# Title: “Machine Learning Project”

# author: “Vivek Pawar”

# date: “October 20, 2018”

# output: html_document

The purpose of this project is to predict how well the dumbell biceps curl exercise was done that was monitired by various sensors. There are five classes of how well the exercise was done. Class A corresponds to the specified execution of the exercise whereas the the other 4 (B to E) correspond to common mistakes.

*Tidyig the data*

The first step is to read the data set that is provided as training dataset and testing dataset. The training dataset has 19622 observations and 160 variables. The testing dataset has 20 observations with 160 variables. View of data in R studio shows that there are several variables that are NAs or missing values. In addition there are time stamps and usernames that are not useful either.

NOTE: The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

```
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
require(ggplot2)
require(rattle)
```

```
## Loading required package: rattle
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
require(tidyverse)
```

```
## Loading required package: tidyverse
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  1.4.2     v purrr   0.2.5
## v tidyr   0.8.1     v dplyr   0.7.7
## v readr   1.1.1     v stringr 1.3.1
## v tibble  1.4.2     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
require(klar)
```

```
## Loading required package: klar
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'klar'
```

```
trainingall <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.cs
v"))
testingall <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
))
traindataclean <- trainingall[-c(1:7,12:36, 50:59, 69:83,87:101, 103:112, 125:139,141:150)]
testingdataclean <- testingall[-c(1:7,12:36, 50:59, 69:83,87:101, 103:112, 125:139,141:150)]
dim(traindataclean)
```

```
## [1] 19622    53
```

```
dim(testingdataclean)
```

```
## [1] 20 53
```

So training data has 19,622 observations with 53 varaibles and testing data has 20 observations with 53 vairables. Now lets partition the training data set and use portion of that to train and test our model. So now our training dataset has 14718 variables whereas testing dataset has 4904 observations.

```
inTrain <- createDataPartition(traindataclean$classe, p=0.75, list=FALSE)
traindataclean1 <- traindataclean[inTrain,]
trainvaldataclean <- traindataclean[-inTrain,]
dim(traindataclean)
```

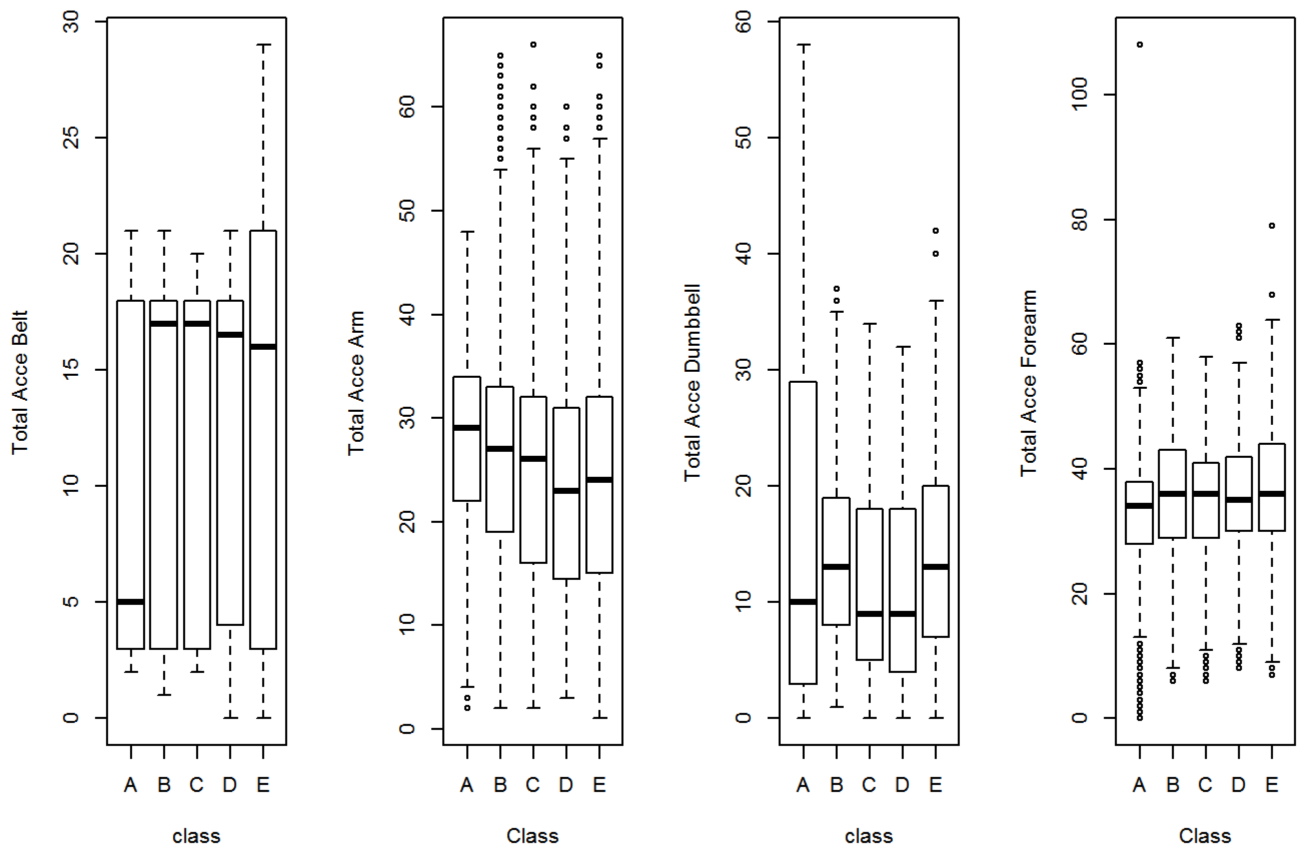
```
## [1] 19622    53
```

```
dim(trainvaldataclean)
```

```
## [1] 4904   53
```

Since there are 53 variables, it will be difficult to look at all of them in a convinient manner. But lets look at some of the key variables such as total_acceleration for belt, arm, dumbell and forarm. The total_acce_belt seems to be different in Class A (correct way to exercise) compared to the other classes (B to E). The other total acceelrations do not show any such diference.

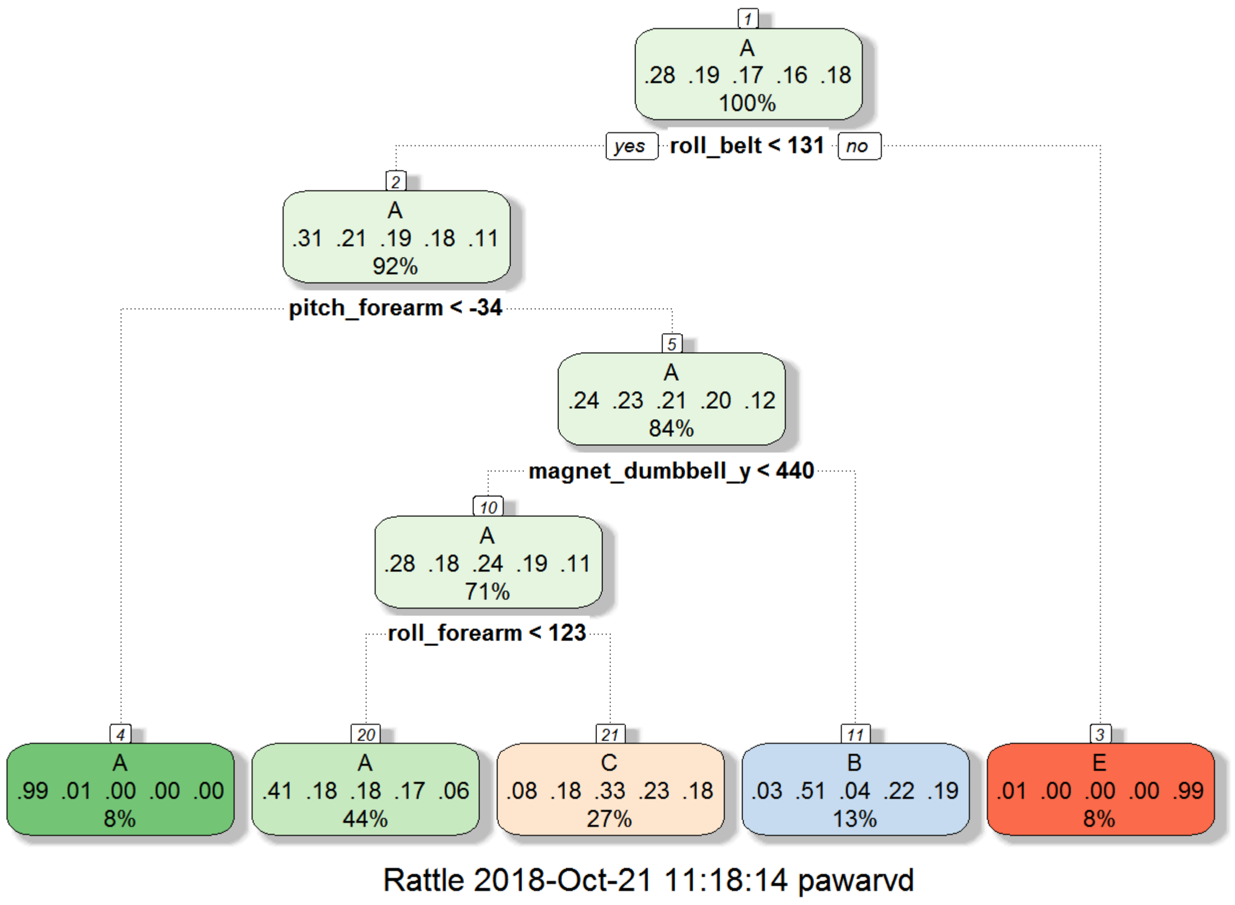
```
par(mfrow=c(1,4))
plot(x=traindataclean1$classe, y=traindataclean1$total_accel_belt, xlab="class", ylab="Total Acc
e Belt")
plot(x=traindataclean1$classe, y=traindataclean1$total_accel_arm, xlab="Class", ylab="Total Acce
 Arm")
plot(x=traindataclean1$classe, y=traindataclean1$total_accel_dumbbell, xlab="class", ylab="Total
 Acce Dumbbell")
plot(x=traindataclean1$classe, y=traindataclean1$total_accel_forearm, xlab="Class", ylab="Total
 Acce Forearm")
```



*Modeling of the data*

As a first approach lets use classification tree method to classify the data and evaluate the model's accuracy on the subset of the data. The overall accuracy is low 0.49.

```
set.seed(1234)
trControl <- trainControl(method="cv", number=5)
rpartmodelfit <- train(classe~., data=traindataclean1, method="rpart", trControl=trControl)
fancyRpartPlot(rpartmodelfit$finalModel)
```



```
predmod <- predict(rpartmodelfit, newdata=trainvaldataclean)
rpartconfM <- confusionMatrix(trainvaldataclean$classe,predmod)
rpartconfM
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1262   18  111    0    4
##          B  378  341  230    0    0
##          C  391   29  435    0    0
##          D  345  160  299    0    0
##          E  123  126  239    0  413
## 
## Overall Statistics
## 
##                Accuracy : 0.4998          
##                  95% CI : (0.4857, 0.5139)
##     No Information Rate : 0.5096          
##     P-Value [Acc > NIR] : 0.917           
## 
##                   Kappa : 0.3471          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5050  0.50593   0.3311       NA  0.99041
## Specificity            0.9447  0.85626   0.8830   0.8361  0.89124
## Pos Pred Value         0.9047  0.35933   0.5088       NA  0.45838
## Neg Pred Value         0.6475  0.91580   0.7829       NA  0.99900
## Prevalence             0.5096  0.13744   0.2679   0.0000  0.08503
## Detection Rate         0.2573  0.06954   0.0887   0.0000  0.08422
## Detection Prevalence   0.2845  0.19352   0.1743   0.1639  0.18373
## Balanced Accuracy      0.7249  0.68110   0.6070       NA  0.94082
```

Now lets use randomforest method to classify the data and evaluate the model’s accuracy on the subset of the data. The accuracy of the random forest method is 0.99 with 5 fold validation.

```
set.seed(1288)
trControl <- trainControl(method="cv", number=5)
rfmodelfit <- train(classe~., data=traindataclean1, method="rf", trControl=trControl)
predmod1 <- predict(rfmodelfit, newdata=trainvaldataclean)
rfconfM <- confusionMatrix(trainvaldataclean$classe, predmod1)
rfconfM
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    4  943    2    0    0
##          C    0    6  849    0    0
##          D    0    0   11  793    0
##          E    0    0    2    3  896
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9943          
##                  95% CI : (0.9918, 0.9962)
##     No Information Rate : 0.2853          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9928          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9937   0.9826   0.9962   1.0000
## Specificity            1.0000   0.9985   0.9985   0.9973   0.9988
## Pos Pred Value         1.0000   0.9937   0.9930   0.9863   0.9945
## Neg Pred Value         0.9989   0.9985   0.9963   0.9993   1.0000
## Prevalence             0.2853   0.1935   0.1762   0.1623   0.1827
## Detection Rate         0.2845   0.1923   0.1731   0.1617   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9986   0.9961   0.9906   0.9968   0.9994
```

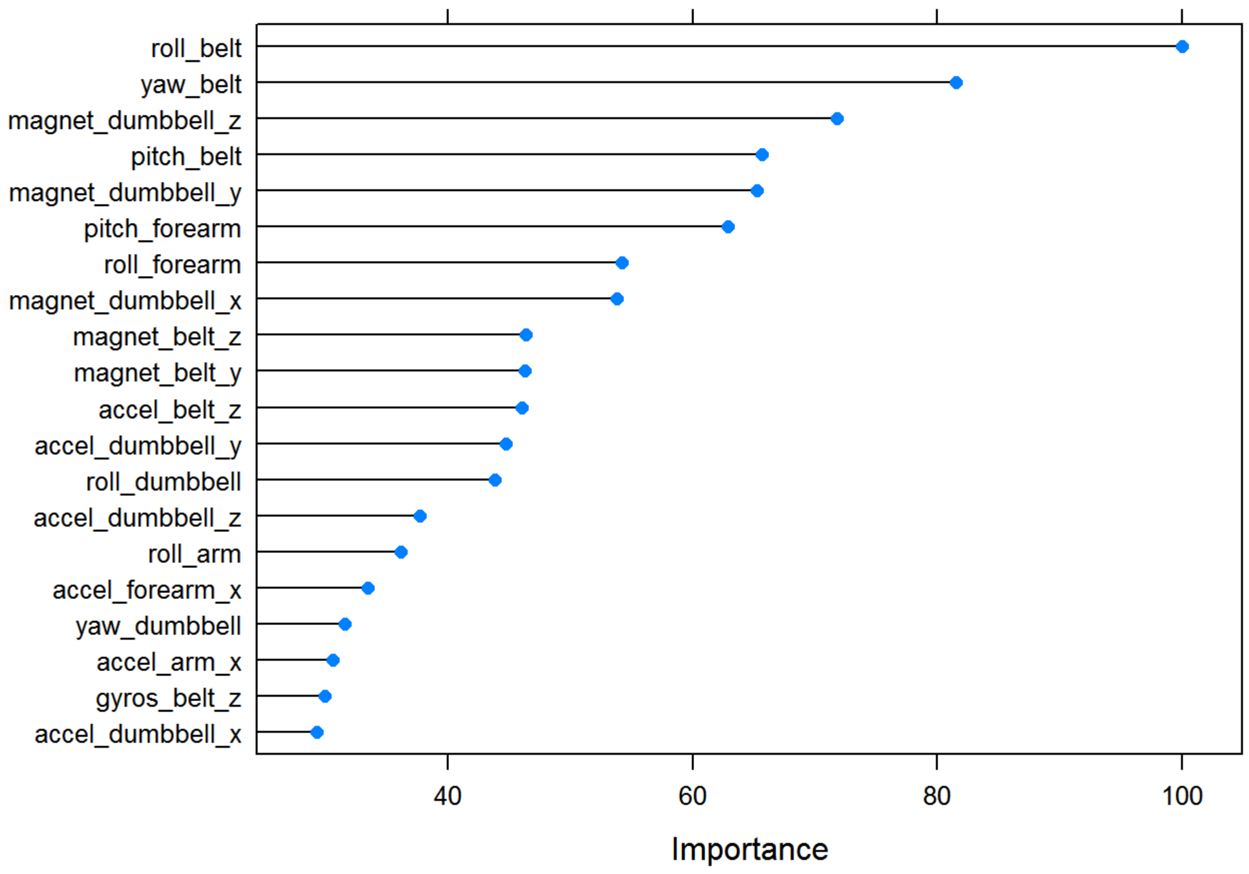
Now lets try gradient boosting method. The accuracy of this is method is also 0.95

```
set.seed(1288)
trControl <- trainControl(method="cv", number=5)
gbmodelfit <- train(classe~., data=traindataclean1, method="gbm", trControl=trControl,verbose=FA
LSE)
predmod2 <- predict(gbmodelfit, newdata=trainvaldataclean)
gbconfM <- confusionMatrix(trainvaldataclean$classe, predmod2)
gbconfM
```

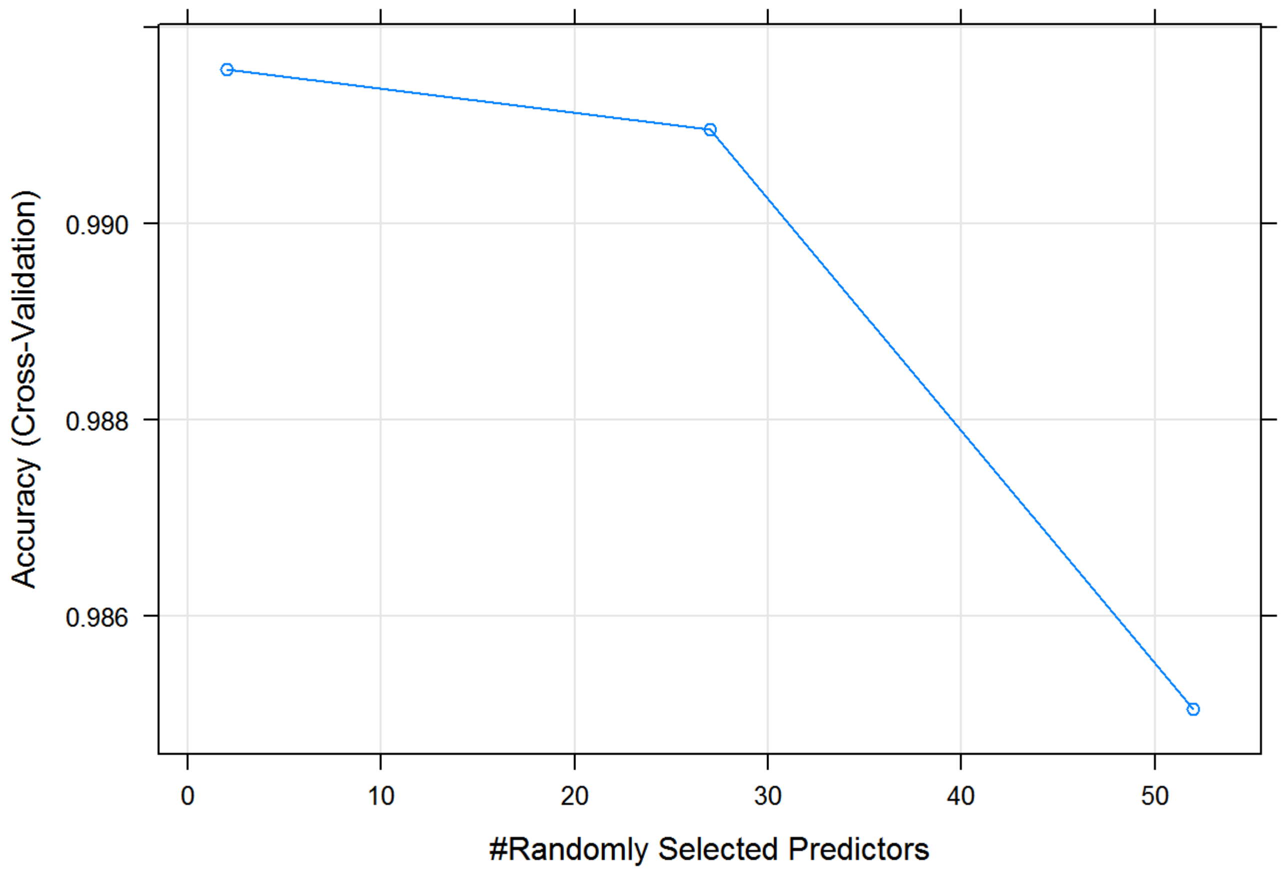
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1374   17    3    1    0
##          B   33  895   19    1    1
##          C    0   36  814    4    1
##          D    0    2   27  772    3
##          E    0   10    5   12  874
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9643          
##                  95% CI : (0.9587, 0.9693)
##     No Information Rate : 0.2869          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9548          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9765   0.9323   0.9378   0.9772   0.9943
## Specificity            0.9940   0.9863   0.9898   0.9922   0.9933
## Pos Pred Value         0.9849   0.9431   0.9520   0.9602   0.9700
## Neg Pred Value         0.9906   0.9836   0.9867   0.9956   0.9988
## Prevalence             0.2869   0.1958   0.1770   0.1611   0.1792
## Detection Rate         0.2802   0.1825   0.1660   0.1574   0.1782
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9853   0.9593   0.9638   0.9847   0.9938
```

Figure below shows the relative importance of top 20 variables in random forest fit and plot of accuracy vs number of predictors. Plot of top two variables is also shown

```
plot(varImp(rfmodelfit), top=20)
```



```
plot(rfmodelfit)
```



*Conclusion* The random forest method gave the maximum accuracy (0.9933). Using that model for the validation/test data set of 20 observation gave following output.

```
predmod3 <- predict(rfmodelfit, testingdataclean)
predmod3
```

```
##  [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
```