Task 1:

1. Use the below-given data set

Data Set

2. Perform the below-given activities:

a. Predict the no of comments in next H hrs

Note:-

1. Use LASSO, Elastic Net and Ridge and other regression techniques that are covered in the module

2. Report the training accuracy and test accuracy

3. compare with linear models and report the accuracy

4. create a graph displaying the accuracy of all models

Solution:

1. Use LASSO, Elastic Net and Ridge and other regression techniques that are covered in the module

```
library(tidyverse)

library(caret)

library(glmnet)

# Load the data

setwd("~/Dataset/Dataset/Training")

Features_Variant_1 <-

read.csv("C:/users/seshan/Documents/Dataset/Dataset/Training/Features_Variant_1.csv")

View(Features_Variant_1)

Features.data <- na.omit(Features_Variant_1)

# Split the data into training and test set

set.seed(123)

training.samples <- Features$X0.19 %>%

 createDataPartition(p = 0.8, list = FALSE)

train.data <- Features_Variant_1[training.samples, ]
```

```
test.data <- Features_Variant_1[-training.samples, ]

# Predictor variables

x <- model.matrix(X0.19~., train.data)[,-1]

# Outcome variable

y <- train.data$X0.19

glmnet(x, y, alpha = 1, lambda = NULL)

# Find the best lambda using cross-validation

set.seed(123)

cv <- cv.glmnet(x, y, alpha = 0)

# Display the best lambda value

cv$lambda.min

plot(cv$lambda.min)

# Fit the final model on the training data

model <- glmnet(x, y, alpha = 0, lambda = cv$lambda.min)

plot(model)

# Display regression coefficients

coef(model)

# Make predictions on the test data

x.test <- model.matrix(X0.19 ~., test.data)[,-1]

predictions <- model %>% predict(x.test) %>% as.vector()

# Model performance metrics

data.frame(

 RMSE = RMSE(predictions, test.data$X0.19),

 Rsquare = R2(predictions, test.data$X0.19)

)

#Computing lasso regression

# Find the best lambda using cross-validation
```

set.seed(123)

cv <- cv.glmnet(x, y, alpha = 1)

Display the best lambda value

Task 2:

1. Use the below-given data set

Data Set

2. Perform the below-given activities:

a. Create a classification model using a logistic regression model

b. verify model goodness of fit

c. Report the accuracy measures

d. Report the variable importance

e. Report the unimportant variables

Solution:

input<- weight_lifting_exercises

View(input)

input1<- as.numeric(input$new_window)

model<-
glm(input1~raw_timestamp_part_1+raw_timestamp_part_2+cvtd_timestamp+num_window+roll_belt+pitch_belt
+yaw_belt+total_accel_belt,data = input)

model

summary(model)

predict<- predict(model, type = "response")

head(predict, 5)

input$predict<- predict

input$predictROUND<- round(predict, digits = 0)

table(input$new_window, predict>= 0.5)

dim(input)

In R script

1. Use the below given data set

2. Perform the below given activities:

a. Create classification model using logistic regression model

```
> predict<- predict(model, type = "response")
> head(predict, 5)
        1         2         3         4         5 
0.9604327 0.9608404 0.9589231 0.9600989 0.9607629 
> input$predict<- predict
> input$predictROUND<- round(predict, digits = 0)
> table(input$new_window, predict>= 0.5)
     
      TRUE
  no  3936
  yes   88
> dim(input)
[1] 4024  161
```

b. verify model goodness of fit

c. Report the accuracy measures

f. interpret the results

g. visualize the results

for questions (b,c,f,g) – Ans is as below

```
> model<-
glm(input1~raw_timestamp_part_1+raw_timestamp_part_2+cvtd_timestamp+num_window+roll_belt+pitch_belt
+yaw_belt+total_accel_belt,data = input)
> model

Call:  glm(formula = input1 ~ raw_timestamp_part_1 + raw_timestamp_part_2 + 
    cvtd_timestamp + num_window + roll_belt + pitch_belt + yaw_belt + 
    total_accel_belt, data = input)

Coefficients:
                 (Intercept)          raw_timestamp_part_1          raw_timestamp_part_2  cvtd_timestamp05-12-2011 
11:23  cvtd_timestamp05-12-2011 11:25  
                  -9.841e+05                     7.440e-04                     1.242e-07                    -1.869e+02                    -
1.870e+02  
cvtd_timestamp05-12-2011 14:22  cvtd_timestamp05-12-2011 14:23  cvtd_timestamp28-11-2011 14:15  
cvtd_timestamp30-11-2011 17:12                    num_window  
                  -1.949e+02                    -1.949e+02                     2.554e+02                     1.192e+02                    -8.201e-
04  
                   roll_belt                    pitch_belt                      yaw_belt              total_accel_belt  
                  -4.217e-04                    -4.897e-04                     9.792e-05                     2.525e-03  

Degrees of Freedom: 4023 Total (i.e. Null);  4010 Residual
```

```
Null Deviance:	    86.08 
Residual Deviance: 80.79 	AIC: -4277




> summary(model)

Call:
glm(formula = input1 ~ raw_timestamp_part_1 + raw_timestamp_part_2 + 
    cvtd_timestamp + num_window + roll_belt + pitch_belt + yaw_belt + 
    total_accel_belt, data = input)

Deviance Residuals: 
     Min        1Q    Median        3Q       Max  
-0.25039  -0.04901  -0.01883   0.01123   0.96934  

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)    
(Intercept)               -9.841e+05  4.774e+06  -0.206    0.837    
raw_timestamp_part_1       7.440e-04  3.609e-03   0.206    0.837    
raw_timestamp_part_2       1.242e-07  7.888e-09  15.747   <2e-16 ***
cvtd_timestamp05-12-2011 11:23 -1.869e+02  9.069e+02  -0.206    0.837    
cvtd_timestamp05-12-2011 11:25 -1.870e+02  9.072e+02  -0.206    0.837    
cvtd_timestamp05-12-2011 14:22 -1.949e+02  9.455e+02  -0.206    0.837    
cvtd_timestamp05-12-2011 14:23 -1.949e+02  9.455e+02  -0.206    0.837    
cvtd_timestamp28-11-2011 14:15  2.554e+02  1.239e+03   0.206    0.837    
cvtd_timestamp30-11-2011 17:12  1.192e+02  5.766e+02   0.207    0.836    
num_window                -8.201e-04  4.223e-03  -0.194    0.846    
roll_belt                 -4.217e-04  5.029e-04  -0.839    0.402    
pitch_belt                -4.897e-04  1.151e-03  -0.426    0.670    
yaw_belt                   9.792e-05  1.168e-04   0.839    0.402    
total_accel_belt           2.525e-03  1.896e-03   1.332    0.183    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.02014796)

    Null deviance: 86.076  on 4023  degrees of freedom
Residual deviance: 80.793  on 4010  degrees of freedom
AIC: -4276.7

Number of Fisher Scoring iterations: 2
```

d. Report the variable importance

Ans: variables highlighted in pink are important variables

e. Report the unimportant variables

Ans: variables highlighted in green are unimportant variables

Task 3:

1. Use the below-given data set

DataSet

2. Perform the below-given activities:

a. Create a classification model using different decision trees.

b. Verify model goodness of fit.

c. Apply all the model validation techniques.

d. Make conclusions

Solution:

```
View(weight_lifting_exercises)
str(weight_lifting_exercises)
weight_lifting_exercises<-data.frame(weight_lifting_exercises[,-
c(11:35,49:58,68:82,86:100,102:111,124:138,140:149)])
str(weight_lifting_exercises)
summary(weight_lifting_exercises)
weightTrain<-weight_lifting_exercises[1:2012,]
weightTest<-weight_lifting_exercises[2013:4024,]
summary(weightTrain)
names(weightTrain)
#Ques.2. Perform the below given activities:
# a. Create classification model using different decision trees.
weightTrain<-data.frame(weightTrain[,-c(11:35,49:58,68:82,86:100,102:111,124:138,140:149)])
library(caret)
library(Hmisc)
weightTrain$raw_timestamp_part_1<-impute(weightTrain$raw_timestamp_part_1,mean)
weightTrain$raw_timestamp_part_2<-impute(weightTrain$raw_timestamp_part_2,mean)
weightTrain$cvtd_timestamp<-impute(weightTrain$cvtd_timestamp,mean)
weightTrain$new_window<-impute(weightTrain$new_window,mean)
weightTrain$num_window<-impute(weightTrain$num_window,mean)
weightTrain$roll_belt<-impute(weightTrain$roll_belt,mean)
weightTrain$pitch_belt<-impute(weightTrain$pitch_belt,mean)
weightTrain$yaw_belt<-impute(weightTrain$yaw_belt,mean)
summary(weightTrain)
str(weightTrain)
weightTrain$cvtd_timestamp<-as.integer(weightTrain$cvtd_timestamp)
weightTrain$new_window<-as.integer(weightTrain$new_window)
```

```
library(tree)
tree<-tree(classe~. ,
        data = weightTrain)
plot(tree,pretty = 0.1)
text(tree,pretty = 1.2)
summary(tree)
library(caret)
pred <- predict(tree,weightTrain,type='class')
str(pred)
dim(pred)
dim(weightTest$classe)
weightTest$classe<-as.factor(weightTest$classe)
dim(weightTest$classe)
table(weightTest$classe,pred)
length(pred)
length(weightTest$classe)
confusionMatrix(pred,weightTest$classe)
#.........
install.packages("rpart")
library(rpart)
fit1 <- rpart(classe~.,data=weightTrain[,-1])
class(fit1)
summary(fit1)
rpart.plot::rpart.plot(fit1)
pred1<-predict(fit1,weightTrain,type = "class")
summary(pred1)
dim(pred1)
weightTest$classe<-as.factor(weightTest$classe)
table(weightTest$classe,pred1)
confusionMatrix(weightTest$classe,pred1)
# b. Verify model goodness of fit.
#........for pred.....
weightTest$classe<-as.factor(weightTest$classe)
dim(weightTest$classe)
table(weightTest$classe,pred)
length(pred)
length(weightTest$classe)
confusionMatrix(pred,weightTest$classe)
#...for fit1....
weightTest$classe<-as.factor(weightTest$classe)
table(weightTest$classe,pred1)
confusionMatrix(weightTest$classe,pred1)
# c. Apply all the model validation techniques.
```

```
set.seed(3)
install.packages('tree')
library(tree)
cv.weight<-cv.tree(tree,FUN = prune.misclass)    #cv->cross validation
cv.weight_lifting_exercises<-cv.tree(tree,FUN = prune.misclass)
names(cv.weight)
cv.weight
par(mfrow = c(1,2))
plot(cv.weight$size,cv.weight$dev,type = 'b',col = 'red')
prune.weight<-prune.misclass(tree,best = 9)
plot(prune.weight)
text(prune.weight,pretty = 0)
weightTrain$cvtd_timestamp<-as.integer(weightTrain$cvtd_timestamp)
weightTrain$new_window<-as.integer(weightTrain$new_window)
tree.pred1<-predict(prune.weight,weightTrain,type = 'class')
table(tree.pred1,weightTest)
#............Random forest.........
library(randomForest)
set.seed(1)
a.weight_lifting_exercises<-randomForest(classe~.,weight_lifting_exercises,
                                subset = weightTrain,mtry = 3,importance = TRUE)
dim(a.weight_lifting_exercises)
importance(a.weight_lifting_exercises)
varImpPlot(a.weight_lifting_exercises,col = 'blue',pch = 10, cex = 1.25)
a.weight_lifting_exercises
test.pred.rf<-predict(a.weight_lifting_exercises, newdata = weight_lifting_exercises[-weightTrain,],type =
'class')
table(test.pred.rf,weightTest)
#.........adaboost..........
install.packages(adabag)
library(adabag)
set.seed(300)
weight_lifting_exercises$classe<-as.character(weight_lifting_exercises$classe)
weight_adaboost<-boosting(classe~., data = weight_lifting_exercises)
p.weight_adaboost<-predict(weight_adaboost,weight_lifting_exercises)
head(p.weight_adaboost)
head(p.weight_adaboost$class)
p.weight_adaboost$confusion
set.seed(300)
car_adaboost_cv<-boosting.cv(classe,data = weight_lifting_exercises)
car_adaboost_cv$confusion
```

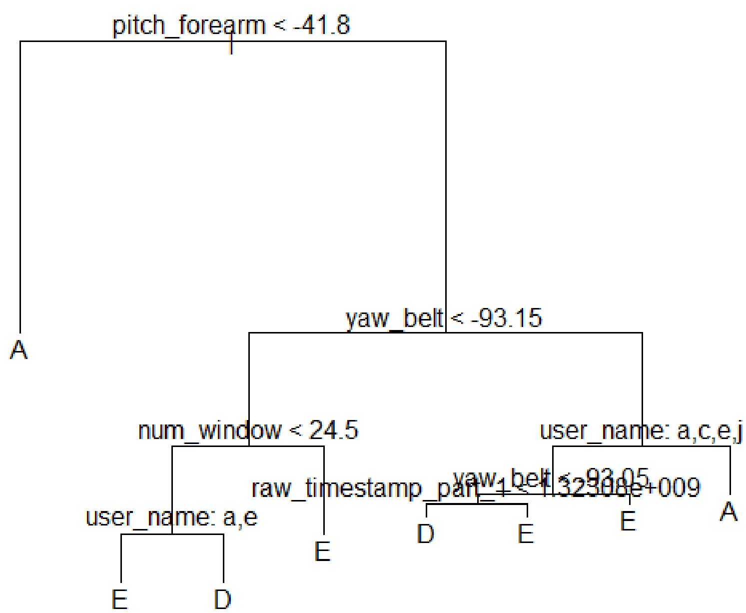
In R script

1. Use the below given data set

2. Perform the below given activities:

a. Create classification model using different decision trees.

<u>Classification based on Decision Tree method</u>



,,,,,

```
> summary(tree)

Classification tree:
tree(formula = classe ~ ., data = weightTrain)
Variables actually used in tree construction:
[1] "pitch_forearm"     "yaw_belt"      "num_window"      "user_name"       "raw_timestamp_part_1"
Number of terminal nodes:  8
Residual mean deviance:  0.003518 = 7.051 / 2004
Misclassification error rate: 0.000497 = 1 / 2012

> table(weightTest$classe,pred)
   pred
     A   B   C   D   E
```

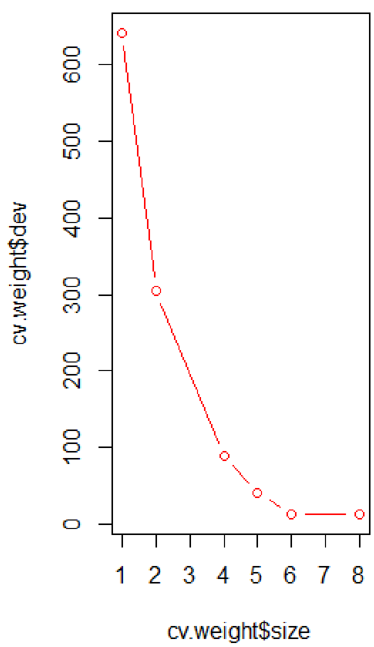
```
A 337   0   0 248 414
B   0   0   0  17 884
C  29   0   0   0  83
D   0   0   0   0   0
E   0   0   0   0   0
```

<u>Classification based on Decision Tree method</u>

b. Verify model goodness of fit.

```
> table(weightTest$classe,pred)
   pred
     A   B   C   D   E
A 337   0   0 248 414
B   0   0   0  17 884
C  29   0   0   0  83
D   0   0   0   0   0
E   0   0   0   0   0
```

c. Apply all the model validation techniques.



d. Make conclusions.

Model having best classification accuracy is selected

Task 4:

1. Use the below-given data set

DataSet

2. Perform the below-given activities:

a. Create a classification model using different classifiers

b. Verify model goodness of fit

c. Apply all the model validation techniques.

Solution:

```
View(cs2m)

#2. Perform the below given activities:

#a. Create classification model using different classifiers

#b. Verify model goodness of fit

#c. Apply all the model validation techniques.

#classification

library(caTools)

library(tree)

#splitting

set.seed(1)

split<- sample.split(cs2m$classe,SplitRatio = 0.70)

cs2mTrain <- subset(cs2m,split == TRUE)

cs2mTest<- subset(cs2m, split == FALSE)

table(cs2m$classe)

table(cs2mTrain$classe)

table(cs2mTest$classe)

prop.table(table(cs2mTest$classe))

table(cs2mTest$classe)
```

```
prop.table(table(cs2mTrain$classe))

modelClassTree<-
tree(classe~cvtd_timestamp+total_accel_belt+yaw_dumbbell+roll_forearm+accel_forearm_y,data = cs2mTrain)

plot(modelClassTree)

text(modelClassTree,pretty = 0 ,cex=0.75)

pred<- predict(modelClassTree,newdata= cs2mTest)

head(pred,3)

cs2m$predict <- predict

cs2m$predictROUND<- round(predict,digits = 0)

#confusion matrix

table(cs2m$classe,predict>= 0.5)

sum<- sum(table(cs2m$classe,predict>= 0.5))

#interpretation, Accuracy and model goodness  of our model

#accuracy of our model

accuracy<- (1185+679)/(2266)

accuracy

#0.8225949

#model goodness

library(verification)

predictTrain<- predict(model,cs2m,type="response")

table(cs2m$classe,predictTrain >=0.5)

head(predictTrain,3)

auc(cs2m$classe,predictTrain)

#conclusions

#****NOTE****

#Area under the curve: 0.9333333

#also our accuracy of our model is 0.8225949
```

#also by seeing various measures like ME,RSS,RMSE,MAPE of our tree which is godd

#by this all things we conclude that our model is good and fit

Task 5:

1. Use the below given data set

Data Set

2. 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

Solution:

View(weight_lifting_exercises)

str(weight_lifting_exercises)

weight_lifting_exercises<-data.frame(weight_lifting_exercises[,-c(11:35,49:58,68:82,86:100,102:111,124:138,140:149)])

str(weight_lifting_exercises)

summary(weight_lifting_exercises)

weightTrain<-weight_lifting_exercises[1:2012,]

weightTest<-weight_lifting_exercises[2013:4024,]

summary(weightTrain)

names(weightTrain)

#Ques.2. Perform the below given activities:

a. Create classification model using different random forest.

```
install.packages("randomForest")

library(randomForest)

set.seed(1)

bag.weight_lifting_exercises <- randomForest(classe~.,weight_lifting_exercises,

                                subset = weightTrain, mtry = 3,importance = TRUE)

dim(bag.weight_lifting_exercises)

#e plot importance of variables

importance(bag.weight_lifting_exercises)

varImpPlot(bag.weight_lifting_exercises,col = 'blue',pch = 10, cex = 1.25)

bag.weight_lifting_exercises
```

b. Verify model goodness of fit.

```
#........for pred.....

test.pred.bag<-predict(bag.weight_lifting_exercises, newdata = weight_lifting_exercises[-weightTrain, ],type = 'class')

table(test.pred.rf,weightTest)
```

c. Apply all the model validation techniques.

```
set.seed(3)

install.packages('tree')

library(tree)

tree.weight_lifting_exercises1<-tree(classe~. , weight_lifting_exercises, subset = weightTrain)

cv.weight_lifting_exercises<-cv.tree(tree.weight_lifting_exercises1,FUN = prune.misclass)  #cv->cross validation

names(cv.weightlifting_exercises)

cv.weightlifting_exercises

par(mfrow = c(1,2))

plot(cv.weight$size,cv.weight$dev,type = 'b',col = 'red')
```

```
prune.weight<-prune.misclass(tree,best = 9)

plot(prune.weight)

text(prune.weight,pretty = 0)

weightTrain$cvtd_timestamp<-as.integer(weightTrain$cvtd_timestamp)

weightTrain$new_window<-as.integer(weightTrain$new_window)

tree.pred1<-predict(prune.weight,weightTrain,type = 'class')

table(tree.pred1,weightTest)

#.........adaboost..........

install.packages(adabag)

library(adabag)

set.seed(300)

weight_lifting_exercises$classe<-as.character(weight_lifting_exercises$classe)

weight_adaboost<-boosting(classe~., data = weight_lifting_exercises)

p.weight_adaboost<-predict(weight_adaboost,weight_lifting_exercises)

head(p.weight_adaboost)

head(p.weight_adaboost$class)

p.weight_adaboost$confusion

set.seed(300)

car_adaboost_cv<-boosting.cv(classe,data = weight_lifting_exercises)

car_adaboost_cv$confusion
```