**# a. Create classification model using different classifiers**

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

**# a. Create classification model using different classifiers**

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

library(adabag); library(gbm)

**# bagging**

model\_bag <- bagging(classe ~., data = train , mfinal = 10) # model

model\_bag$importance

pred\_bag <- predict.bagging(model\_bag, newdata = test) # make prediction

pred\_bag

pred\_bag$confusion # confusion matrix

1-pred\_bag$error # accuracy

**# Boosting**

model\_boost <- boosting(classe ~., data = train, mfinal= 10, coeflearn = "Freund",

boos = FALSE, control = rpart.control(maxdepth = 3)) # model

model\_boost$importance

pred\_boost <- predict.boosting(model\_boost, newdata = test) # make prediction

pred\_boost$confusion # confusion matrix

1-pred\_boost$error # accuracy

**# Gradient Boosting**

train\_control <- trainControl(method = "repeatedcv", number = 5, repeats = 3, search = 'grid')

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

metric = 'Accuracy')

print(rf\_gbm)

plot(rf\_gbm)

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

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

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

summary(rf\_gbm) # var importance - 18

**# class**

model\_class <- tree(classe ~., data = train) # model

summary(model\_class)

plot(model\_class); text(model\_class)

pred\_class <- predict(model\_class, test, type = 'class') # make prediction

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

conf\_class

conf\_class$overall[1]

**# Random forest**

model\_rf <- randomForest(classe ~., train, ntree = 500)

model\_rf

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

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

conf\_rf

conf\_rf$overall[1]

**# Boosted Tree**

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

model\_bst <- train(classe ~ ., data = train, trControl = train\_control, method = "bstTree")

model\_bst

pred\_bst <- predict(model\_bst, test) # make prediction

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

conf\_bst

conf\_bst$overall[1]

**# CART**

model\_cart <- rpart(classe ~ ., data = train) # model

summary(model\_cart)

rpart.plot::rpart.plot(model\_cart)

plotcp(model\_cart)

pred\_cart <- predict(model\_cart, test, type = 'class') # make prediction

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

conf\_cart

conf\_cart$overall[1]

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

model\_cv <- train(classe ~ ., data = train, trControl = train\_control, method = "rpart")

model\_cv

pred\_cv <- predict(model\_cv, test) # make prediction

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

conf\_cv

conf\_cv$overall[1]

**# Ross Quinlan C5.0**

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

model\_c5.0 <- train(classe ~ ., data = train, trControl = train\_control, method = "C5.0")

model\_c5.0

pred\_c5.0 <- predict(model\_c5.0, test) # make prediction

conf\_c5.0 <- confusionMatrix(test$classe, pred\_c5.0) # confusion matrix

conf\_c5.0

conf\_c5.0$overall[1]

**# C5.0 Rules**

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

model\_c5.0rules <- train(classe ~ ., data = train, trControl = train\_control, method = "C5.0Rules")

model\_c5.0rules

pred\_c5.0rules <- predict(model\_c5.0rules, test) # make prediction

conf\_c5.0rules <- confusionMatrix(test$classe, pred\_c5.0rules) # confusion matrix

conf\_c5.0rules

conf\_c5.0rules$overall[1]

**# C5.0 Tree**

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

model\_c5.0tree <- train(classe ~ ., data = train, trControl = train\_control, method = "C5.0Tree")

model\_c5.0tree

pred\_c5.0tree <- predict(model\_c5.0tree, test) # make prediction

conf\_c5.0tree <- confusionMatrix(test$classe, pred\_c5.0tree) # confusion matrix

conf\_c5.0tree

conf\_c5.0tree$overall[1]

**# conditional inference trees**

# Ctree

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

model\_ctree <- train(classe ~ ., data = train, trControl = train\_control, method = "ctree")

model\_ctree

pred\_ctree <- predict(model\_ctree, test) # make prediction

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

conf\_ctree

conf\_ctree$overall[1]

**# Ctree2**

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

model\_ctree2 <- train(classe ~ ., data = train, trControl = train\_control, method = "ctree2")

model\_ctree2

pred\_ctree2 <- predict(model\_ctree2, test) # make prediction

conf\_ctree2 <- confusionMatrix(test$classe, pred\_ctree2) # confusion matrix

conf\_ctree2

conf\_ctree2$overall[1]

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

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

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

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

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

chisq.test(table(test$classe), prop.table(table(pred\_bst))) # pv = 0.2650

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

chisq.test(table(test$classe), prop.table(table(pred\_cv))) # pv = 0.2414

chisq.test(table(test$classe), prop.table(table(pred\_c5.0))) # pv = 0.2202

chisq.test(table(test$classe), prop.table(table(pred\_c5.0rules))) # pv = 0.2202

chisq.test(table(test$classe), prop.table(table(pred\_c5.0tree))) # pv = 0.2202

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

chisq.test(table(test$classe), prop.table(table(pred\_ctree2))) # pv = 0.2202

1- pred\_bag$error # 0.9809603

1- pred\_boost$error # 0.9991722

conf\_rf\_gbm$overall[1] # 1

conf\_class$overall[1] # 0.9511589

conf\_rf$overall[1] # 1

conf\_bst$overall[1] # 0.5612583

conf\_cart$overall[1] # 0.968543

conf\_cv$overall[1] # 0.8807947

conf\_c5.0$overall[1] # 1

conf\_c5.0rules$overall[1] # 1

conf\_c5.0tree$overall[1] # 1

conf\_ctree$overall[1] # 1

conf\_ctree2$overall[1] # 0.9312914

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

**# Performing cross-validation with the bagging method**

**# we use bagging.cv to make a 10-fold classification on the training dataset with 10 iterations:**

model\_bag\_cv <- bagging.cv(classe ~ ., data = train, v=10, mfinal = 10)

model\_bag\_cv$confusion

model\_bag\_cv$error

**# Performing cross-validation with the boosting method**

model\_boost\_cv <- boosting.cv(classe ~ ., data = train, v=10, mfinal = 10,

control = rpart.control(cp=0.01))

model\_boost\_cv$confusion

model\_boost\_cv$error

# 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, pred\_ctree) # 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

**# gradient boosting**

train\_control <- trainControl(method = "repeatedcv", number = 5, repeats = 3, search = 'grid')

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

metric = 'Accuracy')

print(rf\_gbm)

plot(rf\_gbm)

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

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

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

summary(rf\_gbm) # var importance - 18

**# 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$table, col = topo.colors(6))**