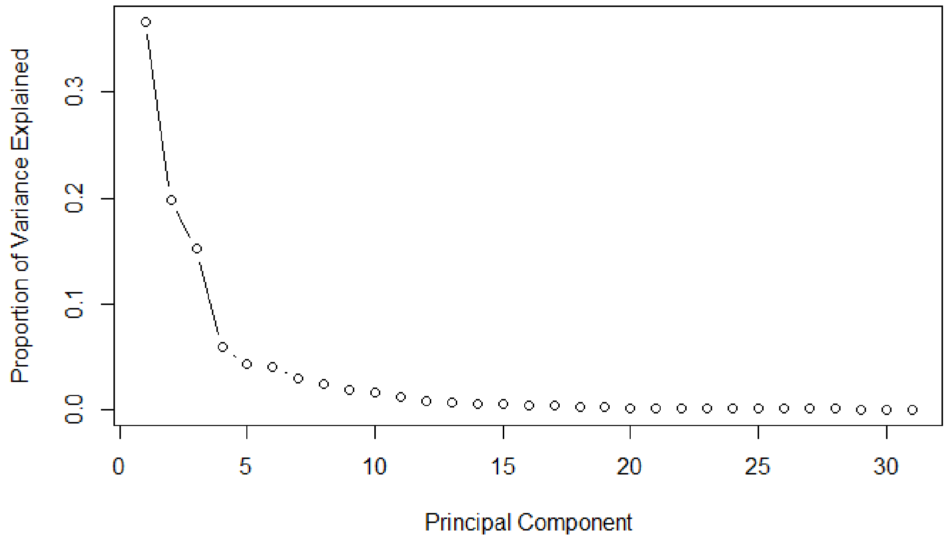
From the below correlation plots it can be observed that there are many features which are highly correlated with each other. We are considering a feature highly correlated with another feature when the correlation between them has a value greater than 1 or less than -1. These are due to the standard values and missing values in the dataset.



To deal with this issue it is required to perform the Principal Component Analysis (PCA) on the data and then choose the significant component scores that are required to build and train a model. Following is the plot which shows the proportionality of variance for all the 53 principal components generated. Out of 53 components we are choosing the first 3 components to build and train the model as it gives the highest accuracy for training and validation data.



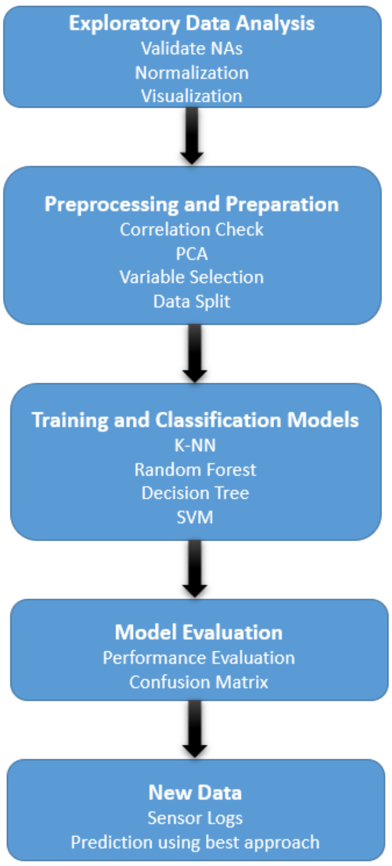




The next step would be to split the dataset into training and validation data. We have split the data into 7:3 ratio, that means we have randomly assigned 70% of data for training and 30% of data for validation.

Data Mining Techniques and implementation

We worked on four different approach for this classification problem. The final required outcome was to have to predict the class of error.



Performance evaluation

Based on the consideration of accuracy and the time of execution, we picked Random Forest model to be our best approach.

Data Mining Technique	Accuracy	Computation Time
KNN	92.3%	14.5s
Random Forest	99.5%	11.71s
Decision Tree	96.3%	14.36
SVM	99.5%	25.97s

- Results:

➢ KNN:

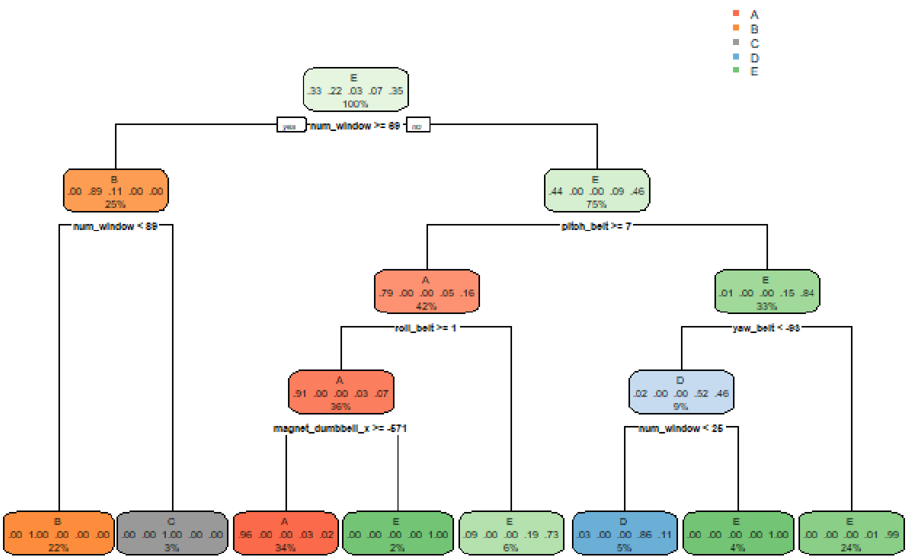
```
Confusion matrix:
    A   B  C   D   E class.error
A 954   1  0   0   0 0.001047120
B   6 625  0   0   0 0.009508716
C   0   3 75   0   0 0.038461538
D   0   0  0 193   0 0.000000000
E   0   0  0   0 959 0.000000000
Confusion Matrix and Statistics

          Reference
Prediction   A   B   C   D   E
         A 410   2   1   0   0
         B   0 268   0   0   2
         C   0   0  33   0   0
         D   0   0   0  82   0
         E   0   0   0   1 409

Overall Statistics

               Accuracy : 0.995
                 95% CI : (0.9892, 0.9982)
    No Information Rate : 0.3402
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.993

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            1.0000   0.9926  0.97059  0.98795   0.9951
Specificity            0.9962   0.9979  1.00000  1.00000   0.9987
Pos Pred Value         0.9927   0.9926  1.00000  1.00000   0.9976
Neg Pred Value         1.0000   0.9979  0.99915  0.99911   0.9975
Prevalence             0.3394   0.2235  0.02815  0.06871   0.3402
Detection Rate         0.3394   0.2219  0.02732  0.06788   0.3386
Detection Prevalence   0.3419   0.2235  0.02732  0.06788   0.3394
Balanced Accuracy      0.9981   0.9952  0.98529  0.99398   0.9969
[1] 0.9950331
```

- ## Random Forest:

```
        OOB estimate of  error rate: 0.36%
Confusion matrix:
    A   B  C   D   E class.error
A 954   1  0   0   0 0.001047120
B   6 625  0   0   0 0.009508716
C   0   3 75   0   0 0.038461538
D   0   0  0 193   0 0.000000000
E   0   0  0   0 959 0.000000000
```

```
Confusion Matrix and Statistics

          Reference
Prediction   A   B   C   D   E
         A 410   2   1   0   0
         B   0 268   0   0   2
         C   0   0  33   0   0
         D   0   0   0  82   0
         E   0   0   0   1 409

Overall Statistics

               Accuracy : 0.995
                 95% CI : (0.9892, 0.9982)
    No Information Rate : 0.3402
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.993

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            1.0000   0.9926  0.97059  0.98795   0.9951
Specificity            0.9962   0.9979  1.00000  1.00000   0.9987
Pos Pred Value         0.9927   0.9926  1.00000  1.00000   0.9976
Neg Pred Value         1.0000   0.9979  0.99915  0.99911   0.9975
Prevalence             0.3394   0.2235  0.02815  0.06871   0.3402
Detection Rate         0.3394   0.2219  0.02732  0.06788   0.3386
Detection Prevalence   0.3419   0.2235  0.02732  0.06788   0.3394
Balanced Accuracy      0.9981   0.9952  0.98529  0.99398   0.9969
```

- ## Decision Tree:



```
Confusion Matrix and Statistics

          Reference
Prediction   A   B   C   D   E
         A 417   0   0   7   5
         B   0 268   0   0   0
         C   0   0  30   0   0
         D   3   0   0  60   5
         E  12   0   0  13 387

Overall Statistics

               Accuracy : 0.9627
                 95% CI : (0.9504, 0.9727)
    No Information Rate : 0.3579
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9473

 Mcnemar's Test P-Value : NA

Statistics by Class:


                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9653    1.000  1.00000  0.75000   0.9748
Specificity            0.9845    1.000  1.00000  0.99290   0.9691
Pos Pred Value         0.9720    1.000  1.00000  0.88235   0.9393
Neg Pred Value         0.9807    1.000  1.00000  0.98244   0.9874
Prevalence             0.3579    0.222  0.02486  0.06628   0.3289
Detection Rate         0.3455    0.222  0.02486  0.04971   0.3206
Detection Prevalence   0.3554    0.222  0.02486  0.05634   0.3413
Balanced Accuracy      0.9749    1.000  1.00000  0.87145   0.9720
```

- SVM:

```
Confusion Matrix and Statistics


test_pred   0 0.25 0.5 0.75   1
     0    681    7   1    0   0
     0.25   1  443   0    0   0
     0.5    0    0  55    0   0
     0.75   0    0   0  137   0
     1      0    0   0    1 685

Overall Statistics

               Accuracy : 0.995
                 95% CI : (0.9909, 0.9976)
    No Information Rate : 0.3406
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.993

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 0.25 Class: 0.5 Class: 0.75 Class: 1
Sensitivity            0.9985      0.9844    0.98214     0.99275   1.0000
Specificity            0.9940      0.9994    1.00000     1.00000   0.9992
Pos Pred Value         0.9884      0.9977    1.00000     1.00000   0.9985
Neg Pred Value         0.9992      0.9955    0.99949     0.99947   1.0000
Prevalence             0.3391      0.2238    0.02785     0.06862   0.3406
Detection Rate         0.3386      0.2203    0.02735     0.06813   0.3406
Detection Prevalence   0.3426      0.2208    0.02735     0.06813   0.3411
Balanced Accuracy      0.9963      0.9919    0.99107     0.99638   0.9996
```

Discussions and Recommendations

KNN is a simple model which is effective at capturing complex interactions among variables without having to define a statistical model. However, it might be time consuming in case of a large training set as it takes a significant amount time to find distances to all the neighbours and then identify the nearest one. In our case, it took about 14.5 seconds to run the model and compute the result which was highest when compared to the other four models. The model is computationally expensive and requires high memory as the algorithm stores all training data.

Random Forest is an ensemble classifier consisting many decision trees, further providing the output by means of class's output by individual trees outputs of the class. The method couples the bagging idea and the random selection. Each tree is built up using a variant bootstrap data sample. In addition to this random forest changes it also constructs a classification of regression trees. In case of standard trees, the best node is split using its variables. Moreover, each node is divided using the best subset of predictors which are randomly chosen at that node. This method seems to be robust against over fitting. The main disadvantage of Random forests is their complexity. They are much harder and time-consuming to construct than decision trees. They also require more computational resources and are also less intuitive. In our case it took around 11.71 seconds for the model to run. This model gave us the highest accuracy of 99.5% and is the recommended model due to comparatively less time and high accuracy.

Decision Tree uses a decision tree as a predictive model to go from observations about an item to conclusions about the item's target value. Tree models where the target variable can take a discrete set of values called classification trees, these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision tree gave us a very high accuracy with good computation time but due to its complicated algorithm is usually a lengthy process. This model has been moderate with handling this dataset as it takes 14.36 seconds to run the model giving 96.3% accuracy.

SVM is the machine learning task for deducing functions from a labeled training data that can be occupied for both classification and regression. Support vector machines are a binary classification algorithm. Support vectors are the data points adjacent to the hyperplanes if the dataset is removed it will change the position of the dividing hyperplane. Advantages of model selection by means of both optimal number as well as the location of functions are obtained automatically during training. SVM is especially effective in cases where the number of dimensions is greater. Since we have many dimensions in our model as a predictor, SVM model also gave us the highest accuracy but took highest computation time of more than 25 second for our model to run.

Summary

Based on the dataset and the prediction of the classification of the type of error an individual performs while performing weightlifting, we select **Random Forest** as the ideal or best fit method for this model due to highest accuracy and lowest computation time. The KNN has the lowest accuracy 92.3% and the computation time being slightly higher as compared to others. There’s a minute difference between the accuracies for the different approaches, but we would recommend **Random Forest** with the **lowest computation time (11.71 seconds) and the highest accuracy (99.5%)**.

Appendix: R Code for use case study

Load the libraries

```
library(dplyr)
library(randomForest)
library(caTools)
library("class")
library("caret")
library(ggplot2)
library(Amelia)
library(rpart.plot)
library(ggcorrplot)
library(pkgsearch)
library(tidyverse)
library(dlstats)
library(ROCR)
library(pROC)
library(ggplot2)
library(dplyr)
library(ggcorrplot)
library(rpart)
library(data.table)
library(mltools)
library(psych)
library(ggplot2)
library(scatterplot3d)
library(plotrix)
library(openxlsx)
library(GPArotation)
library(car)
library(dplyr)
```

```
library(kernlab)
library(MASS)
library(lattice)
library(ggplot2)
library(caret)
library(rsq)
```

Initial Data Loading

```
data <- read.csv("C:/Users/karan/Desktop/DM.CSV",header=TRUE)
dim(data)
head(data)
```

Visualization

```
summary(data)
sapply(data,class)
any(is.na(data))
missmap(data)
ggplot(data = data, aes(classe,fill="row"))+ geom_bar()
ggplot(data = data, aes(roll_belt))+ geom_histogram(color="Blue",fill="Green")
```

PCA and Pre-processing

```
Z <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }
a<-as.data.frame(lapply(y[,1:53],Z))

ggplot(data = a, aes(gyros_forearm_z))+ geom_histogram(color="Blue",fill="Green")
ggplot(data = a, aes(roll_belt))+ geom_histogram(color="Blue",fill="Green")
ggcorrplot(cor(a))
wpn.df_train <- wpm_n[1:70,]
wpn.df_test <- wpm_n[71:100,]
wpm_train_labels <- wpm.df[1:70, 1]
wpm_test_labels <- wpm.df[71:100, 1]
pca <- prcomp(t(wpm_n),scale=TRUE)
```

```
pca$x[,c(1,2,3)]

plot(pca$x[,1], pca$x[,2])

pca.var <- pca$sdev^2

pca.var.per <- round(pca.var/sum(pca.var)*100,1)

barplot(pca.var.per,main="scree plot", xlab="principal component",ylab="percentage
variation")

pca.data <- data.frame(sample=(pca$x),

        X=pca$x[,1],

        Y=pca$x[,2],

        Z=pca$x[,3])



loading_scores<-pca$rotation[,1]

parameter_scores<- abs(loading_scores)

parameter_score_ranked <- sort(parameter_scores, decreasing = TRUE)

pc_var <- (pca$sdev^2)/sum(pca$sdev^2)

plot(pc_var, xlab = "Principal Component", ylab = "Proportion of Variance Explained", type
= "b")

plot(pca, main = "Principal Component Analysis")

wpm2.df <- read.csv("C:/Users/karan/Desktop/DM.csv",header = TRUE, stringsAsFactors =
FALSE)

wpm2.df

wpm2.df <- wpm2.df %>%

mutate(new_window = ifelse(new_window == "no",0,1))

wpm2.df[is.na(wpm2.df)] <- 0
```

KNN

```
df <- read.csv("C:/Users/karan/Desktop/DM2.csv",TRUE,",")

head(df)

start_time<-Sys.time()

ran <- sample(1:nrow(df), 0.9*nrow(df))

ran
```

```
nor <- function(x){
  (x- min(x))/ (max(x)- min(x))
}
nor_DM <- as.data.frame(lapply(df[,c(1:52)], nor))
summary(nor_DM)
df_train <- nor_DM[ran,]
df_test <- nor_DM[-ran,]
df_target_category <- df[ran,53]
df_test_category <- df[-ran,53]
pr <- knn(df_train,df_test,cl=df_target_category,k=63)
pr
tab <- table(pr,df_test_category)
tab
acc <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
acc(tab)
end_time<-Sys.time()
time.taken.knn <-end_time-start_time
time.taken.knn <- round(as.numeric(time.taken.knn),2)
print(time.taken.knn)
```

Random Forest

```
starttime<- Sys.time()
set.seed(2000)
sample = sample.split(data$classe, SplitRatio = .70)
train = subset(data, sample == TRUE)
test  = subset(data, sample == FALSE)
dim(train)
dim(test)
rf <- randomForest(classe~ yaw_belt+roll_belt
+pitch_belt+total_accel_belt+accel_belt_y+roll_arm+magnet_dumbbell_x+magnet_forearm_
y+magnet_forearm_z ,data=train)
rf
```

```
pred = predict(rf, newdata=test[-59], type = "class")
confusionMatrix(pred, test$classe)
endtime<- Sys.time()
mean(pred==test$classe)
cm = table(pred,test[,52])
cm
time.taken.rf <-endtime-starttime
time.taken.rf <- round(as.numeric(time.taken.knn),2)
print(time.taken.rf)
```

SVM

```
y <- read.csv("C:/Users/karan/Desktop/DM2.csv",TRUE,",")
y
str(y)
start_tym<-Sys.time()
y$classe <- as.numeric(y$classe)
y
str(y)
normalize <- function(x) {
  return((x-min(x))/(max(x) - min(x)))
}
z <- as.data.frame(lapply(y[,1:53], normalize))
z
summary(z)
str(z)
colnames(z)
z$classe <- as.factor(z$classe)
z$classe
set.seed(3033)
intrain <- createDataPartition(z$classe, p=0.5 , list = FALSE)
training <- z[intrain, ]
```

```
testing <- z[-intrain, ]

dim(training)

dim(testing)

trctrl <- trainControl('repeatedcv',number=10,repeats=3)

svm_linear <- train(classe~.,data= training,

                method =  "svmLinear",

                trControl = trctrl,

                tuneLength = "10")

svm_linear

test_pred <- predict(svm_linear , newdata = testing)

test_pred

confusionMatrix(table(test_pred ,testing$classe))

end_tym<-Sys.time()

time.taken.SVM <-end_tym-start_tym

time.taken.SVM <- round(as.numeric(time.taken.SVM),2)

print(time.taken.SVM)
```

Decision Tree

```
dt<-
read.csv(file="Example_WearableComputing_weight_lifting_exercises_biceps_curl_variatio
ns.csv", header=TRUE, sep=",")

starttym<-Sys.time()

str(dt)

dt_n<-nrow(dt)

dt_n_train<-round(0.7*dt_n)

set.seed(123)

train_indices <- sample(1:dt_n, dt_n_train)

train <- dt[train_indices, ]

test <- dt[-train_indices, ]

colnames(train)

dt_train_1h <- one_hot(as.data.table(train))

colnames(dt_train_1h)
```

```
df <- dplyr::select_if(dt_train_1h, is.numeric)

r <- cor(df, use="complete.obs")

round(r,2)

ggcorrplot(r)

model <- rpart(formula = classe ~ num_window+yaw_belt+roll_belt
+pitch_belt+total_accel_belt+accel_belt_y+roll_arm+magnet_dumbbell_x+magnet_forearm_
y+magnet_forearm_z,

              data = train,

              method = "class")

rpart.plot(model)

test$pred <- predict(object = model,

                     newdata = test,

                     type = "class")

confusionMatrix(data = test$pred,

               reference = test$classe)

endtym<-Sys.time()

time.taken.DT <-endtym-starttym

time.taken.DT <- round(as.numeric(time.taken.DT),2)

print(time.taken.DT)
```