```
# Predict the no of comments in next H hrs

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


library(dplyr)

library(corrplot)

library(car)

library(MASS)

library(forecast)

library(glmnet)


# import train data set

Variant_1 <- read.csv("C:/Users/TejsD/ownloads/Dataset/fbtrain/Features_Variant_1.csv", header=FALSE)

Variant_2 <- read.csv("C:/Users/TejsD/ownloads/Dataset/fbtrain/Features_Variant_2.csv", header=FALSE)

Variant_3 <- read.csv("C:/Users/TejsD/ownloads/Dataset/fbtrain/Features_Variant_3.csv", header=FALSE)

Variant_4 <- read.csv("C:/Users/TejsD/ownloads/Dataset/fbtrain/Features_Variant_4.csv", header=FALSE)

Variant_5 <- read.csv("C:/Users/TejsD/ownloads/Dataset/fbtrain/Features_Variant_5.csv", header=FALSE)

fbtrain <- rbind(Variant_1, Variant_2, Variant_3, Variant_4, Variant_5)

dim(fbtrain)


# import test data set

setwd("C:/Users/TejsD/ownloads/Dataset/fbtest")

test1 <- read.csv("Test_Case_1.csv", header = F); test2 <- read.csv("Test_Case_2.csv", header = F)

test3 <- read.csv("Test_Case_3.csv", header = F); test4 <- read.csv("Test_Case_4.csv", header = F)

test5 <- read.csv("Test_Case_5.csv", header = F); test6 <- read.csv("Test_Case_6.csv", header = F)
```

```
test7 <- read.csv("Test_Case_7.csv", header = F); test8 <- read.csv("Test_Case_8.csv", header = F)

test9 <- read.csv("Test_Case_9.csv", header = F); test10 <- read.csv("Test_Case_10.csv", header = F)

fbtest  <- rbind(test1, test2, test3, test4, test5, test6, test7, test8, test9, test10)

dim(fbtest)


# Assign variable names to the train and test data set

colnames(fbtrain) <-
c("plikes","checkin","talking","category","d5","d6","d7","d8","d9","d10","d11","d12",

            "d13","d14","d15","d16","d17","d18","d19","d20","d21","d22","d23","d24","d25","d26",

            "d27","d28","d29","cc1","cc2","cc3","cc4","cc5","basetime","postlength","postshre",

            "postpromo","Hhrs","sun","mon","tue","wed","thu","fri","sat","basesun","basemon",

            "basetue","basewed","basethu","basefri","basesat","target")

colnames(fbtest) <-
c("plikes","checkin","talking","category","d5","d6","d7","d8","d9","d10","d11","d12",


"d13","d14","d15","d16","d17","d18","d19","d20","d21","d22","d23","d24","d25","d26",

             "d27","d28","d29","cc1","cc2","cc3","cc4","cc5","basetime","postlength","postshre",

             "postpromo","Hhrs","sun","mon","tue","wed","thu","fri","sat","basesun","basemon",

             "basetue","basewed","basethu","basefri","basesat","target")


dim(fbtrain)

dim(fbtest)

View(fbtrain)

View(fbtest)

str(fbtrain)

str(fbtest)


train <- fbtrain; test <- fbtest
```

```
distinct(train)   # removing overlapping observations if any

dim(train)

colSums(is.na(train)) # no missing values


x.train <- as.matrix(train[,-54]) ; y.train <- train[,54]

x.test <- as.matrix(test[,-54]) ; y.test <- test[,54]


#------------------------------------------------------------------

# Predict the no of comments in next H hrs

#------------------------------------------------------------------


# LEAST ANGLE REGRESSION (LARS)

library(lars)

fit_lars <- lars(x.train, y.train, type = 'lar')

summary(fit_lars)

fit_lars

# select step with minimum error

best_step <- fit_lars$df[which.min(fit_lars$RSS)]

best_step

# Make PRedictions

predictions_lars <- predict(fit_lars, x.train, s= best_step, type = "fit")

# summarise accuracy

mse_lars <- mean((y.train - predictions_lars$fit)^2)

mse_lars


#-------------------------------------------------------------------

# LASSO
```

```
library(glmnet)

fit_lasso <- glmnet(x.train, y.train, family = "gaussian",alpha = 1, lambda=0.001)

fit_lasso

summary(fit_lasso)

# Make PRedictions

predictions_lasso <- predict(fit_lasso, x.train, type = "link")

# summarise accuracy

mse_lasso <- mean((y.train - predictions_lasso)^2)

mse_lasso

#--------------------------------------------------------------------------

# RIDGE


fit_ridge <- glmnet(x.train, y.train, family = "gaussian",alpha = 0, lambda=0.001)

fit_ridge

summary(fit_ridge)

# Make PRedictions

predictions_ridge <- predict(fit_ridge, x.train, type = "link")

# summarise accuracy

mse_ridge <- mean((y.train - predictions_ridge)^2)

mse_ridge

#----------------------------------------------------------------------------

# Elastic Net


for (i in 0:10) {

  assign(paste("fit", i, sep=""), glmnet(x.train, y.train, family="gaussian", alpha=i/10, lambda = 0.001))

}

# 10-fold Cross validation for each alpha = 0, 0.1, ... , 0.9, 1.0
```

```
# (For plots on Right)

# Predictions

yhat0 <- predict(fit0, s=fit0$lambda.1se, newx=x.train)

yhat1 <- predict(fit1, s=fit1$lambda.1se, newx=x.train)

yhat2 <- predict(fit2, s=fit2$lambda.1se, newx=x.train)

yhat3 <- predict(fit3, s=fit3$lambda.1se, newx=x.train)

yhat4 <- predict(fit4, s=fit4$lambda.1se, newx=x.train)

yhat5 <- predict(fit5, s=fit5$lambda.1se, newx=x.train)

yhat6 <- predict(fit6, s=fit6$lambda.1se, newx=x.train)

yhat7 <- predict(fit7, s=fit7$lambda.1se, newx=x.train)

yhat8 <- predict(fit8, s=fit8$lambda.1se, newx=x.train)

yhat9 <- predict(fit9, s=fit9$lambda.1se, newx=x.train)

yhat10 <- predict(fit10, s=fit10$lambda.1se, newx=x.train)

mse0 <- mean((y.train - yhat0)^2)

mse1 <- mean((y.train - yhat1)^2)

mse2 <- mean((y.train - yhat2)^2)

mse3 <- mean((y.train - yhat3)^2)

mse4 <- mean((y.train - yhat4)^2)

mse5 <- mean((y.train - yhat5)^2)

mse6 <- mean((y.train - yhat6)^2)

mse7 <- mean((y.train - yhat7)^2)

mse8 <- mean((y.train - yhat8)^2)

mse9 <- mean((y.train - yhat9)^2)

mse10 <- mean((y.train - yhat10)^2)


mse_elastic <- c(mse0,mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10)

mse_elastic
```

```
mse_elnet <- mse_elastic[which.min(mse_elastic)]

mse_elnet

#------------------------------------------------------------------------------

# MARS - Multivariate Adaptive Regression Splines

library(earth)

fit_mars <- earth(target~., data = train)

fit_mars

summary(fit_mars)    # Model Summary

evimp(fit_mars)      # Summary of importance of input variables

# Make PRedictions

predictions_mars <- predict(fit_mars, train)

predictions_mars

# summarise accuracy

mse_mars <- mean((y.train - predictions_mars)^2)

mse_mars


#------------------------------------------------------------------------------

# Stepwise Regression

# TARGET <- lm(target~., data = train)

library(MASS)

#step <- stepAIC(TARGET, direction = "both")


final_model <- lm(target ~ checkin + talking + d5 + d6 + d7 + d8 + d9 + d10 + d12 +

            d13 + d14 + d17 + d18 + d19 + d21 + d22 + d23 + d24 + d25 +

            d26 + d28 + d29 + cc1 + cc2 + cc3 + cc4 + basetime + postshre +

            Hhrs + tue + wed + thu + fri + basesun + basemon + basetue +

            basewed + basethu, data = train)
```

```
# Fine tune the model and represent important features
fit_step <- lm(target ~ checkin + talking + d5 + d6 + d7 + d8 + d10 + d12 +
               d13 + d17 + d18 + d19 + d22 + d23 + d25 +
               d26 + d28 + d29 + cc2 + cc3 + cc4 + basetime + postshre +
               Hhrs, data = train)
fit_step
summary(fit_step)
# Make PRedictions
predictions_step <- predict(fit_step, train)
predictions_step
# summarise accuracy
mse_step <- mean((y.train - predictions_step)^2)
mse_step

#---------------------------------------------------------------------------
# Principal Component Regression ( PCR)
library(pls)
fit_pcr <- pcr(target~., data=train, validation = "CV")
fit_pcr
summary(fit_pcr)
# Make PRedictions
predictions_pcr <- predict(fit_pcr, train)
as.data.frame(predictions_pcr)[,1]
# summarise accuracy
mse_pcr <- mean((y.train - predictions_pcr)^2)
mse_pcr
```

```
#-----------------------------------------------------------------------------
# PArtial Least Squares
fit_pls <- plsr(target~., data=train, validation = "CV")
fit_pls
summary(fit_pls)
# Make PRedictions
predictions_pls <- predict(fit_pls, train)
predictions_pls
# summarise accuracy
mse_pls <- mean((y.train - predictions_pls)^2)
mse_pls
#-----------------------------------------------------------------------------
# Robust Regression
fit_robust <- rlm(formula = target~., psi = psi.huber,data=train)
fit_robust
summary(fit_robust)
# Make PRedictions
predictions_robust <- predict(fit_robust, train)
predictions_robust
# summarise accuracy
mse_robust <- mean((y.train - predictions_robust)^2)
mse_robust


#-----------------------------------------------------------------------------
# using decision tree
library(rpart)
fit_tree <- rpart(target ~ ., data = train)
```

```
summary(fit_tree)

# Make PRedictions

predictions_tree <- predict(fit_tree, train)

# summarise accuracy

mse_tree <- mean((y.train - predictions_tree)^2)

mse_tree


#-----------------------------------------------------------------------------


# comparing the models and accuracy

Accu <- data.frame(

  Model=c("LArs","Lasso","Ridge","ELNET","MARS","STEP","PCR","Tree"),

  Accuracy = c(mse_lars,mse_lasso,mse_ridge,mse_elnet,mse_mars,mse_step,

             mse_pcr,mse_tree))

Accu$Accuracy <- round(Accu$Accuracy,0)

ACCU <- Accu[which.min(Accu$Accuracy),]

ACCU


# Decision Tree has the minimum error hence the better model amongst all


# Graphical displaying the MSE of all the models

par(mfrow=c(1,1))

x <- barplot(Accu$Accuracy, xlab = "Model", ylab = "MSE", col = heat.colors(8),

       names.arg = c("LArs","Lasso","Ridge","ELNET","MARS","STEP","PCR","Tree"),

       angle = 45, lwd =3, las = 2)

text(x, 0, Accu$Accuracy, cex=1, pos=3, srt = 45)
```

```
new <- data.frame(actual = train[,54], lars = predictions_lars$fit,
        lasso = predictions_lasso, ridge = predictions_ridge,
        elnet = yhat10, mars = predictions_mars, step = predictions_step,
        pcr = as.data.frame(predictions_pcr)[,1], tree = predictions_tree)
colnames(new) <- c("Actual","Lars","Lasso","Ridge","elnet","mars","step","pcr","tree")

# Calculating residual from the predictions from all models
new$LarsRes <- new$Actual-new$Lars; new$LassoRes <- new$Actual-new$Lasso;
new$RidgeRes <- new$Actual-new$Ridge; new$elnetRes <- new$Actual-new$elnet;
new$marsRes <- new$Actual-new$mars; new$stepRes <- new$Actual-new$step;
new$pcrRes <- new$Actual-new$pcr; new$treeRes <- new$Actual-new$tree

#  plotting of Residuals Vs. Fitted

scatterplot(new$Lars,new$LarsRes)
scatterplot(new$Lasso,new$LassoRes)
scatterplot(new$Ridge,new$RidgeRes)
scatterplot(new$elnet,new$elnetRes)
scatterplot(new$mars,new$marsRes)
scatterplot(new$step,new$stepRes)
scatterplot(new$pcr,new$pcrRes)
scatterplot(new$tree,new$treeRes)
```