Weight_Exercise_prediction

Reema Singla

29/12/2019

Executive Summary

Based on a dataset provide by HAR http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) we will try to train a predictive model to predict what exercise was performed using a dataset with 159 features

We'll take the following steps:

- Process the data, for use of this project
- Explore the data, especially focussing on the two paramaters we are interested in
- Model selection, where we try different models to help us answer our questions
- Model examination, to see wether our best model holds up to our standards
- A Conclusion where we answer the questions based on the data
- Predicting the classification of the model on test set

Processing

```
set.seed(111)
training_data = read.csv("pml-training.csv")
testing_data = read.csv("pml-testing.csv")
```

Exploratory data analyses

```
#Remove columns with more than 20% missing values
maxNAallowed = ceiling(nrow(training_data)/100 * 20)
removeColumns = which(colSums(is.na(training_data)| training_data=="")>maxNAallowed)
training_data_clean = training_data[,-c(1:7,removeColumns)]
testing_data_clean = testing_data[,-c(1:7,removeColumns)]

#remove time related columns
remove_time = grep("timestamp",names(training_data_clean))
training_without_time = training_data_clean[,-c(1,remove_time)]
testing_without_time = testing_data_clean[,-c(1,remove_time)]

#final data
train_data = training_without_time
testing_data = testing_without_time
```

Model selection

```
#split train data into test and train
partition <- createDataPartition(y=train_data$classe, p=0.8, list=FALSE)
train_sub_Train <- train_data[partition, ]
train_sub_Test <- train_data[-partition, ]

#Decision Tree
system.time(
  modelDT <- rpart(classe ~ ., method = "class", data = train_sub_Train)
)
```

```
##    user  system elapsed
##    5.34    0.00    5.48
```

```
predictDT <- predict(modelDT, train_sub_Test, type = "class")
cM <- confusionMatrix(predictDT, train_sub_Test$classe)
cM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 990 117  10  23  29
##          B  30 388  66  46 107
##          C  22  99 479  80 118
##          D  58 127 109 459  95
##          E  16  28  20  35 372
##
## Overall Statistics
##
##                Accuracy : 0.6852
##                  95% CI : (0.6704, 0.6997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6019
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8871   0.5112   0.7003   0.7138  0.51595
## Specificity            0.9362   0.9213   0.9015   0.8814  0.96908
## Pos Pred Value         0.8469   0.6091   0.6003   0.5413  0.78981
## Neg Pred Value         0.9542   0.8871   0.9344   0.9402  0.89890
## Prevalence             0.2845   0.1935   0.1744   0.1639  0.18379
## Detection Rate         0.2524   0.0989   0.1221   0.1170  0.09483
## Detection Prevalence   0.2980   0.1624   0.2034   0.2162  0.12006
## Balanced Accuracy      0.9117   0.7163   0.8009   0.7976  0.74252
```

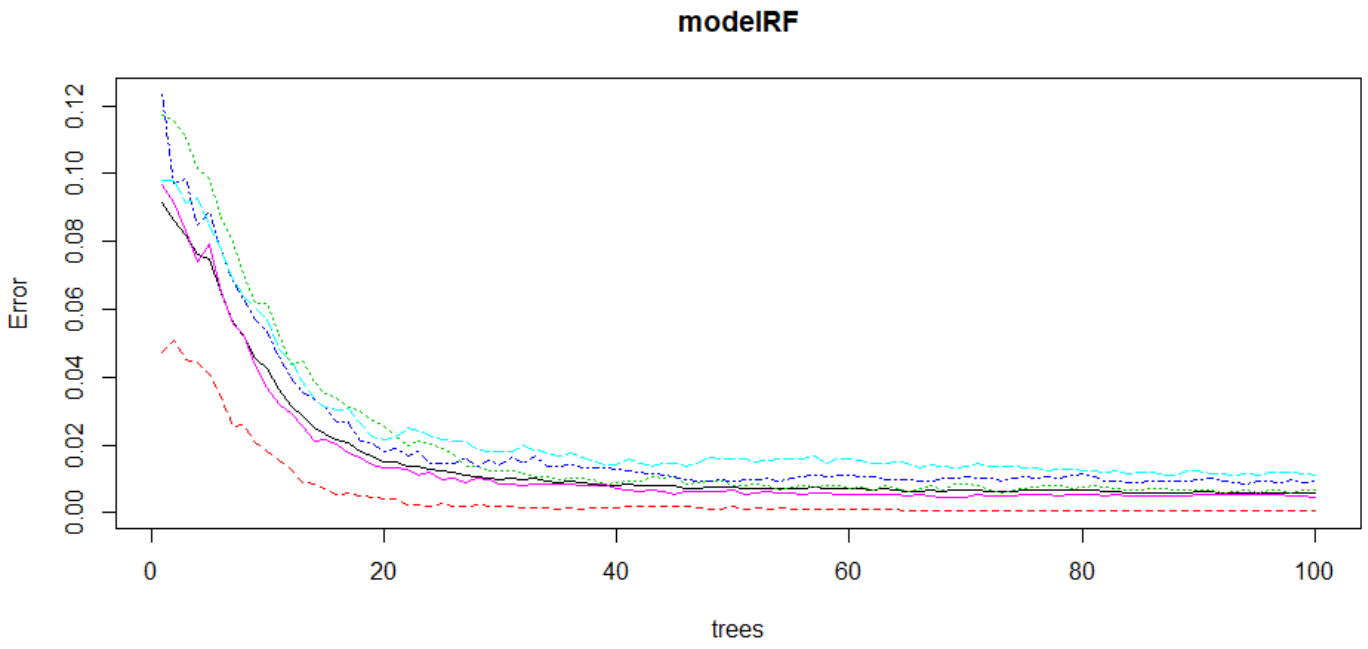
```
round(cM$overall["Accuracy"][[1]], 4) * 100
```

```
## [1] 68.52
```

```
#Random Forest
system.time(
  modelRF <- randomForest(classe ~ ., data = train_sub_Train, ntree = 100)
)
```

```
##    user  system elapsed
##   22.05    0.15   22.68
```

```
plot(modelRF)
```




```
predictRF <- predict(modelRF, train_sub_Test, type = "class")
cM <- confusionMatrix(predictRF, train_sub_Test$classe)
cM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    5    0    0    0
##          B    0  754    3    0    0
##          C    0    0  681   10    1
##          D    0    0    0  633    1
##          E    0    0    0    0  719
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9921, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9934   0.9956   0.9844   0.9972
## Specificity            0.9982   0.9991   0.9966   0.9997   1.0000
## Pos Pred Value         0.9955   0.9960   0.9841   0.9984   1.0000
## Neg Pred Value         1.0000   0.9984   0.9991   0.9970   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1922   0.1736   0.1614   0.1833
## Detection Prevalence   0.2858   0.1930   0.1764   0.1616   0.1833
## Balanced Accuracy      0.9991   0.9962   0.9961   0.9921   0.9986
```

```
round(cM$overall["Accuracy"][[1]], 4) * 100
```

```
## [1] 99.49
```

```
#GBM
system.time(
  modelGBM <- train(classe ~ ., method = "gbm", data = train_sub_Train,
                    trControl = trainControl(method = "repeatedcv", number = 5, repeats = 1))
)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1155
##      2        1.5371             nan     0.1000    0.0767
##      3        1.4874             nan     0.1000    0.0608
##      4        1.4493             nan     0.1000    0.0518
##      5        1.4168             nan     0.1000    0.0389
##      6        1.3923             nan     0.1000    0.0425
##      7        1.3655             nan     0.1000    0.0369
##      8        1.3410             nan     0.1000    0.0341
##      9        1.3182             nan     0.1000    0.0326
##     10        1.2979             nan     0.1000    0.0263
##     20        1.1508             nan     0.1000    0.0169
##     40        0.9747             nan     0.1000    0.0105
##     60        0.8637             nan     0.1000    0.0065
##     80        0.7798             nan     0.1000    0.0038
##    100        0.7133             nan     0.1000    0.0037
##    120        0.6593             nan     0.1000    0.0036
##    140        0.6135             nan     0.1000    0.0023
##    150        0.5937             nan     0.1000    0.0024
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1720
##      2        1.5046             nan     0.1000    0.1241
##      3        1.4266             nan     0.1000    0.0953
##      4        1.3678             nan     0.1000    0.0732
##      5        1.3206             nan     0.1000    0.0675
##      6        1.2771             nan     0.1000    0.0730
##      7        1.2315             nan     0.1000    0.0582
##      8        1.1950             nan     0.1000    0.0523
##      9        1.1626             nan     0.1000    0.0463
##     10        1.1341             nan     0.1000    0.0381
##     20        0.9434             nan     0.1000    0.0195
##     40        0.7059             nan     0.1000    0.0155
##     60        0.5651             nan     0.1000    0.0078
##     80        0.4783             nan     0.1000    0.0055
##    100        0.4144             nan     0.1000    0.0047
##    120        0.3568             nan     0.1000    0.0030
##    140        0.3157             nan     0.1000    0.0019
##    150        0.2981             nan     0.1000    0.0018
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2166
##      2        1.4724             nan     0.1000    0.1527
##      3        1.3790             nan     0.1000    0.1165
##      4        1.3063             nan     0.1000    0.0937
##      5        1.2467             nan     0.1000    0.0891
##      6        1.1915             nan     0.1000    0.0718
##      7        1.1461             nan     0.1000    0.0777
##      8        1.0994             nan     0.1000    0.0581
##      9        1.0633             nan     0.1000    0.0568
##     10        1.0277             nan     0.1000    0.0519
##     20        0.8029             nan     0.1000    0.0242
##     40        0.5531             nan     0.1000    0.0165
```

```
##     60        0.4151             nan     0.1000    0.0075
##     80        0.3306             nan     0.1000    0.0051
##    100        0.2742             nan     0.1000    0.0038
##    120        0.2327             nan     0.1000    0.0024
##    140        0.2001             nan     0.1000    0.0019
##    150        0.1857             nan     0.1000    0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1163
##      2        1.5376             nan     0.1000    0.0748
##      3        1.4903             nan     0.1000    0.0609
##      4        1.4521             nan     0.1000    0.0467
##      5        1.4230             nan     0.1000    0.0477
##      6        1.3933             nan     0.1000    0.0380
##      7        1.3685             nan     0.1000    0.0392
##      8        1.3447             nan     0.1000    0.0297
##      9        1.3247             nan     0.1000    0.0298
##     10        1.3058             nan     0.1000    0.0296
##     20        1.1567             nan     0.1000    0.0161
##     40        0.9821             nan     0.1000    0.0092
##     60        0.8668             nan     0.1000    0.0057
##     80        0.7810             nan     0.1000    0.0046
##    100        0.7148             nan     0.1000    0.0035
##    120        0.6614             nan     0.1000    0.0025
##    140        0.6160             nan     0.1000    0.0017
##    150        0.5959             nan     0.1000    0.0024
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1679
##      2        1.5038             nan     0.1000    0.1129
##      3        1.4337             nan     0.1000    0.0980
##      4        1.3734             nan     0.1000    0.0830
##      5        1.3218             nan     0.1000    0.0650
##      6        1.2812             nan     0.1000    0.0654
##      7        1.2409             nan     0.1000    0.0592
##      8        1.2042             nan     0.1000    0.0523
##      9        1.1716             nan     0.1000    0.0468
##     10        1.1420             nan     0.1000    0.0408
##     20        0.9354             nan     0.1000    0.0194
##     40        0.7083             nan     0.1000    0.0086
##     60        0.5733             nan     0.1000    0.0076
##     80        0.4847             nan     0.1000    0.0043
##    100        0.4151             nan     0.1000    0.0041
##    120        0.3605             nan     0.1000    0.0033
##    140        0.3171             nan     0.1000    0.0038
##    150        0.2985             nan     0.1000    0.0016
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2175
##      2        1.4747             nan     0.1000    0.1441
##      3        1.3853             nan     0.1000    0.1203
##      4        1.3104             nan     0.1000    0.1025
##      5        1.2463             nan     0.1000    0.0818
##      6        1.1957             nan     0.1000    0.0740
```

```
##      7        1.1504             nan     0.1000    0.0698
##      8        1.1072             nan     0.1000    0.0686
##      9        1.0658             nan     0.1000    0.0435
##     10        1.0374             nan     0.1000    0.0538
##     20        0.7928             nan     0.1000    0.0317
##     40        0.5521             nan     0.1000    0.0114
##     60        0.4127             nan     0.1000    0.0063
##     80        0.3288             nan     0.1000    0.0034
##    100        0.2723             nan     0.1000    0.0033
##    120        0.2277             nan     0.1000    0.0024
##    140        0.1929             nan     0.1000    0.0016
##    150        0.1790             nan     0.1000    0.0020
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1176
##      2        1.5363             nan     0.1000    0.0791
##      3        1.4869             nan     0.1000    0.0627
##      4        1.4477             nan     0.1000    0.0501
##      5        1.4163             nan     0.1000    0.0447
##      6        1.3875             nan     0.1000    0.0403
##      7        1.3613             nan     0.1000    0.0371
##      8        1.3377             nan     0.1000    0.0320
##      9        1.3164             nan     0.1000    0.0333
##     10        1.2947             nan     0.1000    0.0332
##     20        1.1464             nan     0.1000    0.0151
##     40        0.9719             nan     0.1000    0.0101
##     60        0.8583             nan     0.1000    0.0072
##     80        0.7745             nan     0.1000    0.0043
##    100        0.7080             nan     0.1000    0.0037
##    120        0.6533             nan     0.1000    0.0027
##    140        0.6074             nan     0.1000    0.0025
##    150        0.5877             nan     0.1000    0.0023
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1754
##      2        1.5015             nan     0.1000    0.1142
##      3        1.4282             nan     0.1000    0.0995
##      4        1.3663             nan     0.1000    0.0803
##      5        1.3166             nan     0.1000    0.0635
##      6        1.2765             nan     0.1000    0.0727
##      7        1.2314             nan     0.1000    0.0621
##      8        1.1924             nan     0.1000    0.0503
##      9        1.1606             nan     0.1000    0.0469
##     10        1.1306             nan     0.1000    0.0392
##     20        0.9317             nan     0.1000    0.0258
##     40        0.7113             nan     0.1000    0.0123
##     60        0.5690             nan     0.1000    0.0073
##     80        0.4789             nan     0.1000    0.0039
##    100        0.4103             nan     0.1000    0.0048
##    120        0.3581             nan     0.1000    0.0023
##    140        0.3159             nan     0.1000    0.0017
##    150        0.2957             nan     0.1000    0.0025
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.6094             nan     0.1000    0.2201
##      2        1.4721             nan     0.1000    0.1502
##      3        1.3797             nan     0.1000    0.1283
##      4        1.3006             nan     0.1000    0.1016
##      5        1.2372             nan     0.1000    0.0840
##      6        1.1851             nan     0.1000    0.0752
##      7        1.1399             nan     0.1000    0.0751
##      8        1.0941             nan     0.1000    0.0600
##      9        1.0573             nan     0.1000    0.0492
##     10        1.0269             nan     0.1000    0.0459
##     20        0.7941             nan     0.1000    0.0228
##     40        0.5473             nan     0.1000    0.0153
##     60        0.4179             nan     0.1000    0.0066
##     80        0.3338             nan     0.1000    0.0062
##    100        0.2772             nan     0.1000    0.0026
##    120        0.2338             nan     0.1000    0.0019
##    140        0.2002             nan     0.1000    0.0024
##    150        0.1854             nan     0.1000    0.0006
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1141
##      2        1.5368             nan     0.1000    0.0767
##      3        1.4869             nan     0.1000    0.0587
##      4        1.4493             nan     0.1000    0.0475
##      5        1.4189             nan     0.1000    0.0436
##      6        1.3908             nan     0.1000    0.0418
##      7        1.3644             nan     0.1000    0.0411
##      8        1.3392             nan     0.1000    0.0321
##      9        1.3188             nan     0.1000    0.0287
##     10        1.2994             nan     0.1000    0.0307
##     20        1.1530             nan     0.1000    0.0171
##     40        0.9770             nan     0.1000    0.0108
##     60        0.8650             nan     0.1000    0.0077
##     80        0.7759             nan     0.1000    0.0034
##    100        0.7138             nan     0.1000    0.0038
##    120        0.6589             nan     0.1000    0.0033
##    140        0.6152             nan     0.1000    0.0044
##    150        0.5946             nan     0.1000    0.0020
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1728
##      2        1.5032             nan     0.1000    0.1185
##      3        1.4280             nan     0.1000    0.0991
##      4        1.3658             nan     0.1000    0.0798
##      5        1.3170             nan     0.1000    0.0656
##      6        1.2760             nan     0.1000    0.0595
##      7        1.2380             nan     0.1000    0.0573
##      8        1.2029             nan     0.1000    0.0496
##      9        1.1720             nan     0.1000    0.0555
##     10        1.1384             nan     0.1000    0.0431
##     20        0.9315             nan     0.1000    0.0219
##     40        0.7029             nan     0.1000    0.0110
##     60        0.5763             nan     0.1000    0.0080
##     80        0.4824             nan     0.1000    0.0057
```

```
##    100        0.4096             nan     0.1000    0.0028
##    120        0.3581             nan     0.1000    0.0029
##    140        0.3177             nan     0.1000    0.0028
##    150        0.2988             nan     0.1000    0.0017
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2167
##      2        1.4744             nan     0.1000    0.1489
##      3        1.3806             nan     0.1000    0.1199
##      4        1.3060             nan     0.1000    0.1002
##      5        1.2448             nan     0.1000    0.0824
##      6        1.1927             nan     0.1000    0.0802
##      7        1.1437             nan     0.1000    0.0643
##      8        1.1030             nan     0.1000    0.0546
##      9        1.0678             nan     0.1000    0.0489
##     10        1.0365             nan     0.1000    0.0496
##     20        0.7997             nan     0.1000    0.0211
##     40        0.5520             nan     0.1000    0.0101
##     60        0.4207             nan     0.1000    0.0052
##     80        0.3351             nan     0.1000    0.0044
##    100        0.2748             nan     0.1000    0.0032
##    120        0.2325             nan     0.1000    0.0033
##    140        0.1994             nan     0.1000    0.0015
##    150        0.1859             nan     0.1000    0.0020
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1146
##      2        1.5364             nan     0.1000    0.0790
##      3        1.4867             nan     0.1000    0.0628
##      4        1.4472             nan     0.1000    0.0496
##      5        1.4158             nan     0.1000    0.0432
##      6        1.3895             nan     0.1000    0.0418
##      7        1.3631             nan     0.1000    0.0394
##      8        1.3385             nan     0.1000    0.0310
##      9        1.3189             nan     0.1000    0.0314
##     10        1.2987             nan     0.1000    0.0304
##     20        1.1504             nan     0.1000    0.0205
##     40        0.9755             nan     0.1000    0.0092
##     60        0.8646             nan     0.1000    0.0067
##     80        0.7784             nan     0.1000    0.0058
##    100        0.7127             nan     0.1000    0.0040
##    120        0.6579             nan     0.1000    0.0032
##    140        0.6134             nan     0.1000    0.0033
##    150        0.5935             nan     0.1000    0.0030
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1694
##      2        1.5031             nan     0.1000    0.1197
##      3        1.4272             nan     0.1000    0.0958
##      4        1.3665             nan     0.1000    0.0822
##      5        1.3157             nan     0.1000    0.0692
##      6        1.2732             nan     0.1000    0.0595
##      7        1.2361             nan     0.1000    0.0488
##      8        1.2052             nan     0.1000    0.0453
```

```
##      9        1.1755             nan     0.1000    0.0475
##     10        1.1448             nan     0.1000    0.0428
##     20        0.9415             nan     0.1000    0.0223
##     40        0.7049             nan     0.1000    0.0081
##     60        0.5687             nan     0.1000    0.0067
##     80        0.4783             nan     0.1000    0.0052
##    100        0.4100             nan     0.1000    0.0034
##    120        0.3584             nan     0.1000    0.0037
##    140        0.3149             nan     0.1000    0.0015
##    150        0.2964             nan     0.1000    0.0020
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2147
##      2        1.4747             nan     0.1000    0.1493
##      3        1.3818             nan     0.1000    0.1304
##      4        1.3031             nan     0.1000    0.0950
##      5        1.2426             nan     0.1000    0.0897
##      6        1.1873             nan     0.1000    0.0813
##      7        1.1387             nan     0.1000    0.0611
##      8        1.1002             nan     0.1000    0.0702
##      9        1.0583             nan     0.1000    0.0528
##     10        1.0250             nan     0.1000    0.0531
##     20        0.7882             nan     0.1000    0.0315
##     40        0.5489             nan     0.1000    0.0097
##     60        0.4177             nan     0.1000    0.0083
##     80        0.3346             nan     0.1000    0.0043
##    100        0.2757             nan     0.1000    0.0030
##    120        0.2306             nan     0.1000    0.0031
##    140        0.1975             nan     0.1000    0.0016
##    150        0.1828             nan     0.1000    0.0014
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.2132
##      2        1.4775             nan     0.1000    0.1544
##      3        1.3828             nan     0.1000    0.1090
##      4        1.3149             nan     0.1000    0.1104
##      5        1.2484             nan     0.1000    0.0759
##      6        1.2001             nan     0.1000    0.0807
##      7        1.1510             nan     0.1000    0.0789
##      8        1.1045             nan     0.1000    0.0582
##      9        1.0677             nan     0.1000    0.0495
##     10        1.0366             nan     0.1000    0.0479
##     20        0.7962             nan     0.1000    0.0264
##     40        0.5474             nan     0.1000    0.0101
##     60        0.4202             nan     0.1000    0.0070
##     80        0.3363             nan     0.1000    0.0056
##    100        0.2783             nan     0.1000    0.0032
##    120        0.2343             nan     0.1000    0.0025
##    140        0.2012             nan     0.1000    0.0009
##    150        0.1878             nan     0.1000    0.0021
```

```
##    user  system elapsed
##  730.75    2.37  746.42
```

```
predictGBM <- predict(modelGBM, train_sub_Test)
cM <- confusionMatrix(predictGBM, train_sub_Test$classe)
cM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1099   28    0    2    2
##          B   14  701   22    5   10
##          C    1   30  653   27    8
##          D    1    0    8  601    6
##          E    1    0    1    8  695
##
## Overall Statistics
##
##                Accuracy : 0.9556
##                  95% CI : (0.9487, 0.9619)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9439
##
##  Mcnemar's Test P-Value : 2.972e-05
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9848   0.9236   0.9547   0.9347   0.9639
## Specificity            0.9886   0.9839   0.9796   0.9954   0.9969
## Pos Pred Value         0.9717   0.9322   0.9082   0.9756   0.9858
## Neg Pred Value         0.9939   0.9817   0.9903   0.9873   0.9919
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2801   0.1787   0.1665   0.1532   0.1772
## Detection Prevalence   0.2883   0.1917   0.1833   0.1570   0.1797
## Balanced Accuracy      0.9867   0.9537   0.9672   0.9651   0.9804
```

```
round(cM$overall["Accuracy"][[1]], 4) * 100
```

```
## [1] 95.56
```

```
#LDA
system.time(
  modelLDA <- train(classe ~ ., method = "lda", data = train_sub_Train,
                    trControl = trainControl(method = "repeatedcv", number = 5, repeats = 1))
)
```

```
##    user  system elapsed
##    5.02    0.19    5.23
```

```
predictLDA <- predict(modelLDA, train_sub_Test)
cM <- confusionMatrix(predictLDA, train_sub_Test$classe)
cM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 908 112  72  32  25
##          B  28 491  51  31 133
##          C 101  92 461  87  74
##          D  77  23  84 447  91
##          E   2  41  16  46 398
##
## Overall Statistics
##
##                Accuracy : 0.6895
##                  95% CI : (0.6748, 0.704)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6073
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8136   0.6469   0.6740   0.6952   0.5520
## Specificity            0.9141   0.9232   0.8907   0.9162   0.9672
## Pos Pred Value         0.7903   0.6689   0.5656   0.6191   0.7913
## Neg Pred Value         0.9250   0.9160   0.9282   0.9388   0.9056
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2315   0.1252   0.1175   0.1139   0.1015
## Detection Prevalence   0.2929   0.1871   0.2077   0.1840   0.1282
## Balanced Accuracy      0.8639   0.7851   0.7823   0.8057   0.7596
```

```
round(cM$overall["Accuracy"][[1]], 4) * 100
```

```
## [1] 68.95
```

```
#Neural Networks
system.time(
  modelDL <- nnet(classe ~ ., train_sub_Train, size = 5, rang = .1, decay = 5e-4, maxit = 100,
                  trControl = trainControl(method = "repeatedcv", number = 5, repeats = 1))
)
```

```
## # weights:  290
## initial  value 25344.736575
## iter  10 value 23765.751311
## iter  20 value 23495.771752
## iter  30 value 22861.470413
## iter  40 value 22336.941867
## iter  50 value 22025.974366
## iter  60 value 21671.166621
## iter  70 value 21225.465381
## iter  80 value 21139.356477
## iter  90 value 20913.854464
## iter 100 value 20649.707306
## final  value 20649.707306
## stopped after 100 iterations
```

```
##    user  system elapsed
##   22.28    0.00   22.59
```

```
predictDL <- predict(modelDL, train_sub_Test, type = "class")
cM <- confusionMatrix(as.factor(predictDL), train_sub_Test$classe)
cM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 710  66  76  23  32
##          B  23 116  30  11  79
##          C 206 216 514 216 244
##          D 177 330  56 393 318
##          E   0  31   8   0  48
##
## Overall Statistics
##
##                Accuracy : 0.454
##                  95% CI : (0.4383, 0.4697)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3193
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6362  0.15283   0.7515   0.6112  0.06657
## Specificity            0.9298  0.95480   0.7277   0.7314  0.98782
## Pos Pred Value         0.7828  0.44788   0.3682   0.3085  0.55172
## Neg Pred Value         0.8654  0.82451   0.9327   0.9056  0.82456
## Prevalence             0.2845  0.19347   0.1744   0.1639  0.18379
## Detection Rate         0.1810  0.02957   0.1310   0.1002  0.01224
## Detection Prevalence   0.2312  0.06602   0.3559   0.3248  0.02218
## Balanced Accuracy      0.7830  0.55382   0.7396   0.6713  0.52720
```

```
round(cM$overall["Accuracy"][[1]], 4) * 100
```

```
## [1] 45.4
```

#using Random Forest because of best accuracy

```
predict(modelRF, testing_data, type = "class")
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```