**# Perform the below given activities:**

**# a. Create classification model using different random forest models**

**# b. Verify model goodness of fit**

**# c. Apply all the model validation techniques**

**# d. Make conclusions**

**# e. Plot importance of variables**

**# import data set**

data\_set <- read.csv("E:/Data Analytics with RET/Assignment/Dataset/Example\_WearableComputing\_weight\_lifting\_exercises\_biceps\_curl\_variations.csv")

View(data\_set)

**# remove irrelevant collumns viz. name, cvtd\_timestamp, new\_window**

data <- data\_set[,-c(1,4,5)]

View(data)

str(data)

sum(is.na(data)) # there are no missing values

**# spliting the data set for train and test**

library(caTools)

set.seed(123)

split = sample.split(data$classe, SplitRatio = 0.7)

train = subset(data, split == TRUE) # train data

test = subset(data, split == FALSE) # test data

dim(train)

dim(test)

**# a. Create classification model using different random forest models**

library(tree); library(rpart); library(caret); library(C50); library(randomForest)

library(adabag); library(gbm)

**# 1**

train\_control <- trainControl(method = "cv", number = 10)

cvmodel1 <- train(classe ~ ., data = train, trControl = train\_control, method = "rf")

cvpred1 <- predict(cvmodel1, test) # make prediction

cvconf1 <- confusionMatrix(test$classe, cvpred1) # confusion matrix

cvconf1$overall[1] # accuracy

**# default**

set.seed(123)

train\_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

rf\_default <- train(classe ~ ., data = train, trControl = train\_control, method = "rf",

metric = 'Accuracy', tuneGrid = expand.grid(.mtry = sqrt(ncol(train))))

pred\_rf\_default <- predict(rf\_default, test) # make prediction

conf\_rf\_default <- confusionMatrix(test$classe, pred\_rf\_default) # confusion matrix

conf\_rf\_default$overall[1] # accuracy

varImp(rf\_default) # var importance - 20

**# random search for parameters**

train\_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = 'random')

rf\_random <- train(classe ~ ., data = train, trControl = train\_control, method = "rf",

metric = 'Accuracy', tuneLength = 15)

pred\_rf\_random <- predict(rf\_random, test) # make prediction

conf\_rf\_random <- confusionMatrix(test$classe, pred\_rf\_random) # confusion matrix

conf\_rf\_random$overall[1] # accuracy

varImp(rf\_random) # var importance - 20

**# Grid Search**

train\_control <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = 'grid')

rf\_grid <- train(classe ~ ., data = train, trControl = train\_control, method = "rf",

metric = 'Accuracy', tuneGrid = expand.grid(.mtry=c(1:15)))

pred\_rf\_grid <- predict(rf\_grid, test) # make prediction

conf\_rf\_grid <- confusionMatrix(test$classe, pred\_rf\_grid) # confusion matrix

conf\_rf\_grid$overall[1] # accuracy

varImp(rf\_grid) # var importance - 20

**# Goodness of Fit**

chisq.test(table(test$classe), prop.table(table(cvpred1))) # pv = 0.2202

chisq.test(table(test$classe), prop.table(table(pred\_rf\_default))) # pv = 0.2202

chisq.test(table(test$classe), prop.table(table(pred\_rf\_random))) # pv = 0.2202

chisq.test(table(test$classe), prop.table(table(pred\_rf\_grid))) # pv = 0.2202

**# Problem was to predict how well the activity is performed**

**# The target variable is the 5 classe; 1 accurate and 4 type of error**

**# occured during the activity**

**# error (target) detection was done by classifying an**

**# execution to one of the mistake classes**

**# we could detect mistakes fairly accurately**

**# Gradient bossting model is most accurate with less number of predictors**

**# Model is good fit and the Accuracy is 1**

plot <- plot(conf\_rf\_grid$table, col = topo.colors(6))