Conducting Ridge and Lasso regression with a dataset

Subhadip Pal

2023-09-21

## Loading the Package and the Dataset

library(olsrr)

##   
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':  
##   
## rivers

data("surgical")  
head(surgical,2)

## bcs pindex enzyme\_test liver\_test age gender alc\_mod alc\_heavy y  
## 1 6.7 62 81 2.59 50 0 1 0 695  
## 2 5.1 59 66 1.70 39 0 0 0 403

## Ridge Regression

## Fitting a Ridge Regression using ( function ) in library(glmnet)

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.0-2

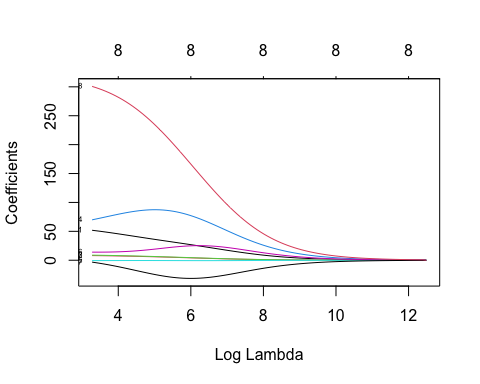
data("surgical") # from the package library(olsrr)  
names(surgical[,1:8])

## [1] "bcs" "pindex" "enzyme\_test" "liver\_test" "age"   
## [6] "gender" "alc\_mod" "alc\_heavy"

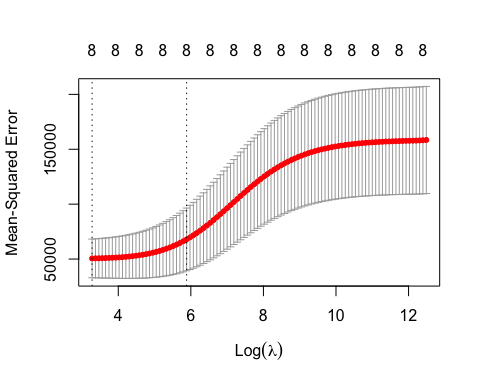
# alpha=0 for fitting a Ridge Regression model  
fit\_ridge<-glmnet(x =as.matrix(surgical[,1:8]) , y =surgical$y, alpha = 0 )  
fit\_ridge

##   
## Call: glmnet(x = as.matrix(surgical[, 1:8]), y = surgical$y, alpha = 0)   
##   
## Df %Dev Lambda  
## 1 8 0.00 265400  
## 2 8 0.43 241800  
## 3 8 0.48 220400  
## 4 8 0.52 200800  
## 5 8 0.57 182900  
## 6 8 0.63 166700  
## 7 8 0.69 151900  
## 8 8 0.75 138400  
## 9 8 0.83 126100  
## 10 8 0.91 114900  
## 11 8 0.99 104700  
## 12 8 1.09 95390  
## 13 8 1.20 86910  
## 14 8 1.31 79190  
## 15 8 1.44 72160  
## 16 8 1.57 65750  
## 17 8 1.72 59910  
## 18 8 1.89 54580  
## 19 8 2.07 49740  
## 20 8 2.27 45320  
## 21 8 2.48 41290  
## 22 8 2.72 37620  
## 23 8 2.97 34280  
## 24 8 3.25 31240  
## 25 8 3.56 28460  
## 26 8 3.89 25930  
## 27 8 4.25 23630  
## 28 8 4.65 21530  
## 29 8 5.08 19620  
## 30 8 5.54 17870  
## 31 8 6.05 16290  
## 32 8 6.60 14840  
## 33 8 7.19 13520  
## 34 8 7.84 12320  
## 35 8 8.53 11230  
## 36 8 9.28 10230  
## 37 8 10.09 9320  
## 38 8 10.96 8492  
## 39 8 11.89 7737  
## 40 8 12.89 7050  
## 41 8 13.96 6424  
## 42 8 15.10 5853  
## 43 8 16.32 5333  
## 44 8 17.61 4859  
## 45 8 18.97 4428  
## 46 8 20.41 4034  
## 47 8 21.93 3676  
## 48 8 23.52 3349  
## 49 8 25.18 3052  
## 50 8 26.91 2781  
## 51 8 28.71 2534  
## 52 8 30.56 2309  
## 53 8 32.47 2103  
## 54 8 34.42 1917  
## 55 8 36.41 1746  
## 56 8 38.43 1591  
## 57 8 40.47 1450  
## 58 8 42.51 1321  
## 59 8 44.55 1204  
## 60 8 46.58 1097  
## 61 8 48.58 999  
## 62 8 50.54 910  
## 63 8 52.46 830  
## 64 8 54.33 756  
## 65 8 56.13 689  
## 66 8 57.87 628  
## 67 8 59.52 572  
## 68 8 61.10 521  
## 69 8 62.59 475  
## 70 8 63.99 433  
## 71 8 65.30 394  
## 72 8 66.53 359  
## 73 8 67.66 327  
## 74 8 68.71 298  
## 75 8 69.68 272  
## 76 8 70.56 248  
## 77 8 71.37 226  
## 78 8 72.11 206  
## 79 8 72.77 187  
## 80 8 73.38 171  
## 81 8 73.92 156  
## 82 8 74.41 142  
## 83 8 74.85 129  
## 84 8 75.24 118  
## 85 8 75.59 107  
## 86 8 75.90 98  
## 87 8 76.18 89  
## 88 8 76.43 81  
## 89 8 76.65 74  
## 90 8 76.84 67  
## 91 8 77.02 61  
## 92 8 77.17 56  
## 93 8 77.30 51  
## 94 8 77.42 46  
## 95 8 77.52 42  
## 96 8 77.61 39  
## 97 8 77.69 35  
## 98 8 77.76 32  
## 99 8 77.82 29  
## 100 8 77.87 27

plot(fit\_ridge, xvar = "lambda", label = TRUE)



# Cross validation to Choose the optimal \Lambda Parameter  
cvfit <- cv.glmnet(x =as.matrix(surgical[,1:8]) , y =surgical$y, alpha=0, nfolds = 10)  
plot(cvfit)



# the optimal value for lambda for which minimum MSE is achieved is cvfit$lambda.min  
paste0("Optimum Lambda=", cvfit$lambda.min)

## [1] "Optimum Lambda=26.5423230926393"

# Fitting the model with optimum value of Lambda  
Ridge\_opt\_Lambda.model <- glmnet(x=as.matrix(surgical[,1:8]), y=surgical$y,  
 alpha = 0,   
 lambda = cvfit$lambda.min)  
# Printing the model output  
Ridge\_opt\_Lambda.model

##   
## Call: glmnet(x = as.matrix(surgical[, 1:8]), y = surgical$y, alpha = 0, lambda = cvfit$lambda.min)   
##   
## Df %Dev Lambda  
## 1 8 77.87 26.54

#Alternatively  
coef(cvfit, s = "lambda.min")

## 9 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1002.4910407  
## bcs 51.8370557  
## pindex 8.0497572  
## enzyme\_test 8.7789615  
## liver\_test 69.9033668  
## age -0.6486542  
## gender 13.9924291  
## alc\_mod -3.2776238  
## alc\_heavy 300.7347029

# Fitting A Lasso model

## Fitting a Lasso Regression using ( function ) in library(glmnet)

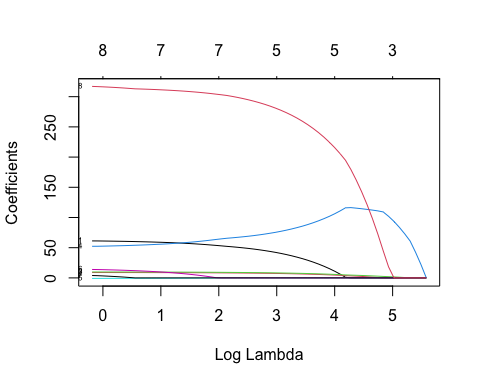
library(glmnet)  
data("surgical") # from the package library(olsrr)  
names(surgical[,1:8])

## [1] "bcs" "pindex" "enzyme\_test" "liver\_test" "age"   
## [6] "gender" "alc\_mod" "alc\_heavy"

# alpha=0 for fitting a Ridge Regression model  
fit\_lasso<-glmnet(x =as.matrix(surgical[,1:8]) , y =surgical$y, alpha = 1 )  
fit\_lasso

##   
## Call: glmnet(x = as.matrix(surgical[, 1:8]), y = surgical$y, alpha = 1)   
##   
## Df %Dev Lambda  
## 1 0 0.00 265.400  
## 2 1 7.72 241.800  
## 3 1 14.12 220.400  
## 4 1 19.44 200.800  
## 5 2 25.67 182.900  
## 6 2 30.84 166.700  
## 7 2 35.14 151.900  
## 8 3 39.79 138.400  
## 9 3 44.22 126.100  
## 10 4 49.36 114.900  
## 11 4 53.72 104.700  
## 12 4 57.33 95.390  
## 13 4 60.33 86.910  
## 14 4 62.83 79.190  
## 15 4 64.90 72.160  
## 16 5 66.72 65.750  
## 17 5 68.65 59.910  
## 18 5 70.25 54.580  
## 19 5 71.58 49.740  
## 20 5 72.68 45.320  
## 21 5 73.60 41.290  
## 22 5 74.36 37.620  
## 23 5 75.00 34.280  
## 24 5 75.52 31.240  
## 25 5 75.96 28.460  
## 26 5 76.32 25.930  
## 27 5 76.62 23.630  
## 28 5 76.87 21.530  
## 29 5 77.08 19.620  
## 30 5 77.25 17.870  
## 31 5 77.39 16.290  
## 32 5 77.51 14.840  
## 33 5 77.61 13.520  
## 34 5 77.69 12.320  
## 35 5 77.76 11.230  
## 36 5 77.81 10.230  
## 37 5 77.86 9.320  
## 38 5 77.90 8.492  
## 39 6 77.94 7.737  
## 40 7 77.98 7.050  
## 41 7 78.01 6.424  
## 42 7 78.04 5.853  
## 43 7 78.06 5.333  
## 44 7 78.08 4.859  
## 45 7 78.10 4.428  
## 46 7 78.11 4.034  
## 47 7 78.12 3.676  
## 48 7 78.13 3.349  
## 49 7 78.14 3.052  
## 50 7 78.15 2.781  
## 51 7 78.15 2.534  
## 52 7 78.16 2.309  
## 53 7 78.16 2.103  
## 54 7 78.16 1.917  
## 55 7 78.16 1.746  
## 56 8 78.17 1.591  
## 57 8 78.17 1.450  
## 58 8 78.17 1.321  
## 59 8 78.17 1.204  
## 60 8 78.18 1.097  
## 61 8 78.18 0.999  
## 62 8 78.18 0.910  
## 63 8 78.18 0.830

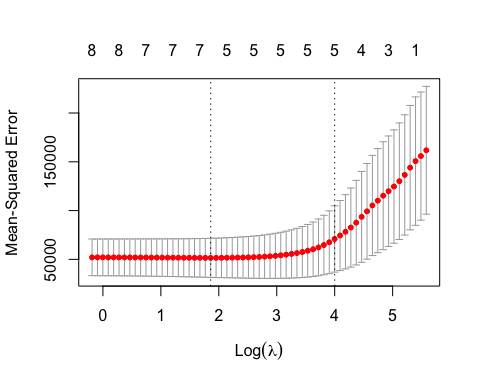
plot(fit\_lasso, xvar = "lambda", label = TRUE)



# Cross validation to Choose the optimal \Lambda Parameter  
cvfit <- cv.glmnet(x =as.matrix(surgical[,1:8]) , y =surgical$y,alpha=1, nfolds = 10)  
  
# the optimal value for lambda for which minimum MSE is achieved is cvfit$lambda.min  
paste0("Optimum Lambda=", cvfit$lambda.min)

## [1] "Optimum Lambda=6.4235826001426"

# Fitting the model with optimum value of Lambda  
plot(cvfit)



Lasso\_opt\_Lambda.model <- glmnet(x=as.matrix(surgical[,1:8]), y=surgical$y,  
 alpha = 1,   
 lambda = cvfit$lambda.min)  
#Printing the model Parameters.  
Lasso\_opt\_Lambda.model

##   
## Call: glmnet(x = as.matrix(surgical[, 1:8]), y = surgical$y, alpha = 1, lambda = cvfit$lambda.min)   
##   
## Df %Dev Lambda  
## 1 7 78.01 6.424

#Alternatively  
coef(cvfit, s = "lambda.min")

## 9 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1081.3106548  
## bcs 54.5057109  
## pindex 8.4290926  
## enzyme\_test 9.2978764  
## liver\_test 62.5433802  
## age -0.2091579  
## gender 1.6907337  
## alc\_mod .   
## alc\_heavy 305.2284325

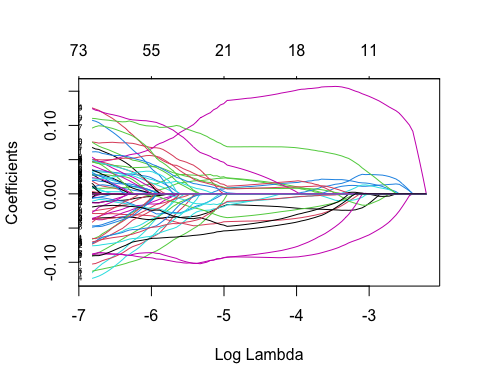
# A Geneset Microarray Data (High)

#library(flare)# for the eyedata  
load(url("https://github.com/subhadippal2019/STAT380UAEU/raw/main/eyedata.Rdata"))

fit\_lasso\_Gene<-glmnet(x =eyeData$X , y =eyeData$y, alpha = 1 )  
fit\_lasso\_Gene

##   
## Call: glmnet(x = eyeData$X, y = eyeData$y, alpha = 1)   
##   
## Df %Dev Lambda  
## 1 0 0.00 0.109400  
## 2 1 5.13 0.104500  
## 3 1 9.81 0.099720  
## 4 1 14.07 0.095190  
## 5 1 17.95 0.090860  
## 6 4 22.26 0.086730  
## 7 4 26.50 0.082790  
## 8 4 30.36 0.079030  
## 9 4 33.88 0.075430  
## 10 8 37.32 0.072010  
## 11 8 40.52 0.068730  
## 12 9 43.45 0.065610  
## 13 9 46.16 0.062630  
## 14 9 48.62 0.059780  
## 15 10 50.88 0.057060  
## 16 10 52.99 0.054470  
## 17 10 54.91 0.051990  
## 18 11 56.67 0.049630  
## 19 12 58.27 0.047380  
## 20 13 59.77 0.045220  
## 21 13 61.27 0.043170  
## 22 13 62.70 0.041200  
## 23 15 64.06 0.039330  
## 24 17 65.34 0.037540  
## 25 17 66.54 0.035840  
## 26 18 67.64 0.034210  
## 27 17 68.64 0.032650  
## 28 17 69.52 0.031170  
## 29 19 70.34 0.029750  
## 30 19 71.07 0.028400  
## 31 20 71.75 0.027110  
## 32 21 72.38 0.025880  
## 33 20 72.96 0.024700  
## 34 19 73.47 0.023580  
## 35 18 73.95 0.022510  
## 36 18 74.37 0.021480  
## 37 18 74.76 0.020510  
## 38 18 75.12 0.019580  
## 39 19 75.45 0.018690  
## 40 18 75.75 0.017840  
## 41 18 76.02 0.017030  
## 42 18 76.28 0.016250  
## 43 18 76.50 0.015510  
## 44 19 76.71 0.014810  
## 45 19 76.91 0.014140  
## 46 19 77.09 0.013490  
## 47 19 77.25 0.012880  
## 48 19 77.40 0.012290  
## 49 19 77.53 0.011740  
## 50 19 77.65 0.011200  
## 51 19 77.77 0.010690  
## 52 19 77.87 0.010210  
## 53 19 77.96 0.009743  
## 54 19 78.04 0.009300  
## 55 19 78.12 0.008877  
## 56 20 78.19 0.008474  
## 57 20 78.26 0.008089  
## 58 20 78.31 0.007721  
## 59 20 78.37 0.007370  
## 60 21 78.47 0.007035  
## 61 21 78.90 0.006715  
## 62 23 79.38 0.006410  
## 63 24 79.90 0.006119  
## 64 25 80.39 0.005841  
## 65 24 80.85 0.005575  
## 66 25 81.25 0.005322  
## 67 25 81.62 0.005080  
## 68 27 81.96 0.004849  
## 69 29 82.37 0.004629  
## 70 30 82.83 0.004418  
## 71 31 83.33 0.004217  
## 72 31 83.78 0.004026  
## 73 32 84.20 0.003843  
## 74 34 84.62 0.003668  
## 75 38 85.13 0.003501  
## 76 40 85.70 0.003342  
## 77 40 86.28 0.003190  
## 78 41 86.82 0.003045  
## 79 42 87.31 0.002907  
## 80 46 87.79 0.002775  
## 81 50 88.26 0.002649  
## 82 52 88.80 0.002528  
## 83 55 89.34 0.002413  
## 84 55 89.92 0.002304  
## 85 54 90.41 0.002199  
## 86 54 90.87 0.002099  
## 87 55 91.29 0.002004  
## 88 57 91.69 0.001913  
## 89 59 92.10 0.001826  
## 90 61 92.54 0.001743  
## 91 62 92.93 0.001663  
## 92 62 93.29 0.001588  
## 93 62 93.62 0.001516  
## 94 64 93.92 0.001447  
## 95 64 94.21 0.001381  
## 96 66 94.48 0.001318  
## 97 69 94.72 0.001258  
## 98 71 94.99 0.001201  
## 99 73 95.28 0.001147  
## 100 73 95.53 0.001094

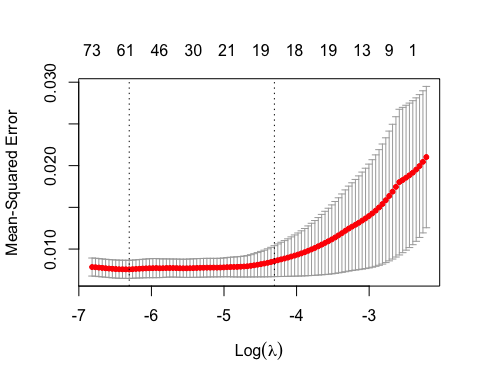
plot(fit\_lasso\_Gene, xvar = "lambda", label = TRUE)



# Cross validation to Choose the optimal \Lambda Parameter  
cvfit <- cv.glmnet(x =eyeData$X , y =eyeData$y,alpha=1, nfolds = 10)  
  
# the optimal value for lambda for which minimum MSE is achieved is cvfit$lambda.min  
paste0("Optimum Lambda=", cvfit$lambda.min)

## [1] "Optimum Lambda=0.00182561773995717"

# Fitting the model with optimum value of Lambda  
  
plot(cvfit)



Lasso\_opt\_Lambda.model <- glmnet(x =eyeData$X , y =eyeData$y,  
 alpha = 1, lambda = cvfit$lambda.min)  
#Printing the model Parameters.  
Lasso\_opt\_Lambda.model$beta

## 200 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## 1377 .   
## 1748 -0.008929719  
## 2487 .   
## 2679 0.003924517  
## 2789 .   
## 2875 .   
## 3244 .   
## 3375 .   
## 3732 .   
## 5892 .   
## 6222 0.009345808  
## 6242 .   
## 6247 0.027812613  
## 6359 .   
## 6690 .   
## 7069 .   
## 7261 .   
## 7941 .   
## 8675 .   
## 8835 .   
## 9061 .   
## 9096 .   
## 9187 .   
## 9303 .   
## 9340 .   
## 9972 .   
## 10144 .   
## 10196 .   
## 10326 .   
## 10438 .   
## 10540 -0.072038679  
## 10693 -0.001421863  
## 10780 .   
## 11024 .   
## 11421 .   
## 11609 .   
## 11711 .   
## 11719 .   
## 11928 .   
## 11995 .   
## 12081 -0.060433859  
## 12085 .   
## 12205 .   
## 12813 .   
## 12997 .   
## 13092 0.027138708  
## 13629 .   
## 13858 .   
## 13901 .   
## 14046 0.073106006  
## 14461 .   
## 14631 .   
## 14903 .   
## 14949 .   
## 15224 0.012907287  
## 15289 .   
## 15368 .   
## 15636 0.012754709  
## 15752 -0.025119287  
## 15787 .   
## 15850 .   
## 15863 -0.029960656  
## 15940 0.023416832  
## 16014 0.006996311  
## 16313 .   
## 16541 -0.050364597  
## 16569 0.004164662  
## 16801 .   
## 16924 .   
## 16964 .   
## 16984 -0.082408182  
## 16988 .   
## 17200 .   
## 17270 .   
## 17436 .   
## 17599 -0.099724106  
## 17645 .   
## 17723 .   
## 17803 .   
## 17816 .   
## 17986 .   
## 18062 .   
## 18283 .   
## 18389 .   
## 18405 .   
## 19331 .   
## 21092 -0.089798848  
## 21094 .   
## 21469 .   
## 21550 -0.033898649  
## 21564 .   
## 21680 -0.031662967  
## 21701 .   
## 21791 .   
## 21864 .   
## 21907 0.051021169  
## 21978 .   
## 22016 .   
## 22029 .   
## 22043 .   
## 22110 .   
## 22140 -0.067899528  
## 22200 0.025725212  
## 22277 .   
## 22304 .   
## 22423 0.012381207  
## 22640 .   
## 22694 -0.015134515  
## 22731 .   
## 22813 -0.016143946  
## 22869 .   
## 22896 .   
## 22935 -0.019920257  
## 22938 0.041763914  
## 22978 .   
## 22980 .   
## 23006 .   
## 23041 .   
## 23050 .   
## 23110 .   
## 23161 .   
## 23206 .   
## 23288 -0.008094970  
## 23348 -0.029396078  
## 23404 .   
## 23618 0.008661633  
## 23804 .   
## 23805 .   
## 23877 .   
## 23942 .   
## 24087 .   
## 24198 .   
## 24225 .   
## 24245 0.092298716  
## 24282 .   
## 24353 .   
## 24396 -0.027312544  
## 24413 .   
## 24422 .   
## 24565 0.111616361  
## 24597 .   
## 24618 .   
## 24653 .   
## 24783 .   
## 24857 0.031755335  
## 24892 0.039859643  
## 24901 0.072758096  
## 25000 .   
## 25014 .   
## 25055 .   
## 25105 .   
## 25109 0.001196989  
## 25141 0.057243530  
## 25281 .   
## 25367 0.052308884  
## 25403 .   
## 25425 0.084929288  
## 25439 .   
## 25443 .   
## 25852 .   
## 25903 0.054294304  
## 25909 .   
## 26369 .   
## 26672 -0.015737044  
## 26696 .   
## 26712 .   
## 26725 .   
## 26738 .   
## 26809 0.019597452  
## 26868 .   
## 26932 -0.033437325  
## 27179 .   
## 27244 -0.031576687  
## 27354 -0.088825100  
## 27408 .   
## 28164 .   
## 28306 .   
## 28343 .   
## 28383 -0.041567160  
## 28680 0.105658775  
## 28738 -0.053416659  
## 28891 .   
## 28899 .   
## 28964 0.034747216  
## 28967 -0.088630240  
## 28983 .   
## 29041 -0.017144052  
## 29045 -0.060738553  
## 29566 .   
## 29665 .   
## 29773 0.009156526  
## 29842 .   
## 29896 .   
## 29912 .   
## 29984 .   
## 30031 0.025267433  
## 30037 .   
## 30078 .   
## 30116 .   
## 30141 -0.072861599

#Alternatively  
coef(cvfit, s = "lambda.min")

## 201 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 7.136283e+00  
## 1377 .   
## 1748 -8.388125e-03  
## 2487 .   
## 2679 2.517005e-03  
## 2789 .   
## 2875 .   
## 3244 .   
## 3375 .   
## 3732 .   
## 5892 .   
## 6222 9.656356e-03  
## 6242 .   
## 6247 2.903650e-02  
## 6359 .   
## 6690 .   
## 7069 .   
## 7261 .   
## 7941 .   
## 8675 .   
## 8835 .   
## 9061 .   
## 9096 .   
## 9187 .   
## 9303 .   
## 9340 .   
## 9972 .   
## 10144 .   
## 10196 .   
## 10326 .   
## 10438 .   
## 10540 -7.321566e-02  
## 10693 -9.669570e-05  
## 10780 .   
## 11024 .   
## 11421 .   
## 11609 .   
## 11711 .   
## 11719 .   
## 11928 .   
## 11995 .   
## 12081 -6.009212e-02  
## 12085 .   
## 12205 .   
## 12813 .   
## 12997 .   
## 13092 2.767179e-02  
## 13629 .   
## 13858 .   
## 13901 .   
## 14046 7.414505e-02  
## 14461 .   
## 14631 .   
## 14903 .   
## 14949 .   
## 15224 1.466523e-02  
## 15289 .   
## 15368 .   
## 15636 1.273523e-02  
## 15752 -2.480796e-02  
## 15787 .   
## 15850 6.129724e-05  
## 15863 -3.016556e-02  
## 15940 2.335782e-02  
## 16014 6.452452e-03  
## 16313 .   
## 16541 -5.021544e-02  
## 16569 2.152853e-03  
## 16801 .   
## 16924 .   
## 16964 .   
## 16984 -7.987130e-02  
## 16988 .   
## 17200 .   
## 17270 .   
## 17436 .   
## 17599 -9.999899e-02  
## 17645 .   
## 17723 .   
## 17803 .   
## 17816 .   
## 17986 .   
## 18062 .   
## 18283 .   
## 18389 .   
## 18405 .   
## 19331 .   
## 21092 -9.007310e-02  
## 21094 .   
## 21469 .   
## 21550 -3.417784e-02  
## 21564 .   
## 21680 -3.382954e-02  
## 21701 .   
## 21791 .   
## 21864 .   
## 21907 5.136857e-02  
## 21978 .   
## 22016 .   
## 22029 .   
## 22043 .   
## 22110 .   
## 22140 -6.615476e-02  
## 22200 2.656443e-02  
## 22277 .   
## 22304 .   
## 22423 1.084368e-02  
## 22640 .   
## 22694 -1.457103e-02  
## 22731 .   
## 22813 -1.347397e-02  
## 22869 .   
## 22896 .   
## 22935 -1.882933e-02  
## 22938 4.214377e-02  
## 22978 .   
## 22980 .   
## 23006 .   
## 23041 .   
## 23050 .   
## 23110 .   
## 23161 .   
## 23206 .   
## 23288 -8.404179e-03  
## 23348 -3.005609e-02  
## 23404 .   
## 23618 1.210533e-02  
## 23804 .   
## 23805 .   
## 23877 .   
## 23942 .   
## 24087 .   
## 24198 .   
## 24225 .   
## 24245 9.020940e-02  
## 24282 .   
## 24353 .   
## 24396 -2.909648e-02  
## 24413 .   
## 24422 .   
## 24565 1.089531e-01  
## 24597 .   
## 24618 .   
## 24653 .   
## 24783 .   
## 24857 3.369137e-02  
## 24892 3.879099e-02  
## 24901 6.920856e-02  
## 25000 .   
## 25014 .   
## 25055 .   
## 25105 .   
## 25109 .   
## 25141 5.699528e-02  
## 25281 .   
## 25367 5.174176e-02  
## 25403 .   
## 25425 8.673643e-02  
## 25439 .   
## 25443 .   
## 25852 .   
## 25903 5.370966e-02  
## 25909 .   
## 26369 .   
## 26672 -1.526488e-02  
## 26696 .   
## 26712 .   
## 26725 .   
## 26738 .   
## 26809 1.927732e-02  
## 26868 .   
## 26932 -3.286326e-02  
## 27179 .   
## 27244 -3.450573e-02  
## 27354 -8.859353e-02  
## 27408 .   
## 28164 .   
## 28306 .   
## 28343 .   
## 28383 -4.251659e-02  
## 28680 1.064465e-01  
## 28738 -5.299277e-02  
## 28891 .   
## 28899 .   
## 28964 3.386027e-02  
## 28967 -8.975478e-02  
## 28983 .   
## 29041 -1.828190e-02  
## 29045 -5.997785e-02  
## 29566 .   
## 29665 .   
## 29773 6.880201e-03  
## 29842 .   
## 29896 .   
## 29912 .   
## 29984 .   
## 30031 2.847425e-02  
## 30037 .   
## 30078 .   
## 30116 .   
## 30141 -7.261278e-02