# 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("E:/Data Analytics with RET/Assignment/Dataset/fbtrain/Features\_Variant\_1.csv", header=FALSE)

Variant\_2 <- read.csv("E:/Data Analytics with RET/Assignment/Dataset/fbtrain/Features\_Variant\_2.csv", header=FALSE)

Variant\_3 <- read.csv("E:/Data Analytics with RET/Assignment/Dataset/fbtrain/Features\_Variant\_3.csv", header=FALSE)

Variant\_4 <- read.csv("E:/Data Analytics with RET/Assignment/Dataset/fbtrain/Features\_Variant\_4.csv", header=FALSE)

Variant\_5 <- read.csv("E:/Data Analytics with RET/Assignment/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("E:/Data Analytics with RET/Assignment/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)