**Session 16 – Assignment-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

Attribute Information:

(39 - This describes the H hrs, for which we have the target variable/ comments received.

, 54 -Target Variable - Decimal Target The no of comments in next H hrs(H is given in Feature no 39).

39

H Local

ï¿¼Decimal(0-23) Encoding

Other feature

This describes the H hrs, for which we have the target variable/ comments received.

54

Target Variable

Decimal

Target

The no of comments in next H hrs(H is given in Feature no 39).

Prediction Accuracy

A good learner is the one which has good prediction accuracy; in other words, which has the

smallest prediction error.

Let us try to understand the prediction problem intuitively. Consider the simple case of fitting a linear

regression model to the observed data. A model is a good fit, if it provides a high R

2

value.

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

cv$lambda.min

# Fit the final model on the training data

model <- glmnet(x, y, alpha = 1, lambda = cv$lambda.min)

# Dsiplay 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 elastic net regession

# Build the model using the training set

set.seed(123)

model <- train(X0.19 ~., data = train.data, method = "glmnet",

trControl = trainControl("cv", number = 10),

tuneLength = 10

)

# Best tuning parameter

model$bestTun

plot(model$bestTun)

# Coefficient of the final model. You need

# to specify the best lambda

coef(model$finalModel, model$bestTune$lambda)

# Make predictions on the test data

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

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

# Model performance metrics

data.frame(

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

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

)

#Comparing the different models

#Using caret package

#Setup a grid range of lambda values:

lambda <- 10^seq(-3, 3, length = 100)

#Compute ridge regression

# Build the model

set.seed(123)

ridge <- train(

X0.19 ~., data = train.data, method = "glmnet",

trControl = trainControl("cv", number = 10),

tuneGrid = expand.grid(alpha = 0, lambda = lambda)

)

# Model coefficients

coef(ridge$finalModel, ridge$bestTune$lambda)

# Make predictions

predictions <- ridge %>% predict(test.data)

plot(predictions)

# Model prediction performance

data.frame(

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

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

)

#Compute lasso regression

# Build the model

set.seed(123)

lasso <- train(

X0.19 ~., data = train.data, method = "glmnet",

trControl = trainControl("cv", number = 10),

tuneGrid = expand.grid(alpha = 1, lambda = lambda)

)

plot( lasso)

# Model coefficients

coef(lasso$finalModel, lasso$bestTune$lambda)

# Make predictions

predictions <- lasso %>% predict(test.data)

# Model prediction performance

data.frame(

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

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

)

#Elastic net regression

# Build the model

set.seed(123)

elastic <- train(

X0.19 ~., data = train.data, method = "glmnet",

trControl = trainControl("cv", number = 10),

tuneLength = 10

)

# Model coefficients

coef(elastic$finalModel, elastic$bestTune$lambda)

# Make predictions

predictions <- elastic %>% predict(test.data)

plot( predictions)

# Model prediction performance

data.frame(

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

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

)

#Comparing models performance:

models <- list(ridge = ridge, lasso = lasso, elastic = elastic)

resamples(models) %>% summary( metric = "RMSE")

#k-fold Cross Validation

# load the library

library(caret)

# define training control

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

# fix the parameters of the algorithm

grid <- expand.grid(.fL=c(0), .usekernel=c(FALSE))

# train the model

model <- train(X0.19~., data=Features\_Variant\_1, trControl=train\_control, method="nb",

tuneGrid=grid)

# summarize results

print(model)

# load the library

library(caret)

# define training control

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

# train the model

model <- train(X0.19~., data=Features\_Variant\_1, trControl=train\_control, method="nb")

# summarize results

print(model)

#create a graph displaying the accuracy of all models

plot(model)

plot(varImp(ridge$finalModel))

plot(cv)

plot(ridge)

hist(Features$X0.19,col = "green")

hist(Features$X24,col = "red")

hist(Features$X11.291044776119403,col = 'yellow')

fit = glmnet(x, y)

plot(fit)

cvfit = cv.glmnet(x, y)

plot(cvfit)

tfit=glmnet(x,y,lower=-.7,upper=.5)

plot(tfit)

> 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)

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

Df %Dev Lambda

[1,] 0 0.00000 18.250000

[2,] 1 0.04881 16.630000

[3,] 1 0.08933 15.150000

[4,] 1 0.12300 13.810000

[5,] 1 0.15090 12.580000

[6,] 1 0.17410 11.460000

[7,] 1 0.19330 10.440000

[8,] 1 0.20930 9.516000

[9,] 1 0.22260 8.671000

[10,] 1 0.23360 7.900000

[11,] 1 0.24270 7.198000

[12,] 1 0.25030 6.559000

[13,] 1 0.25660 5.976000

[14,] 2 0.26350 5.445000

[15,] 2 0.26950 4.962000

[16,] 3 0.27660 4.521000

[17,] 3 0.28290 4.119000

[18,] 4 0.28840 3.753000

[19,] 4 0.29290 3.420000

[20,] 4 0.29670 3.116000

[21,] 6 0.30040 2.839000

[22,] 6 0.30370 2.587000

[23,] 6 0.30640 2.357000

[24,] 7 0.30870 2.148000

[25,] 7 0.31060 1.957000

[26,] 8 0.31210 1.783000

[27,] 8 0.31350 1.625000

[28,] 8 0.31490 1.480000

[29,] 8 0.31610 1.349000

[30,] 8 0.31710 1.229000

[31,] 8 0.31790 1.120000

[32,] 8 0.31860 1.020000

[33,] 8 0.31920 0.929700

[34,] 8 0.31970 0.847100

[35,] 9 0.32020 0.771900

[36,] 9 0.32060 0.703300

[37,] 9 0.32090 0.640800

[38,] 9 0.32120 0.583900

[39,] 10 0.32150 0.532000

[40,] 10 0.32180 0.484800

[41,] 11 0.32200 0.441700

[42,] 11 0.32240 0.402500

[43,] 11 0.32270 0.366700

[44,] 12 0.32290 0.334100

[45,] 12 0.32320 0.304400

[46,] 13 0.32330 0.277400

[47,] 14 0.32370 0.252800

[48,] 16 0.32390 0.230300

[49,] 17 0.32420 0.209800

[50,] 17 0.32440 0.191200

[51,] 18 0.32460 0.174200

[52,] 22 0.32480 0.158700

[53,] 22 0.32500 0.144600

[54,] 23 0.32520 0.131800

[55,] 24 0.32540 0.120100

[56,] 24 0.32550 0.109400

[57,] 26 0.32570 0.099690

[58,] 29 0.32590 0.090830

[59,] 29 0.32600 0.082770

[60,] 30 0.32620 0.075410

[61,] 30 0.32630 0.068710

[62,] 30 0.32640 0.062610

[63,] 31 0.32640 0.057050

[64,] 31 0.32650 0.051980

[65,] 34 0.32660 0.047360

[66,] 35 0.32660 0.043150

[67,] 36 0.32680 0.039320

[68,] 36 0.32700 0.035830

[69,] 37 0.32730 0.032640

[70,] 38 0.32750 0.029740

[71,] 37 0.32770 0.027100

[72,] 40 0.32790 0.024690

[73,] 40 0.32820 0.022500

[74,] 45 0.32840 0.020500

[75,] 45 0.32880 0.018680

[76,] 46 0.32930 0.017020

[77,] 46 0.32960 0.015510

[78,] 47 0.32980 0.014130

[79,] 47 0.33000 0.012880

[80,] 47 0.33010 0.011730

[81,] 47 0.33030 0.010690

[82,] 47 0.33040 0.009740

[83,] 47 0.33050 0.008875

[84,] 47 0.33060 0.008086

[85,] 48 0.33070 0.007368

[86,] 48 0.33080 0.006713

[87,] 48 0.33090 0.006117

[88,] 48 0.33090 0.005574

[89,] 48 0.33100 0.005078

[90,] 48 0.33100 0.004627

[91,] 48 0.33110 0.004216

[92,] 48 0.33110 0.003842

[93,] 49 0.33120 0.003500

[94,] 49 0.33120 0.003189

[95,] 49 0.33120 0.002906

[96,] 49 0.33120 0.002648

[97,] 49 0.33130 0.002413

[98,] 49 0.33130 0.002198

[99,] 49 0.33130 0.002003

[100,] 49 0.33130 0.001825

> # 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

[1] 3.189382

> 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)

54 x 1 sparse Matrix of class "dgCMatrix"

s0

(Intercept) -2.538449e+00

X634995 4.086976e-08

X0 -2.391902e-05

X463 -1.118531e-05

X1 7.735292e-04

X0.0 1.350121e-02

X806.0 1.553809e-03

X11.291044776119403 9.304117e-03

X1.0 1.793778e-04

X70.49513846124168 2.001045e-02

X0.0.1 -1.965736e-02

X806.0.1 -1.021345e-03

X7.574626865671642 2.932718e-02

X0.0.2 1.122924e-01

X69.435826365571 -3.244738e-03

X0.0.3 -7.678915e-03

X76.0 -2.361222e-04

X2.6044776119402986 2.333540e-02

X0.0.4 1.156420e-01

X8.50550186882253 4.524854e-03

X0.0.5 -1.178544e-02

X806.0.2 -1.639273e-03

X10.649253731343284 -3.665694e-03

X1.0.1 -1.572424e-02

X70.25478763764251 5.195732e-03

X.69.0 1.960444e-04

X806.0.3 -5.160122e-04

X4.970149253731344 4.298933e-02

X0.0.6 -3.755897e-02

X69.85058043098057 4.831630e-03

X0.1 3.362289e-03

X0.2 1.151692e-01

X0.3 2.359120e-02

X0.4 5.274079e-04

X0.5 6.427015e-02

X65 -1.912346e-01

X166 2.483775e-04

X2 1.745596e-03

X0.6 .

X24 3.970945e-01

X0.7 -8.263988e-01

X0.8 -4.301798e-01

X0.9 -4.379730e-01

X1.1 7.482821e-01

X0.10 4.288169e-01

X0.11 5.881030e-01

X0.12 -2.262055e-01

X0.13 -5.948819e-01

X0.14 4.781008e-01

X0.15 -7.372765e-02

X0.16 1.043352e+00

X0.17 -1.065425e-01

X0.18 -2.291072e-01

X1.2 -5.230538e-01

>

> # 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)

+ )

RMSE Rsquare

1 34.21296 0.3043655

>

> #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

> cv$lambda.min

[1] 0.7033004

> # Fit the final model on the training data

> model <- glmnet(x, y, alpha = 1, lambda = cv$lambda.min)

> # Dsiplay regression coefficients

> coef(model)

54 x 1 sparse Matrix of class "dgCMatrix"

s0

(Intercept) 5.0667129773

X634995 .

X0 .

X463 .

X1 .

X0.0 .

X806.0 .

X11.291044776119403 .

X1.0 .

X70.49513846124168 0.0131953053

X0.0.1 .

X806.0.1 .

X7.574626865671642 0.0278254674

X0.0.2 0.0955320465

X69.435826365571 .

X0.0.3 .

X76.0 .

X2.6044776119402986 .

X0.0.4 0.1082157028

X8.50550186882253 .

X0.0.5 .

X806.0.2 .

X10.649253731343284 .

X1.0.1 .

X70.25478763764251 .

X.69.0 .

X806.0.3 .

X4.970149253731344 .

X0.0.6 .

X69.85058043098057 .

X0.1 .

X0.2 0.1661874069

X0.3 .

X0.4 .

X0.5 0.0264102499

X65 -0.1704729896

X166 .

X2 0.0007766533

X0.6 .

X24 0.0658390632

X0.7 .

X0.8 .

X0.9 .

X1.1 .

X0.10 .

X0.11 .

X0.12 .

X0.13 .

X0.14 .

X0.15 .

X0.16 .

X0.17 .

X0.18 .

X1.2 .

> # 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))

RMSE Rsquare

1 34.35851 0.2981491

> #Computing elastic net regession

> # Build the model using the training set

> set.seed(123)

> model <- train(X0.19 ~., data = train.data, method = "glmnet",

+ trControl = trainControl("cv", number = 10),

+ tuneLength = 10

+ )

> # Best tuning parameter

> model$bestTun

alpha lambda

96 1 0.5547923

> plot(model$bestTun)

> # Coefficient of the final model. You need

> # to specify the best lambda

> coef(model$finalModel, model$bestTune$lambda)

54 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 3.243771e+00

X634995 .

X0 -1.451030e-06

X463 .

X1 .

X0.0 .

X806.0 .

X11.291044776119403 .

X1.0 .

X70.49513846124168 1.537445e-02

X0.0.1 .

X806.0.1 .

X7.574626865671642 2.330706e-02

X0.0.2 1.020902e-01

X69.435826365571 .

X0.0.3 .

X76.0 .

X2.6044776119402986 .

X0.0.4 1.150414e-01

X8.50550186882253 .

X0.0.5 .

X806.0.2 .

X10.649253731343284 .

X1.0.1 .

X70.25478763764251 .

X.69.0 .

X806.0.3 .

X4.970149253731344 .

X0.0.6 .

X69.85058043098057 .

X0.1 .

X0.2 1.650641e-01

X0.3 .

X0.4 .

X0.5 2.819936e-02

X65 -1.762485e-01

X166 .

X2 9.079099e-04

X0.6 .

X24 1.461588e-01

X0.7 .

X0.8 .

X0.9 .

X1.1 .

X0.10 .

X0.11 .

X0.12 .

X0.13 .

X0.14 .

X0.15 .

X0.16 .

X0.17 .

X0.18 .

X1.2 .

> # Make predictions on the test data

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

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

> # Model performance metrics

> data.frame(

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

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

+ )

RMSE Rsquare

1 34.31238 0.2993523

> #Comparing the different models

> #Using caret package

> #Setup a grid range of lambda values:

> lambda <- 10^seq(-3, 3, length = 100)

> #Compute ridge regression

> # Build the model

> set.seed(123)

> ridge <- train(

+ X0.19 ~., data = train.data, method = "glmnet",

+ trControl = trainControl("cv", number = 10),

+ tuneGrid = expand.grid(alpha = 0, lambda = lambda)

+ )

> # Model coefficients

> coef(ridge$finalModel, ridge$bestTune$lambda)

54 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) -2.780069e+00

X634995 5.350956e-08

X0 -2.364834e-05

X463 -1.329873e-05

X1 6.216835e-04

X0.0 2.038074e-02

X806.0 1.700351e-03

X11.291044776119403 8.550170e-03

X1.0 -4.004515e-03

X70.49513846124168 2.306081e-02

X0.0.1 -5.381912e-02

X806.0.1 -1.655099e-03

X7.574626865671642 2.994944e-02

X0.0.2 1.245636e-01

X69.435826365571 -7.884156e-03

X0.0.3 2.622963e-02

X76.0 -3.747457e-04

X2.6044776119402986 2.978466e-02

X0.0.4 1.358288e-01

X8.50550186882253 3.211969e-03

X0.0.5 -1.596299e-02

X806.0.2 -2.000906e-03

X10.649253731343284 -3.364991e-03

X1.0.1 -1.984096e-02

X70.25478763764251 7.880343e-03

X.69.0 -2.562913e-04

X806.0.3 9.298950e-05

X4.970149253731344 5.279758e-02

X0.0.6 -3.668527e-02

X69.85058043098057 3.578648e-03

X0.1 2.690513e-03

X0.2 1.241846e-01

X0.3 3.087028e-02

X0.4 -5.186126e-03

X0.5 6.542507e-02

X65 -1.911067e-01

X166 2.396281e-04

X2 1.857963e-03

X0.6 .

X24 4.067486e-01

X0.7 -8.764705e-01

X0.8 -4.849690e-01

X0.9 -4.740900e-01

X1.1 7.672004e-01

X0.10 4.799051e-01

X0.11 6.261735e-01

X0.12 -2.236307e-01

X0.13 -6.182879e-01

X0.14 5.103142e-01

X0.15 -3.737265e-02

X0.16 1.097674e+00

X0.17 -1.186700e-01

X0.18 -2.627125e-01

X1.2 -5.621735e-01

> # Make predictions

> predictions <- ridge %>% predict(test.data)

> plot(predictions)

> # Model prediction performance

> data.frame(

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

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

+ )

RMSE Rsquare

1 34.15522 0.3061988

>

> #Compute lasso regression

> # Build the model

> set.seed(123)

> lasso <- train(

+ X0.19 ~., data = train.data, method = "glmnet",

+ trControl = trainControl("cv", number = 10),

+ tuneGrid = expand.grid(alpha = 1, lambda = lambda)

+ )

In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :

There were missing values in resampled performance measures.

> plot( lasso)

> # Model coefficients

> coef(lasso$finalModel, lasso$bestTune$lambda)

54 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 2.988187e+00

X634995 .

X0 -2.504262e-06

X463 .

X1 .

X0.0 .

X806.0 .

X11.291044776119403 .

X1.0 .

X70.49513846124168 1.574129e-02

X0.0.1 .

X806.0.1 .

X7.574626865671642 2.257750e-02

X0.0.2 1.029959e-01

X69.435826365571 .

X0.0.3 .

X76.0 .

X2.6044776119402986 .

X0.0.4 1.160087e-01

X8.50550186882253 .

X0.0.5 .

X806.0.2 .

X10.649253731343284 .

X1.0.1 .

X70.25478763764251 .

X.69.0 .

X806.0.3 .

X4.970149253731344 .

X0.0.6 .

X69.85058043098057 .

X0.1 .

X0.2 1.649295e-01

X0.3 .

X0.4 .

X0.5 2.844279e-02

X65 -1.770593e-01

X166 .

X2 9.269250e-04

X0.6 .

X24 1.574830e-01

X0.7 .

X0.8 .

X0.9 .

X1.1 .

X0.10 .

X0.11 .

X0.12 .

X0.13 .

X0.14 .

X0.15 .

X0.16 .

X0.17 .

X0.18 .

X1.2 .

> # Make predictions

> predictions <- lasso %>% predict(test.data)

> # Model prediction performance

> data.frame(

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

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

+ )

RMSE Rsquare

1 34.3056 0.2995397

> #Elastic net regression

> # Build the model

> set.seed(123)

> elastic <- train(

+ X0.19 ~., data = train.data, method = "glmnet",

+ trControl = trainControl("cv", number = 10),

+ tuneLength = 10

+ )

> # Model coefficients

> coef(elastic$finalModel, elastic$bestTune$lambda)

54 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 3.243771e+00

X634995 .

X0 -1.451030e-06

X463 .

X1 .

X0.0 .

X806.0 .

X11.291044776119403 .

X1.0 .

X70.49513846124168 1.537445e-02

X0.0.1 .

X806.0.1 .

X7.574626865671642 2.330706e-02

X0.0.2 1.020902e-01

X69.435826365571 .

X0.0.3 .

X76.0 .

X2.6044776119402986 .

X0.0.4 1.150414e-01

X8.50550186882253 .

X0.0.5 .

X806.0.2 .

X10.649253731343284 .

X1.0.1 .

X70.25478763764251 .

X.69.0 .

X806.0.3 .

X4.970149253731344 .

X0.0.6 .

X69.85058043098057 .

X0.1 .

X0.2 1.650641e-01

X0.3 .

X0.4 .

X0.5 2.819936e-02

X65 -1.762485e-01

X166 .

X2 9.079099e-04

X0.6 .

X24 1.461588e-01

X0.7 .

X0.8 .

X0.9 .

X1.1 .

X0.10 .

X0.11 .

X0.12 .

X0.13 .

X0.14 .

X0.15 .

X0.16 .

X0.17 .

X0.18 .

X1.2 .

>

> # Make predictions

> predictions <- elastic %>% predict(test.data)

> plot( predictions)

> # Model prediction performance

> data.frame(

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

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

+ )

RMSE Rsquare

1 34.31238 0.2993523

> #Comparing models performance:

> models <- list(ridge = ridge, lasso = lasso, elastic = elastic)

> resamples(models) %>% summary( metric = "RMSE")

Call:

summary.resamples(object = ., metric = "RMSE")

Models: ridge, lasso, elastic

Number of resamples: 10

RMSE

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

ridge 21.64890 25.52171 28.01852 27.99756 30.88915 35.66276 0

lasso 21.52017 25.58002 28.07640 27.93523 30.71819 35.53640 0

elastic 21.51814 25.58351 28.07020 27.93467 30.72364 35.53586 0

>

>

> #k-fold Cross Validation

> # load the library

> library(caret)

>

> # define training control

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

> # fix the parameters of the algorithm

> grid <- expand.grid(.fL=c(0), .usekernel=c(FALSE))

> # train the model

> model <- train(X0.19~., data=Features\_Variant\_1, trControl=train\_control, method=

"nb", tuneGrid=grid)

> # summarize results

> print(model)

glmnet

32760 samples

53 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 29484, 29485, 29484, 29484, 29483, 29484, ...

Resampling results across tuning parameters:

alpha lambda RMSE Rsquared MAE

0.1 0.008432348 28.00932 0.3176368 8.073572

0.1 0.019479818 28.00913 0.3176396 8.072474

0.1 0.045000908 28.00646 0.3176551 8.050171

0.1 0.103957936 28.00846 0.3174455 8.016234

0.1 0.240156321 28.00059 0.3176130 7.975161

0.1 0.554792263 27.98067 0.3181937 7.923823

0.1 1.281642117 27.96815 0.3182085 7.868204

0.1 2.960759592 27.97201 0.3179401 7.737538

0.1 6.839738840 28.04120 0.3164505 7.526685

0.1 15.800684229 28.33270 0.3138865 7.217222

0.2 0.008432348 28.01765 0.3172223 8.072621

0.2 0.019479818 28.01210 0.3174223 8.061405

0.2 0.045000908 28.00976 0.3174643 8.039239

0.2 0.103957936 28.00648 0.3174175 7.996145

0.2 0.240156321 27.98642 0.3180788 7.943594

0.2 0.554792263 27.96948 0.3183579 7.887480

0.2 1.281642117 27.95241 0.3186087 7.775205

0.2 2.960759592 27.96450 0.3182276 7.562917

0.2 6.839738840 28.09348 0.3170458 7.222323

0.2 15.800684229 28.65765 0.3151025 7.133642

0.3 0.008432348 28.00299 0.3177588 8.065477

0.3 0.019479818 28.00437 0.3177326 8.057176

0.3 0.045000908 28.01072 0.3174032 8.031060

0.3 0.103957936 28.00038 0.3176307 7.975466

0.3 0.240156321 27.97652 0.3183974 7.915900

0.3 0.554792263 27.95698 0.3186058 7.846411

0.3 1.281642117 27.94976 0.3185072 7.683206

0.3 2.960759592 27.97534 0.3181303 7.400328

0.3 6.839738840 28.17680 0.3172627 6.985857

0.3 15.800684229 29.13696 0.3103036 7.570747

0.4 0.008432348 28.01830 0.3172446 8.079680

0.4 0.019479818 28.00882 0.3176078 8.058608

0.4 0.045000908 28.01072 0.3173242 8.019561

0.4 0.103957936 27.99402 0.3178502 7.957537

0.4 0.240156321 27.96938 0.3185741 7.892792

0.4 0.554792263 27.94835 0.3187576 7.805517

0.4 1.281642117 27.94390 0.3185641 7.599557

0.4 2.960759592 27.99377 0.3179287 7.243370

0.4 6.839738840 28.28988 0.3165676 6.848175

0.4 15.800684229 29.67198 0.2994897 7.987755

0.5 0.008432348 28.01315 0.3174913 8.081913

0.5 0.019479818 28.00741 0.3176329 8.055975

0.5 0.045000908 28.00761 0.3174288 8.007801

0.5 0.103957936 27.98665 0.3181211 7.942926

0.5 0.240156321 27.96218 0.3187504 7.873342

0.5 0.554792263 27.94327 0.3188512 7.760826

0.5 1.281642117 27.94205 0.3185650 7.519772

0.5 2.960759592 28.02238 0.3174298 7.096167

0.5 6.839738840 28.45004 0.3134711 6.885703

0.5 15.800684229 30.10478 0.2911250 8.323360

0.6 0.008432348 28.01123 0.3175588 8.077084

0.6 0.019479818 28.00535 0.3177547 8.051417

0.6 0.045000908 28.00633 0.3174460 7.995543

0.6 0.103957936 27.98081 0.3183217 7.929786

0.6 0.240156321 27.95596 0.3188777 7.854967

0.6 0.554792263 27.94107 0.3188351 7.718358

0.6 1.281642117 27.94808 0.3183242 7.443956

0.6 2.960759592 28.05422 0.3169109 6.963000

0.6 6.839738840 28.64667 0.3078443 7.060181

0.6 15.800684229 30.45088 0.2909322 8.582636

0.7 0.008432348 28.01327 0.3174691 8.070577

0.7 0.019479818 28.00938 0.3175657 8.046031

0.7 0.045000908 28.00293 0.3175654 7.982522

0.7 0.103957936 27.97610 0.3184735 7.917651

0.7 0.240156321 27.95104 0.3189454 7.835833

0.7 0.554792263 27.94103 0.3187177 7.678114

0.7 1.281642117 27.95454 0.3181156 7.369797

0.7 2.960759592 28.09005 0.3163118 6.846891

0.7 6.839738840 28.85612 0.3001402 7.277639

0.7 15.800684229 30.85446 0.2909322 8.878579

0.8 0.008432348 28.01238 0.3175718 8.070449

0.8 0.019479818 28.00936 0.3175163 8.039224

0.8 0.045000908 28.00020 0.3176370 7.972411

0.8 0.103957936 27.97254 0.3185722 7.906252

0.8 0.240156321 27.94734 0.3189795 7.817664

0.8 0.554792263 27.94076 0.3186304 7.640068

0.8 1.281642117 27.96246 0.3178774 7.295586

0.8 2.960759592 28.13781 0.3151418 6.755044

0.8 6.839738840 29.02499 0.2946305 7.449163

0.8 15.800684229 31.33025 0.2909322 9.216728

0.9 0.008432348 28.01155 0.3175817 8.069278

0.9 0.019479818 28.01019 0.3174714 8.033794

0.9 0.045000908 27.99737 0.3177359 7.963507

0.9 0.103957936 27.96951 0.3186458 7.895521

0.9 0.240156321 27.94458 0.3189922 7.800738

0.9 0.554792263 27.93743 0.3186621 7.602655

0.9 1.281642117 27.97328 0.3175364 7.223312

0.9 2.960759592 28.19981 0.3131783 6.691255

0.9 6.839738840 29.16270 0.2914991 7.572163

0.9 15.800684229 31.88869 0.2909322 9.602631

1.0 0.008432348 28.00984 0.3176179 8.064194

1.0 0.019479818 28.01048 0.3174197 8.027480

1.0 0.045000908 27.99330 0.3178932 7.955851

1.0 0.103957936 27.96647 0.3187201 7.885657

1.0 0.240156321 27.94291 0.3189672 7.784590

1.0 0.554792263 27.93467 0.3186856 7.565986

1.0 1.281642117 27.98635 0.3171131 7.153438

1.0 2.960759592 28.26663 0.3108182 6.661504

1.0 6.839738840 29.27235 0.2909322 7.666803

1.0 15.800684229 32.54609 0.2909322 10.044511

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were alpha = 1 and lambda = 0.5547923.

>

> # load the library

> library(caret)

>

> # define training control

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

> # train the model

> model <- train(X0.19~., data=Features\_Variant\_1, trControl=train\_control, method=

"nb")

> # summarize results

> print(model)

glmnet

32760 samples

53 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 29484, 29485, 29484, 29484, 29483, 29484, ...

Resampling results across tuning parameters:

alpha lambda RMSE Rsquared MAE

0.1 0.008432348 28.00932 0.3176368 8.073572

0.1 0.019479818 28.00913 0.3176396 8.072474

0.1 0.045000908 28.00646 0.3176551 8.050171

0.1 0.103957936 28.00846 0.3174455 8.016234

0.1 0.240156321 28.00059 0.3176130 7.975161

0.1 0.554792263 27.98067 0.3181937 7.923823

0.1 1.281642117 27.96815 0.3182085 7.868204

0.1 2.960759592 27.97201 0.3179401 7.737538

0.1 6.839738840 28.04120 0.3164505 7.526685

0.1 15.800684229 28.33270 0.3138865 7.217222

0.2 0.008432348 28.01765 0.3172223 8.072621

0.2 0.019479818 28.01210 0.3174223 8.061405

0.2 0.045000908 28.00976 0.3174643 8.039239

0.2 0.103957936 28.00648 0.3174175 7.996145

0.2 0.240156321 27.98642 0.3180788 7.943594

0.2 0.554792263 27.96948 0.3183579 7.887480

0.2 1.281642117 27.95241 0.3186087 7.775205

0.2 2.960759592 27.96450 0.3182276 7.562917

0.2 6.839738840 28.09348 0.3170458 7.222323

0.2 15.800684229 28.65765 0.3151025 7.133642

0.3 0.008432348 28.00299 0.3177588 8.065477

0.3 0.019479818 28.00437 0.3177326 8.057176

0.3 0.045000908 28.01072 0.3174032 8.031060

0.3 0.103957936 28.00038 0.3176307 7.975466

0.3 0.240156321 27.97652 0.3183974 7.915900

0.3 0.554792263 27.95698 0.3186058 7.846411

0.3 1.281642117 27.94976 0.3185072 7.683206

0.3 2.960759592 27.97534 0.3181303 7.400328

0.3 6.839738840 28.17680 0.3172627 6.985857

0.3 15.800684229 29.13696 0.3103036 7.570747

0.4 0.008432348 28.01830 0.3172446 8.079680

0.4 0.019479818 28.00882 0.3176078 8.058608

0.4 0.045000908 28.01072 0.3173242 8.019561

0.4 0.103957936 27.99402 0.3178502 7.957537

0.4 0.240156321 27.96938 0.3185741 7.892792

0.4 0.554792263 27.94835 0.3187576 7.805517

0.4 1.281642117 27.94390 0.3185641 7.599557

0.4 2.960759592 27.99377 0.3179287 7.243370

0.4 6.839738840 28.28988 0.3165676 6.848175

0.4 15.800684229 29.67198 0.2994897 7.987755

0.5 0.008432348 28.01315 0.3174913 8.081913

0.5 0.019479818 28.00741 0.3176329 8.055975

0.5 0.045000908 28.00761 0.3174288 8.007801

0.5 0.103957936 27.98665 0.3181211 7.942926

0.5 0.240156321 27.96218 0.3187504 7.873342

0.5 0.554792263 27.94327 0.3188512 7.760826

0.5 1.281642117 27.94205 0.3185650 7.519772

0.5 2.960759592 28.02238 0.3174298 7.096167

0.5 6.839738840 28.45004 0.3134711 6.885703

0.5 15.800684229 30.10478 0.2911250 8.323360

0.6 0.008432348 28.01123 0.3175588 8.077084

0.6 0.019479818 28.00535 0.3177547 8.051417

0.6 0.045000908 28.00633 0.3174460 7.995543

0.6 0.103957936 27.98081 0.3183217 7.929786

0.6 0.240156321 27.95596 0.3188777 7.854967

0.6 0.554792263 27.94107 0.3188351 7.718358

0.6 1.281642117 27.94808 0.3183242 7.443956

0.6 2.960759592 28.05422 0.3169109 6.963000

0.6 6.839738840 28.64667 0.3078443 7.060181

0.6 15.800684229 30.45088 0.2909322 8.582636

0.7 0.008432348 28.01327 0.3174691 8.070577

0.7 0.019479818 28.00938 0.3175657 8.046031

0.7 0.045000908 28.00293 0.3175654 7.982522

0.7 0.103957936 27.97610 0.3184735 7.917651

0.7 0.240156321 27.95104 0.3189454 7.835833

0.7 0.554792263 27.94103 0.3187177 7.678114

0.7 1.281642117 27.95454 0.3181156 7.369797

0.7 2.960759592 28.09005 0.3163118 6.846891

0.7 6.839738840 28.85612 0.3001402 7.277639

0.7 15.800684229 30.85446 0.2909322 8.878579

0.8 0.008432348 28.01238 0.3175718 8.070449

0.8 0.019479818 28.00936 0.3175163 8.039224

0.8 0.045000908 28.00020 0.3176370 7.972411

0.8 0.103957936 27.97254 0.3185722 7.906252

0.8 0.240156321 27.94734 0.3189795 7.817664

0.8 0.554792263 27.94076 0.3186304 7.640068

0.8 1.281642117 27.96246 0.3178774 7.295586

0.8 2.960759592 28.13781 0.3151418 6.755044

0.8 6.839738840 29.02499 0.2946305 7.449163

0.8 15.800684229 31.33025 0.2909322 9.216728

0.9 0.008432348 28.01155 0.3175817 8.069278

0.9 0.019479818 28.01019 0.3174714 8.033794

0.9 0.045000908 27.99737 0.3177359 7.963507

0.9 0.103957936 27.96951 0.3186458 7.895521

0.9 0.240156321 27.94458 0.3189922 7.800738

0.9 0.554792263 27.93743 0.3186621 7.602655

0.9 1.281642117 27.97328 0.3175364 7.223312

0.9 2.960759592 28.19981 0.3131783 6.691255

0.9 6.839738840 29.16270 0.2914991 7.572163

0.9 15.800684229 31.88869 0.2909322 9.602631

1.0 0.008432348 28.00984 0.3176179 8.064194

1.0 0.019479818 28.01048 0.3174197 8.027480

1.0 0.045000908 27.99330 0.3178932 7.955851

1.0 0.103957936 27.96647 0.3187201 7.885657

1.0 0.240156321 27.94291 0.3189672 7.784590

1.0 0.554792263 27.93467 0.3186856 7.565986

1.0 1.281642117 27.98635 0.3171131 7.153438

1.0 2.960759592 28.26663 0.3108182 6.661504

1.0 6.839738840 29.27235 0.2909322 7.666803

1.0 15.800684229 32.54609 0.2909322 10.044511

RMSE was used to select the optimal model using the smallest value.

The final values used for the model were alpha = 1 and lambda = 0.5547923.

>

> #create a graph displaying the accuracy of all models

> plot(model)

> plot(varImp(ridge$finalModel))

> plot(cv)

> plot(ridge)

> hist(Features$X0.19,col = "green")

> hist(Features$X24,col = "red")

> hist(Features$X11.291044776119403,col = 'yellow')

> fit = glmnet(x, y)

> plot(fit)

> cvfit = cv.glmnet(x, y)

> plot(cvfit)

> tfit=glmnet(x,y,lower=-.7,upper=.5)

> plot(tfit)

#compare with linear models and report the accuracy

cor(Features\_Variant\_1$X0.19,Features\_Variant\_1$X24)

mod=lm(Features\_Variant\_1$X0.19~Features\_Variant\_1$X1)

predict(mod)

Features\_Variant\_1$error=mod$residuals

library(car)

dwt(mod)

plot(Features\_Train$X0.19,Features\_Train$X24,

abline(lm(Features\_Variant\_1$X0.19~Features\_Variant\_1$X1),col='red'))

#Assumption1 Linearity

plot(Features\_Variant\_1$X0.19,Features\_Variant\_1$error, xlab="X24",ylab="Residuals",

main="Linearity")

#Assumption - Normality

hist(Features\_Variant\_1$error, xlab = "Residuals",main= "Histogram of Residuals", col="yellow")

#Running Regression

fit<-lm(X0.19~X24+X463+X11.291044776119403+X1.0+X70.49513846124168,

data=Features\_Variant\_1)

fit

#Prediction Accuracy- the one which has good prediction accuracy; in other words, which

#has the smallest prediction error. Consider the simple case of fitting a linear regression

model to the observed data. #A model is a good fit, if it provides a high R2 value.

#Coefficients, Significance of slope, R Square, Model Fit

summary(fit)

#Multicollinearity

vif(fit)

> cor(Features\_Variant\_1$X0.19,Features\_Variant\_1$X24)

[1] 0.01258501

> mod=lm(Features\_Variant\_1$X0.19~Features\_Variant\_1$X1)

> summary(mod)

Call:

lm(formula = Features\_Variant\_1$X0.19 ~ Features\_Variant\_1$X1)

Residuals:

Min 1Q Median 3Q Max

-10.37 -8.27 -6.32 -3.03 1295.68

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 10.502642 0.275383 38.14 <2e-16 \*\*\*

Features\_Variant\_1$X1 -0.131088 0.008768 -14.95 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 35.4 on 40946 degrees of freedom

Multiple R-squared: 0.005429, Adjusted R-squared: 0.005404

F-statistic: 223.5 on 1 and 40946 DF, p-value: < 2.2e-16

> predict(mod)

> Features\_Variant\_1$error=mod$residuals

> library(car)

Loading required package: carData

Attaching package: ‘car’

The following object is masked from ‘package:dplyr’:

recode

The following object is masked from ‘package:purrr’:

some

> dwt(mod)

lag Autocorrelation D-W Statistic p-value

1 0.125323 1.749352 0

Alternative hypothesis: rho != 0

> plot(Features\_Train$X0.19,Features\_Train$X24, abline(lm(Features\_Variant\_1$

X0.19~Features\_Variant\_1$X1),col='red'))

> hist(Features\_Variant\_1$error, xlab = "Residuals",main= "Histogram of Resid

uals", col="yellow")

> plot(Features\_Variant\_1$X0.19,Features\_Variant\_1$error, xlab="X24",ylab="Re

siduals", main="Linearity")

> fit<-lm(X0.19~X24+X463+X11.291044776119403+X1.0+X70.49513846124168, data=Fe

atures\_Variant\_1)

> fit<-lm(X0.19~X24+X463+X11.291044776119403+X1.0+X70.49513846124168, data=Fe

atures\_Variant\_1)

> fit

Call:

lm(formula = X0.19 ~ X24 + X463 + X11.291044776119403 + X1.0 +

X70.49513846124168, data = Features\_Variant\_1)

Coefficients:

(Intercept) X24 X463 X11.2910447761

19403

-1.765e+01 7.229e-01 3.037e-06 6.93

0e-02

X1.0 X70.49513846124168

5.878e-02 2.513e-02

> summary(fit)

Call:

lm(formula = X0.19 ~ X24 + X463 + X11.291044776119403 + X1.0 +

X70.49513846124168, data = Features\_Variant\_1)

Residuals:

Min 1Q Median 3Q Max

-235.02 -5.38 -1.03 0.17 1266.57

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.765e+01 2.079e+00 -8.490 < 2e-16 \*\*\*

X24 7.229e-01 8.654e-02 8.354 < 2e-16 \*\*\*

X463 3.037e-06 1.785e-06 1.701 0.088916 .

X11.291044776119403 6.930e-02 1.775e-02 3.905 9.46e-05 \*\*\*

X1.0 5.878e-02 1.530e-02 3.841 0.000123 \*\*\*

X70.49513846124168 2.513e-02 7.821e-03 3.213 0.001315 \*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 33.41 on 40942 degrees of freedom

Multiple R-squared: 0.1141, Adjusted R-squared: 0.114

F-statistic: 1054 on 5 and 40942 DF, p-value: < 2.2e-16

> vif(fit)

X24 X463 X11.291044776119403 X1

.0

1.012425 1.438330 87.332985 42.0345

25

X70.49513846124168

14.928307

plot(cv$lambda.min)

plot(model)

plot(model$bestTun)

plot(model)

plot(varImp(ridge$finalModel))

plot(cv)

plot(ridge)

hist(Features$X0.19,col = "green")

hist(Features$X24,col = "red")

hist(Features$X11.291044776119403,col = 'yellow')

fit = glmnet(x, y)

plot(fit)

cvfit = cv.glmnet(x, y)

plot(cvfit)

tfit=glmnet(x,y,lower=-.7,upper=.5)

plot(tfit)

plot( lasso)

plot( predictions)

plot(model)

plot(varImp(ridge$finalModel))

plot(cv)

plot(ridge)

fit = glmnet(x, y)

plot(fit)

tfit=glmnet(x,y,lower=-.7,upper=.5)

plot(tfit)