reg_year_r_random_points

March 2, 2024

0.1 Importing

```
[ ]: import xarray as xr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn import preprocessing

from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import BaggingRegressor

from sklearn.metrics import mean_squared_error as mse

import os

from time import sleep
from tqdm.auto import tqdm

import dill
import random

import salishsea_tools.viz_tools as sa_vi
```

0.2 Datasets Preparation

```
[ ]: def datasets_preparation ():

    ds_name = ('/results2/SalishSea/nowcast-green.202111/' + i + '/
↪SalishSea_1d_' + '20' + str(i[5:7]) + str(dict_month[i[2:5]])+str(i[0:2]) +␣
↪'_' + '20' + str(i[5:7]) + str(dict_month[i[2:5]]) + str(i[0:2]) + '_grid_T.
↪nc')
```

```
    ds_bio_name = ('/results2/SalishSea/nowcast-green.202111/' + i + '/
↪SalishSea_1d_'  + '20' + str(i[5:7]) + str(dict_month[i[2:5]])+str(i[0:2]) +␣
↪'_' + '20' + str(i[5:7]) + str(dict_month[i[2:5]]) + str(i[0:2]) + '_biol_T.
↪nc')

    ds = xr.open_dataset (ds_name)
    ds_bio = xr. open_dataset (ds_bio_name)

    date = pd.DatetimeIndex(ds['time_counter'].values)

    temp_i1 = (ds.votemper.where(mask==1)[0,0:15] * ds.e3t.where(mask==1)
        [0,0:15]).sum('deptht', skipna = True, min_count = 15) / mesh.
↪gdepw_0[0,15]
    temp_i2 = (ds.votemper.where(mask==1)[0,15:27] * ds.e3t.where(mask==1)
        [0,15:27]).sum('deptht', skipna = True, min_count = 12) / (mesh.
↪gdepw_0[0,27] - mesh.gdepw_0[0,14])
    saline_i1 = (ds.vosaline.where(mask==1)[0,0:15] * ds.e3t.where(mask==1)
        [0,0:15]).sum('deptht', skipna = True, min_count = 15) / mesh.
↪gdepw_0[0,15]
    saline_i2 = (ds.vosaline.where(mask==1)[0,15:27] * ds.e3t.where(mask==1)
        [0,15:27]).sum('deptht', skipna = True, min_count = 12) / (mesh.
↪gdepw_0[0,27] - mesh.gdepw_0[0,14])

    diat_i = (ds_bio.diatoms.where(mask==1)[0,0:27] * ds.e3t.where(mask==1)
        [0,0:27]).sum('deptht', skipna = True, min_count = 27) / mesh.
↪gdepw_0[0,27]
    # flag_i = (ds_bio.flagellates.where(mask==1)[0,0:27] * ds.e3t.
↪where(mask==1)
    #       [0,0:27]).sum('deptht', skipna = True, min_count = 27) / mesh.
↪gdepw_0[0,27]

    return (date, temp_i1, temp_i2, saline_i1, saline_i2, diat_i)
```

0.3 Regressor

[ ]:
```
def regressor (inputs, targets):

    inputs = inputs.transpose()

    # Regressor
    scale = preprocessing.StandardScaler()
    inputs = scale.fit_transform(inputs)
    X_train, _, y_train, _ = train_test_split(inputs, targets, train_size=0.35)

    drivers_all = np.array([[],[],[],[]])
```

```
    diat_all = np.array([])
    inputs = np.array([[],[],[],[]])
    targets = np.array([])

    model = MLPRegressor(hidden_layer_sizes=200, alpha=0.002)
    regr = BaggingRegressor(model, n_estimators=12, n_jobs=4).fit(X_train,␣
↪y_train)

    return (regr)
```

1 Printing

[ ]:
```
def printing (targets, outputs, m):

    print ('The amount of data points is', outputs.size)
    print ('The slope of the best fitting line is ', np.round(m,3))
    print ('The correlation coefficient is:', np.round(np.corrcoef(targets,␣
↪outputs)[0][1],3))
    print (' The mean square error is:', np.round(mse(targets,outputs),5))
```

1.1 Scatter Plot

[ ]:
```
def scatter_plot(targets, outputs, variable_name):

    # compute slope m and intercept b
    m, b = np.polyfit(targets, outputs, deg=1)

    printing(targets, outputs, m)

    fig, ax = plt.subplots(2, figsize=(5,10), layout='constrained')

    ax[0].scatter(targets,outputs, alpha = 0.2, s = 10)

    lims = [np.min([ax[0].get_xlim(), ax[0].get_ylim()]),
        np.max([ax[0].get_xlim(), ax[0].get_ylim()]),]

    # plot fitted y = m*x + b
    ax[0].axline(xy1=(0, b), slope=m, color='r')

    ax[0].set_xlabel('targets')
    ax[0].set_ylabel('outputs')
    ax[0].set_xlim(lims)
    ax[0].set_ylim(lims)
    ax[0].set_aspect('equal')
```

```
    ax[0].plot(lims, lims,linestyle = '--',color = 'k')

    h = ax[1].hist2d(targets,outputs, bins=100, cmap='jet',
        range=[lims,lims], cmin=0.1, norm='log')

    ax[1].plot(lims, lims,linestyle = '--',color = 'k')

    # plot fitted y = m*x + b
    ax[1].axline(xy1=(0, b), slope=m, color='r')

    ax[1].set_xlabel('targets')
    ax[1].set_ylabel('outputs')
    ax[1].set_aspect('equal')

    fig.colorbar(h[3],ax=ax[1], location='bottom')

    fig.suptitle(variable_name)

    plt.show()

    return (m)
```

1.2 Plotting

[ ]:
```
def plotting(variable, name):

    plt.plot(years,variable, marker = '.', linestyle = '')
    plt.legend(['diatom','flagellate'])
    plt.xlabel('Years')
    plt.ylabel(name)
    plt.show()
```

1.3 Plotting 2

[ ]:
```
def plotting2(variable,title):

    fig, ax = plt.subplots()

    scatter= ax.scatter(dates,variable, marker='.', c=pd.DatetimeIndex(dates).
 ↪month)

    ax.legend(handles=scatter.legend_elements()[0],␣
 ↪labels=['February','March','April'])
```

```
    fig.suptitle('Daily ' + title + ' (15 Feb - 30 Apr)')

    fig.show()
```

1.4 Plotting 3

```
[ ]: def plotting3(targets, model, variable, variable_name):

    fig, ax = plt.subplots(2,2, figsize = (10,15))

    cmap = plt.get_cmap('cubehelix')
    cmap.set_bad('gray')

    variable.plot(ax=ax[0,0], cmap=cmap, vmin = targets.min(), vmax =targets.
 ↪max(), cbar_kwargs={'label': variable_name + ' Concentration  [mmol m-2]'})
    model.plot(ax=ax[0,1], cmap=cmap, vmin = targets.min(), vmax = targets.
 ↪max(), cbar_kwargs={'label': variable_name + ' Concentration  [mmol m-2]'})
    ((variable-model) / variable * 100).plot(ax=ax[1,0], cmap=cmap,␣
 ↪cbar_kwargs={'label': variable_name + ' Concentration  [percentage]'})

    plt.subplots_adjust(left=0.1,
        bottom=0.1,
        right=0.95,
        top=0.95,
        wspace=0.35,
        hspace=0.35)

    sa_vi.set_aspect(ax[0,0])
    sa_vi.set_aspect(ax[0,1])
    sa_vi.set_aspect(ax[1,0])


    ax[0,0].title.set_text(variable_name + ' (targets)')
    ax[0,1].title.set_text(variable_name + ' (outputs)')
    ax[1,0].title.set_text('targets - outputs')
    ax[1,1].axis('off')

    fig.suptitle(str(date.date[0]))

    plt.show()
```

1.5 Regressor 2

```
[ ]: def regressor2 (inputs, targets, variable_name):

         inputs = inputs.transpose()

         # Regressor
         scale = preprocessing.StandardScaler()
         inputs2 = scale.fit_transform(inputs)

         outputs_test = regr.predict(inputs2)

         m = scatter_plot(targets, outputs_test, variable_name)
         r = np.round(np.corrcoef(targets, outputs_test)[0][1],3)
         rms = np.round(mse(targets, outputs_test),4)

         return (r, rms, m)
```

1.6 Regressor 3

```
[ ]: def regressor3 (inputs, targets, variable_name):

         inputs = inputs.transpose()

         # Regressor
         scale = preprocessing.StandardScaler()
         inputs2 = scale.fit_transform(inputs)

         outputs = regr.predict(inputs2)

         # Post processing
         indx2 = np.full((898*398),np.nan)
         indx2[indx[0]] = outputs
         model = np.reshape(indx2,(898,398))

         m = scatter_plot(targets, outputs, variable_name + ' (Testing dataset)')

         # Preparation of the dataarray
         model = xr.DataArray(model,
             coords = {'y': diat_i.y, 'x': diat_i.x},
             dims = ['y','x'],
             attrs=dict( long_name = variable_name + " Concentration",
             units="mmol m-2"),)

         plotting3(targets, model, diat_i, variable_name)
```

1.7 Regressor 4

[ ]:
```
def regressor4 (inputs, targets, variable_name):

    inputs = inputs.transpose()

    # Regressor
    scale = preprocessing.StandardScaler()
    inputs2 = scale.fit_transform(inputs)

    outputs_test = regr.predict(inputs2)

    # compute slope m and intercept b
    m, b = np.polyfit(targets, outputs_test, deg=1)

    r = np.round(np.corrcoef(targets, outputs_test)[0][1],3)
    rms = np.round(mse(targets, outputs_test),4)

    return (r, rms, m)
```

1.8 Training (Random Points)

[ ]:
```
dict_month = {'jan': '01',
         'feb': '02',
         'mar': '03',
         'apr': '04',
         'may': '05',
         'jun': '06',
         'jul': '07',
         'aug': '08',
         'sep': '09',
         'oct': '10',
         'nov': '11',
         'dec': '12'}

path = os.listdir('/results2/SalishSea/nowcast-green.202111/')

# Open the mesh mask
mesh = xr.open_dataset('/home/sallen/MEOPAR/grid/mesh_mask202108.nc')
mask = mesh.tmask.to_numpy()

folders = [x for x in path if ((x[2:5]=='mar' or x[2:5]=='apr' or (x[2:
 ↪5]=='feb' and x[0:2]>'14')) and (x[5:7]<'24'))]
indx_dates=(np.argsort(pd.to_datetime(folders, format="%d%b%y")))
folders = [folders[i] for i in indx_dates]
```

```
drivers_all = np.array([[],[],[],[]])
diat_all = np.array([])

print ('Gathering days for the model')

for i in tqdm(folders):

    date, temp_i1, temp_i2, saline_i1, saline_i2, diat_i = 
 ↪datasets_preparation()

    drivers = np.stack([np.ravel(temp_i1), np.ravel(temp_i2), np.
 ↪ravel(saline_i1), np.ravel(saline_i2)])
    indx = np.where(~np.isnan(drivers).any(axis=0))
    drivers = drivers[:,indx[0]]
    drivers_all = np.concatenate((drivers_all,drivers),axis=1)

    diat = np.ravel(diat_i)
    diat = diat[indx[0]]
    diat_all = np.concatenate((diat_all,diat))

    sleep(0.1)

print ('Done gathering, building the prediction model')
print ('\n')

regr = regressor(drivers_all, diat_all)
```

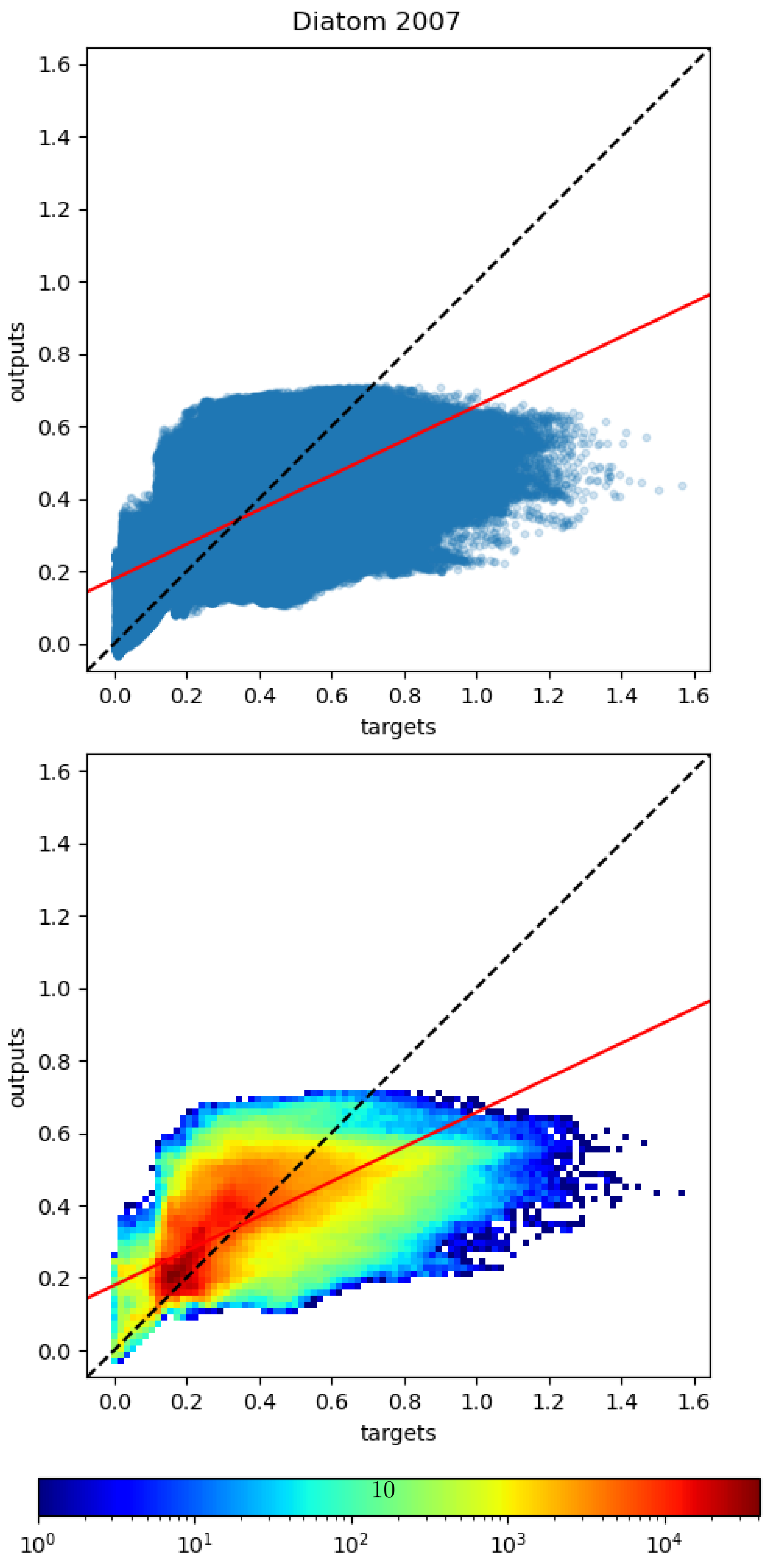
```
Gathering days for the model

  0%|          | 0/1279 [00:00<?, ?it/s]

Done gathering, building the prediction model
```

1.9 Other Years (Anually)

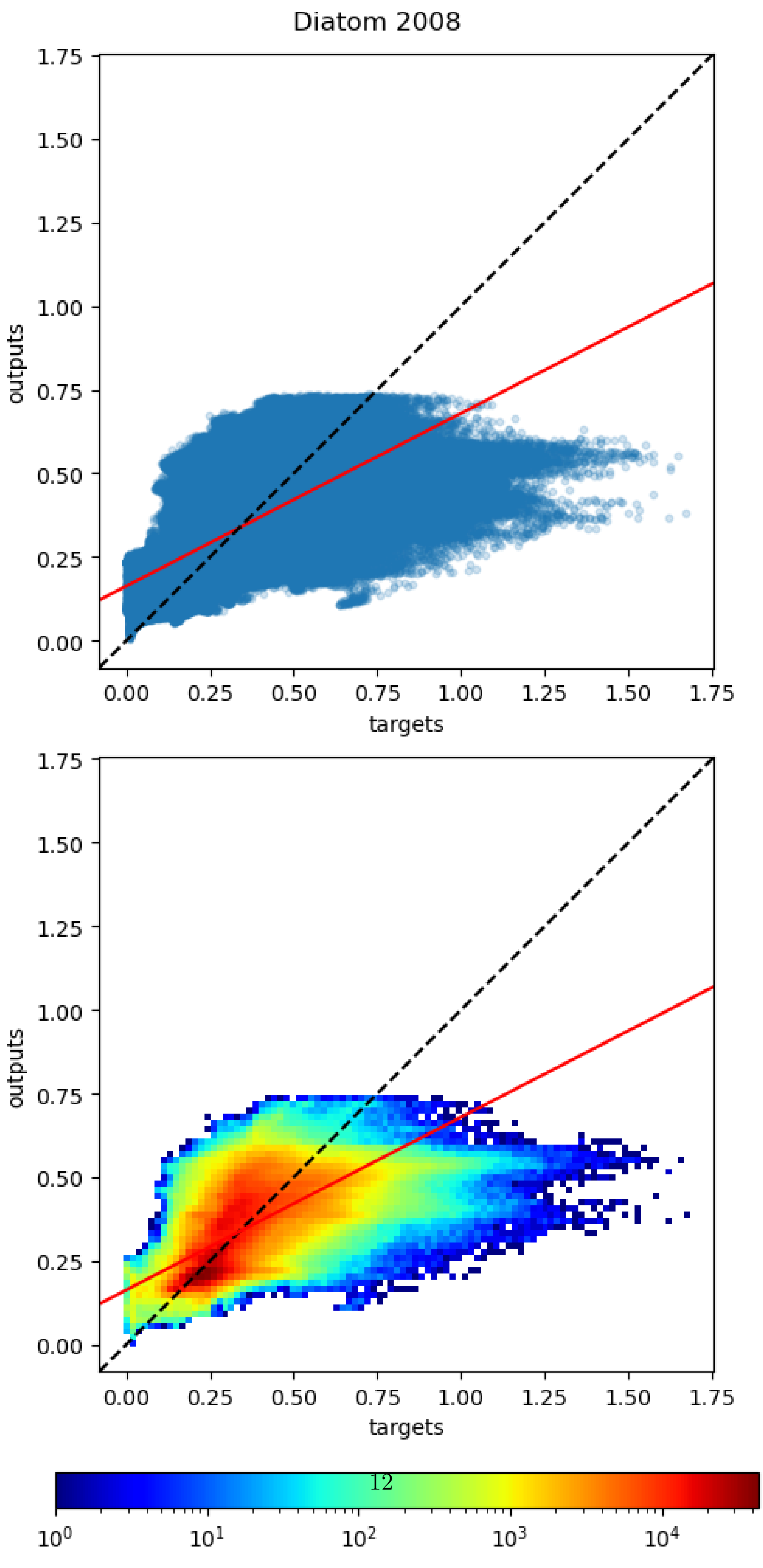
[ ]:
```
years = range (2007,2024)

r_all = []
rms_all = []
slope_all = []

for year in range (2007,2024):

    year_str = str(year)[2:4]
```

```
    folders = [x for x in path if ((x[2:5]=='mar' or x[2:5]=='apr' or (x[2:
↪5]=='feb' and x[0:2] > '14')) and (x[5:7]==year_str))]
    indx_dates=(np.argsort(pd.to_datetime(folders, format="%d%b%y")))
    folders = [folders[i] for i in indx_dates]

    drivers_all = np.array([[],[],[],[]])
    diat_all = np.array([])

    print ('Gathering days for year ' + str(year))
    for i in tqdm(folders):

        date, temp_i1, temp_i2, saline_i1, saline_i2, diat_i =␣
↪datasets_preparation()

        drivers = np.stack([np.ravel(temp_i1), np.ravel(temp_i2), np.
↪ravel(saline_i1), np.ravel(saline_i2)])
        indx = np.where(~np.isnan(drivers).any(axis=0))
        drivers = drivers[:,indx[0]]
        drivers_all = np.concatenate((drivers_all,drivers),axis=1)

        diat = np.ravel(diat_i)
        diat = diat[indx[0]]
        diat_all = np.concatenate((diat_all,diat))

    r, rms, m = regressor2(drivers_all, diat_all, 'Diatom ' + str(year))

    r_all.append(r)
    rms_all.append(rms)
    slope_all.append(m)

plotting(np.transpose(r_all), 'Correlation Coefficient')
plotting(np.transpose(rms_all), 'Mean Square Error')
plotting (np.transpose(slope_all), 'Slope of the best fitting line')
```

```
Gathering days for year 2007

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.477
The correlation coefficient is: 0.66
 The mean square error is: 0.01512
```

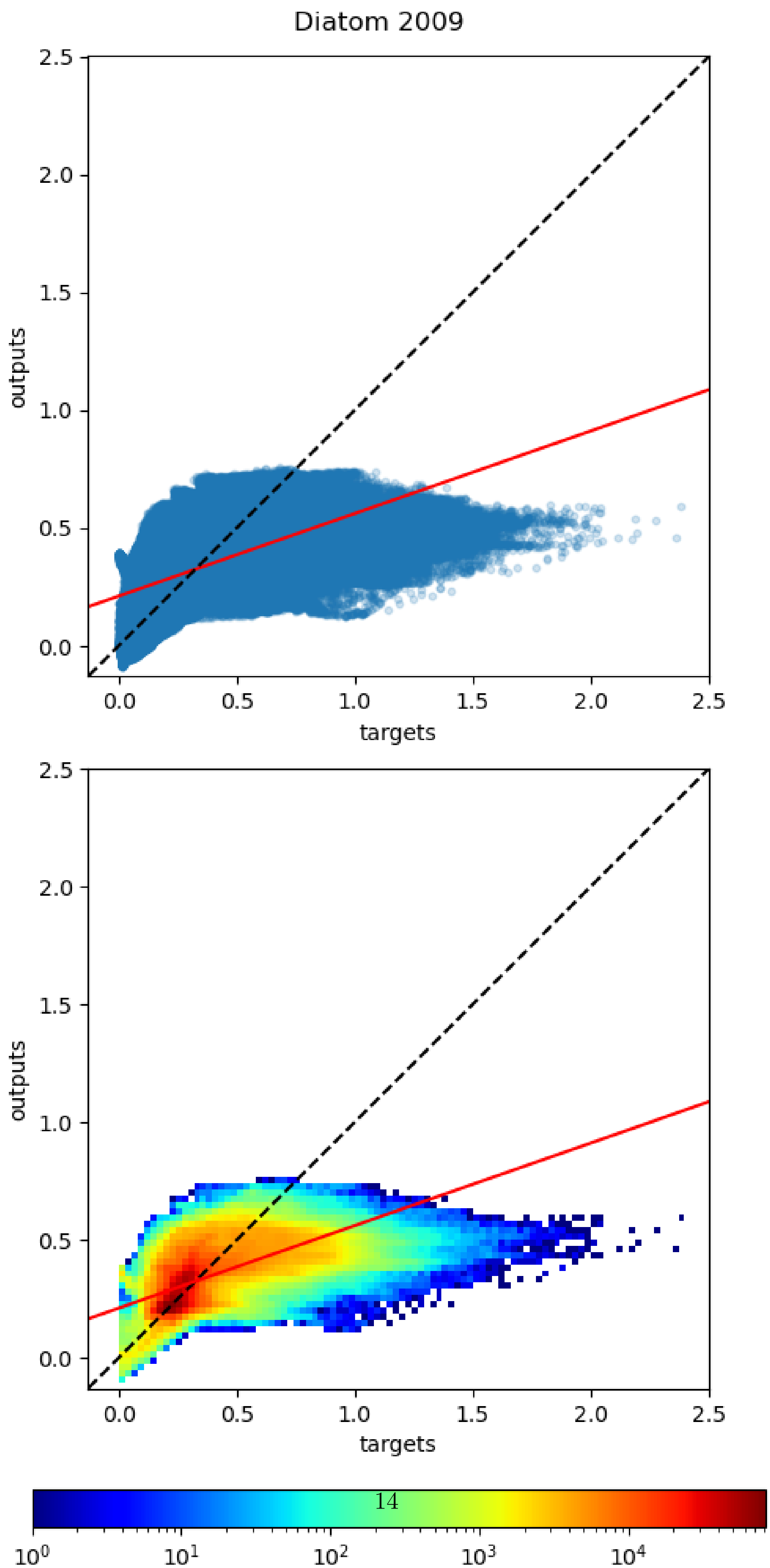

Diatom 2007
outputs
targets
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
10
10^0
10^1
10^2
10^3
10^4

```
Gathering days for year 2008

  0%|          | 0/76 [00:00<?, ?it/s]

The amount of data points is 3532404
The slope of the best fitting line is  0.516
The correlation coefficient is: 0.648
 The mean square error is: 0.01268
```

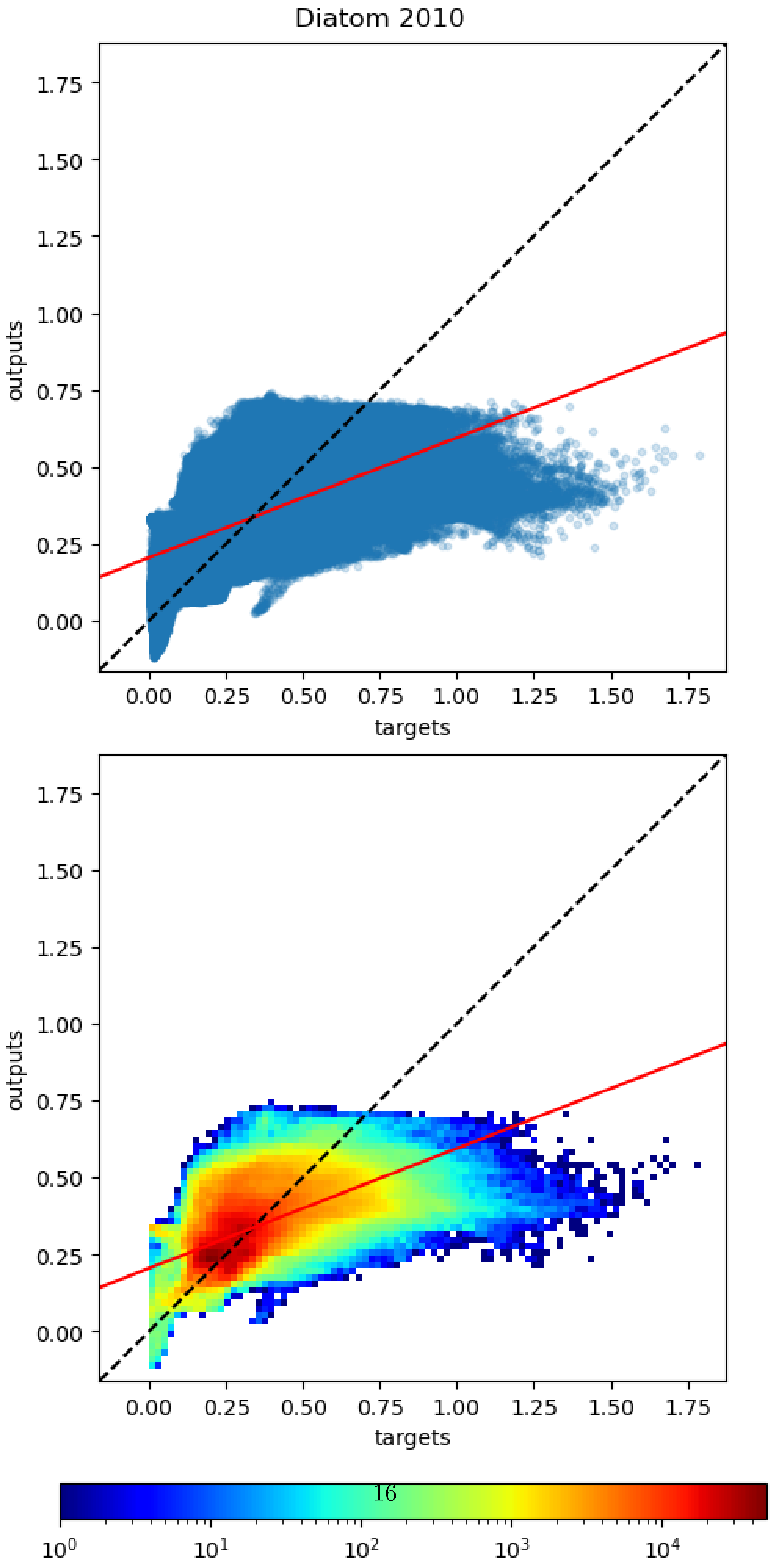


Diatom 2008
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^0
10^1
10^2
10^3
10^4

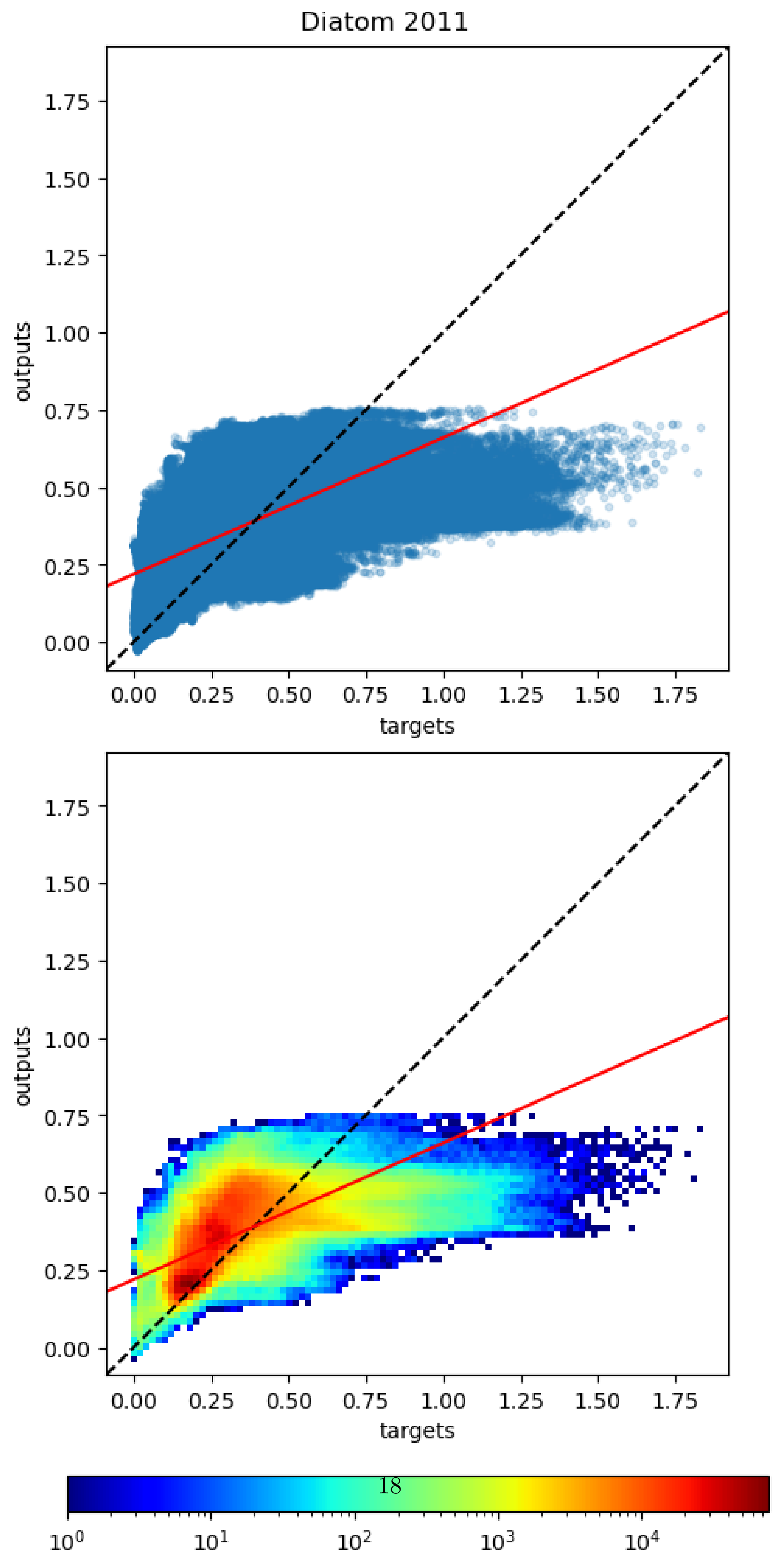
```
Gathering days for year 2009

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.35
The correlation coefficient is: 0.601
 The mean square error is: 0.02485
```


Diatom 2009
outputs
targets
0.0
0.5
1.0
1.5
2.0
2.5
10^0
10^1
10^2
10^3
10^4

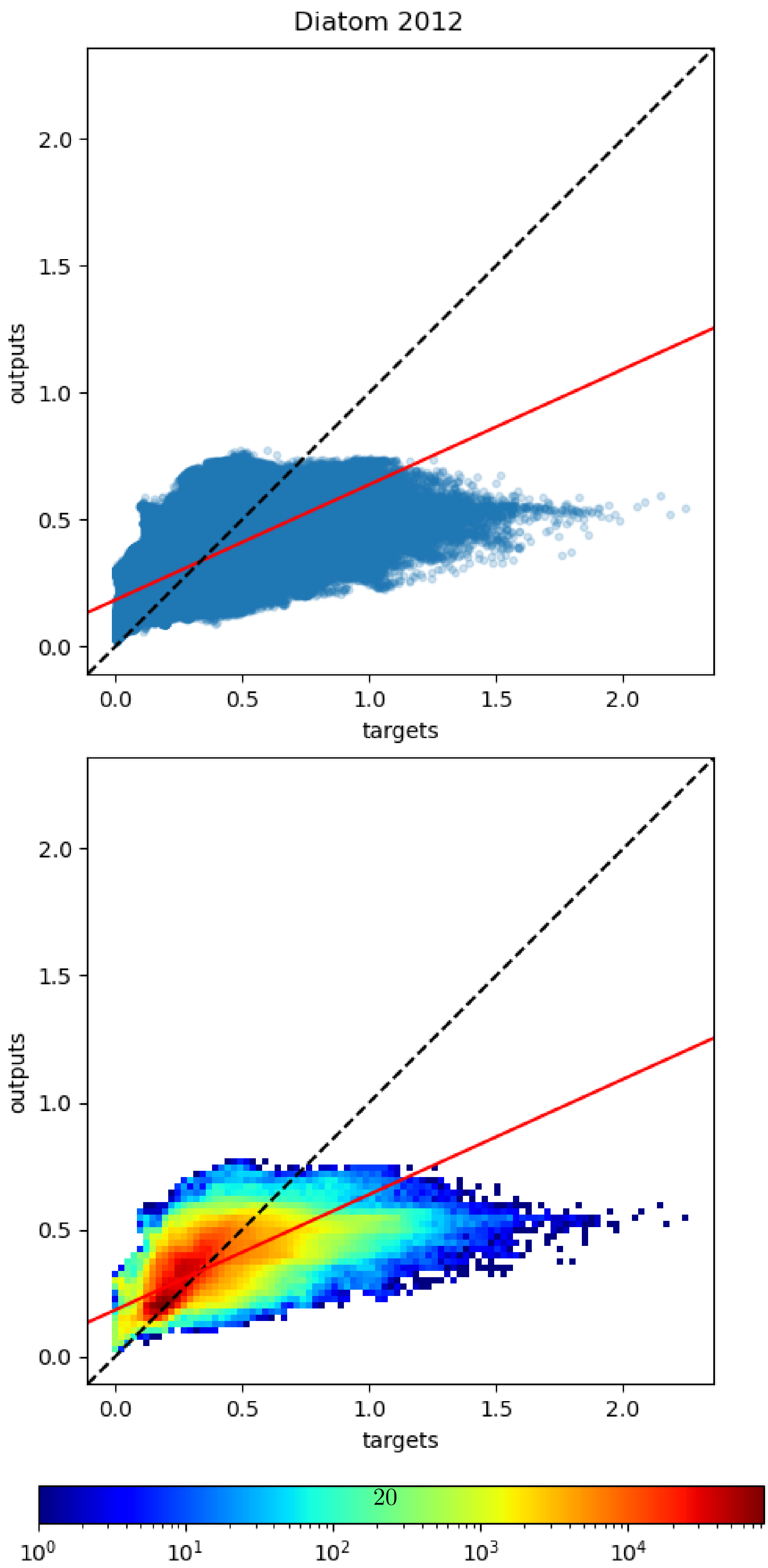
```
Gathering days for year 2010

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.389
The correlation coefficient is: 0.562
 The mean square error is: 0.015
```


Diatom 2010
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
outputs
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75
targets
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
outputs
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75
targets
10^0 10^1 10^2 10^3 10^4

```
Gathering days for year 2011

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.441
The correlation coefficient is: 0.611
 The mean square error is: 0.01821
```

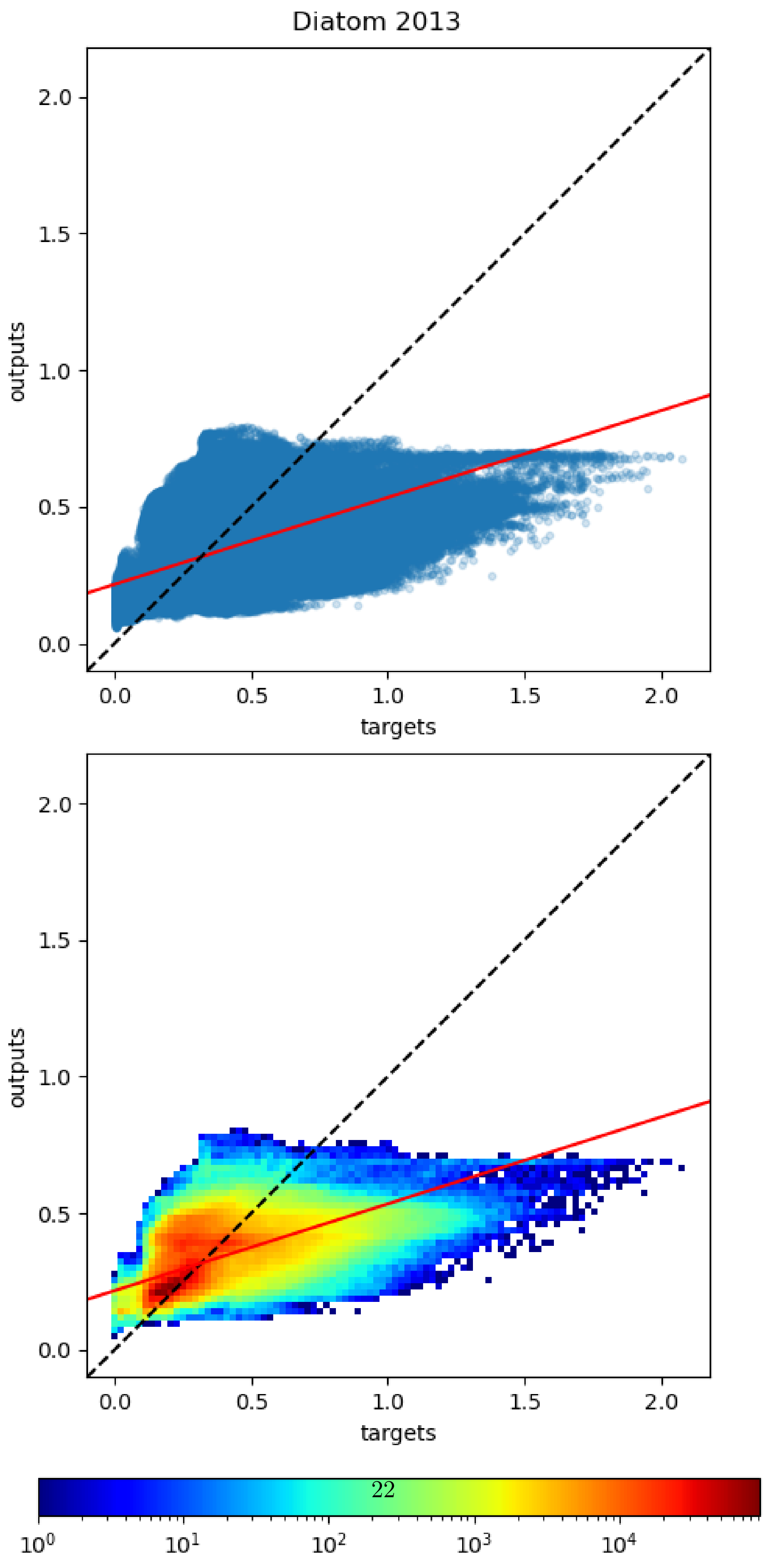

Diatom 2011
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^0
10^1
10^2
10^3
10^4

```
Gathering days for year 2012

  0%|          | 0/76 [00:00<?, ?it/s]

The amount of data points is 3532404
The slope of the best fitting line is  0.454
The correlation coefficient is: 0.683
 The mean square error is: 0.0135
```

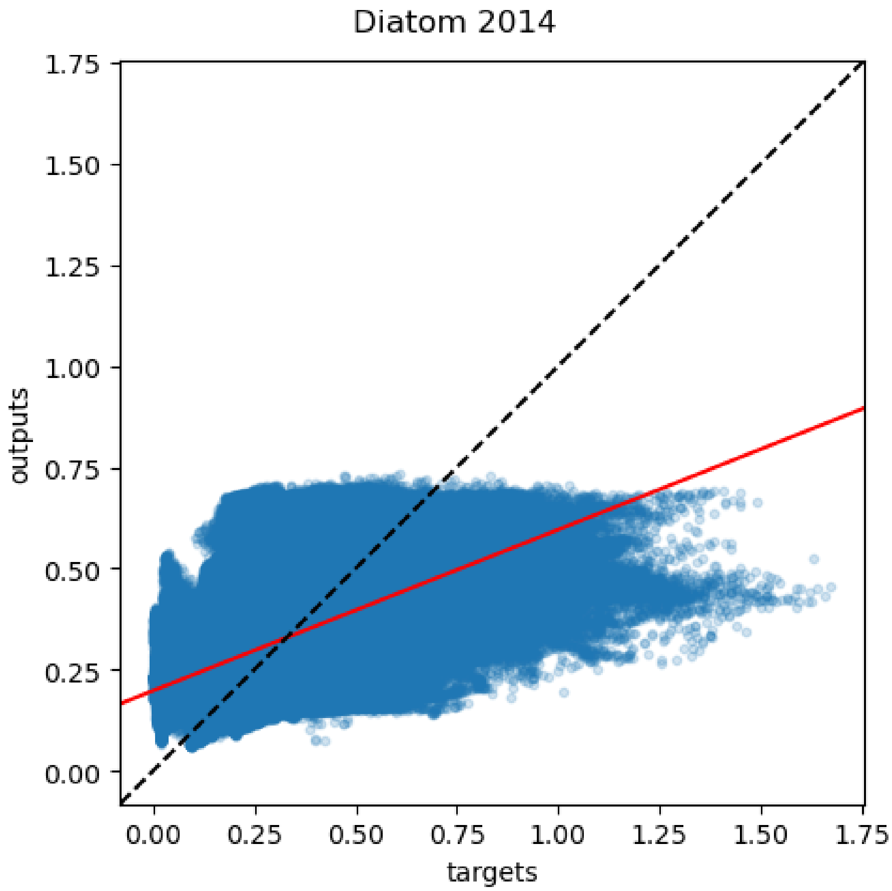


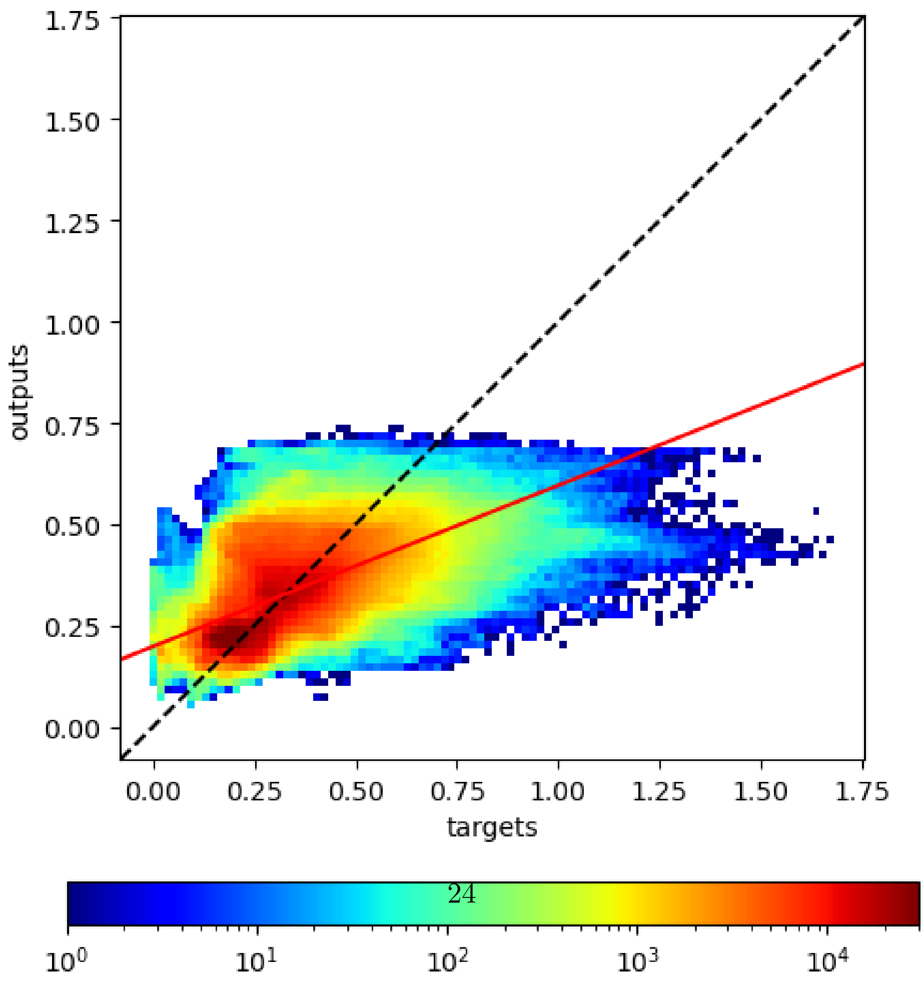
Diatom 2012
outputs
targets
0.0
0.5
1.0
1.5
2.0
outputs
targets
0.0
0.5
1.0
1.5
2.0
10^0
10^1
10^2
10^3
10^4

```
Gathering days for year 2013

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.318
The correlation coefficient is: 0.574
 The mean square error is: 0.0213
```

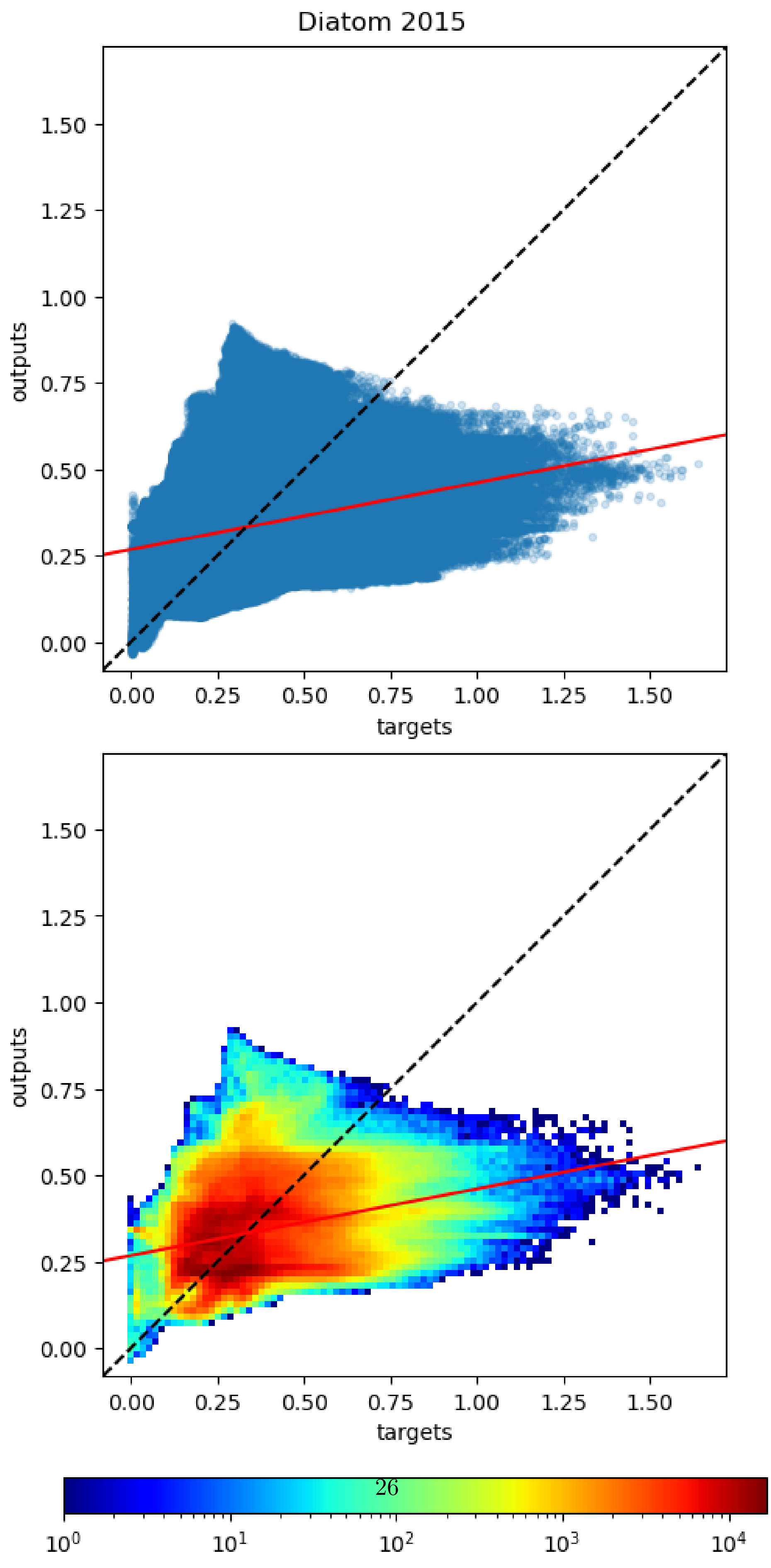

Diatom 2013
outputs
targets
0.0
0.5
1.0
1.5
2.0
10^0
10^1
10^2
10^3
10^4

```
Gathering days for year 2014

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.398
The correlation coefficient is: 0.572
 The mean square error is: 0.01449
```

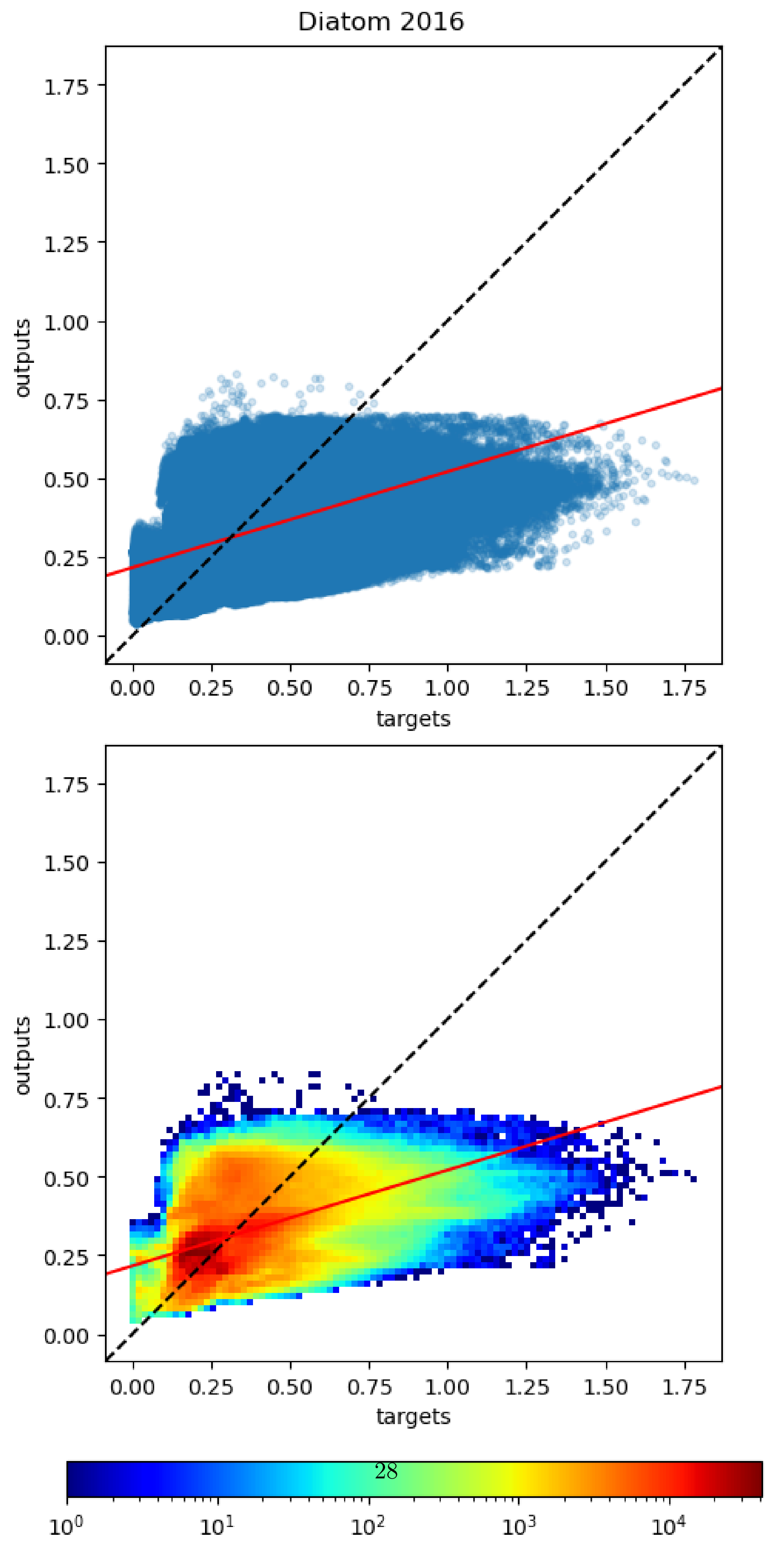

Diatom 2014
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75


outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^0
10^1
10^2
10^3
10^4

```
Gathering days for year 2015

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.193
The correlation coefficient is: 0.262
 The mean square error is: 0.02695
```

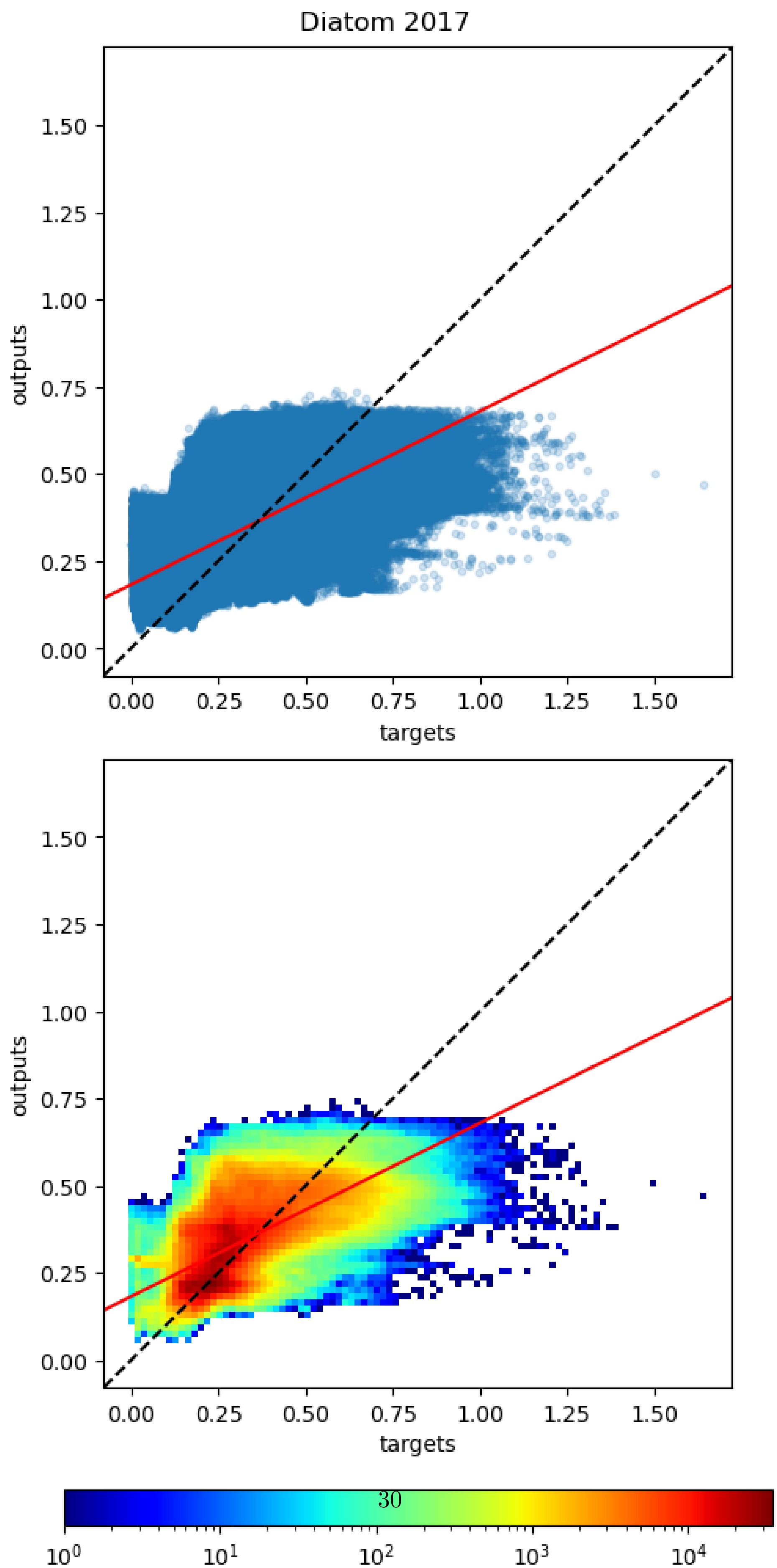

Diatom 2015
outputs
targets
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50
outputs
targets
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50
10^0 10^1 10^2 10^3 10^4

```
Gathering days for year 2016

  0%|          | 0/76 [00:00<?, ?it/s]

The amount of data points is 3532404
The slope of the best fitting line is  0.304
The correlation coefficient is: 0.45
 The mean square error is: 0.02441
```

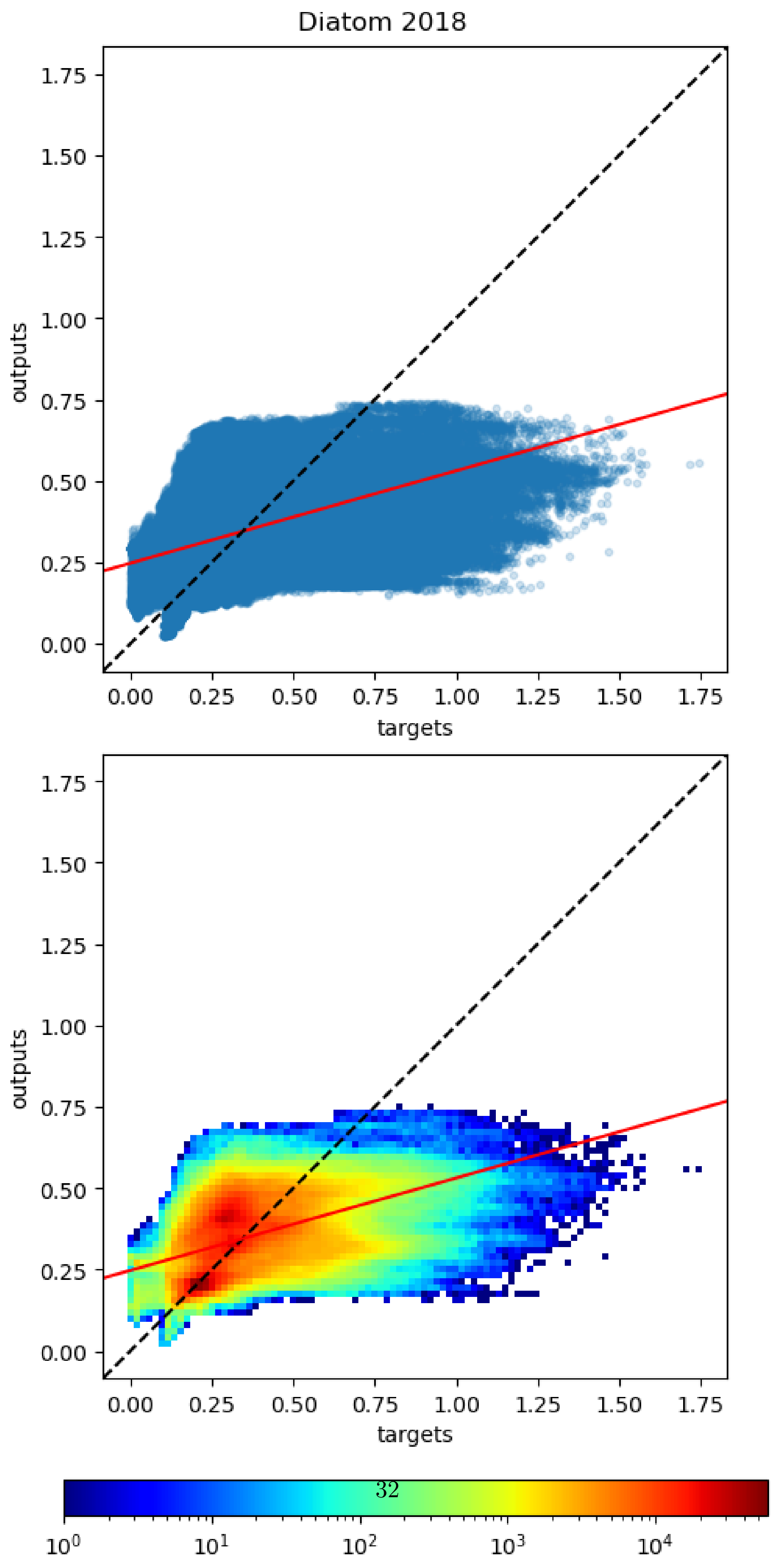


Diatom 2016
outputs
targets
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75
10^0 10^1 10^2 10^3 10^4

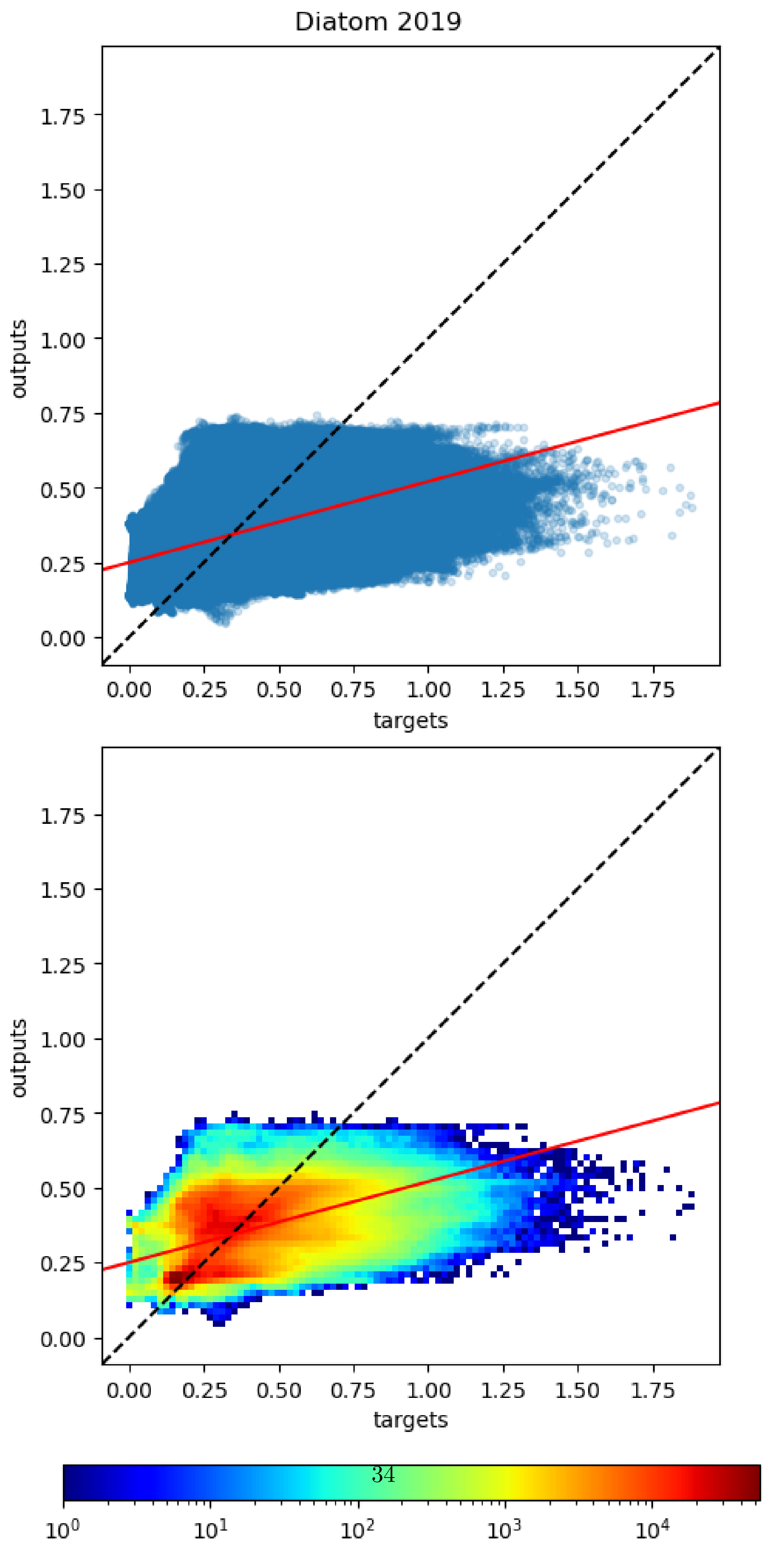
```
Gathering days for year 2017

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.498
The correlation coefficient is: 0.619
 The mean square error is: 0.01253
```


Diatom 2017
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
10^0
10^1
10^2
10^3
10^4

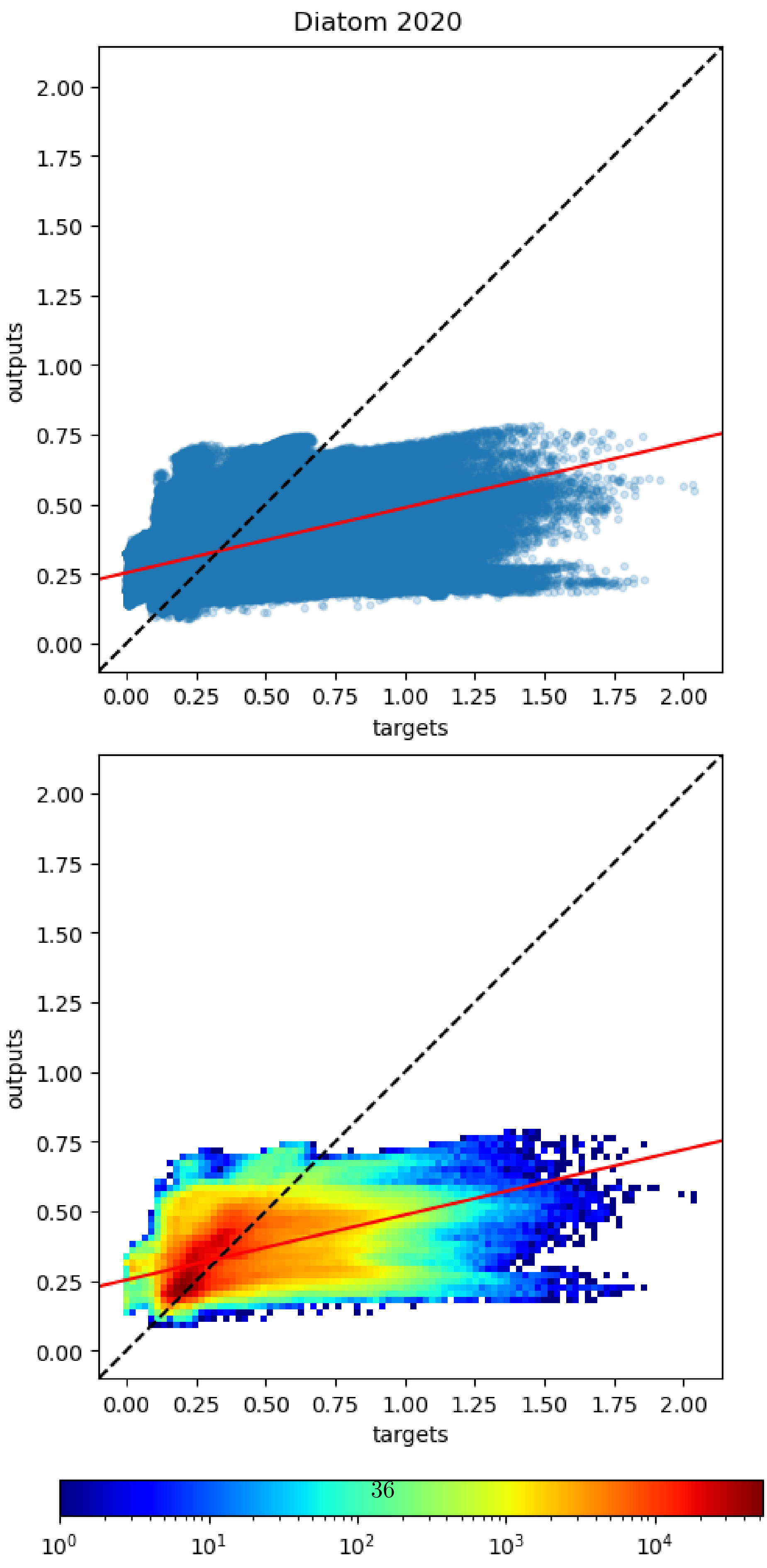
```
Gathering days for year 2018

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.284
The correlation coefficient is: 0.431
 The mean square error is: 0.02127
```


Diatom 2018
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^0
10^1
10^2
10^3
10^4

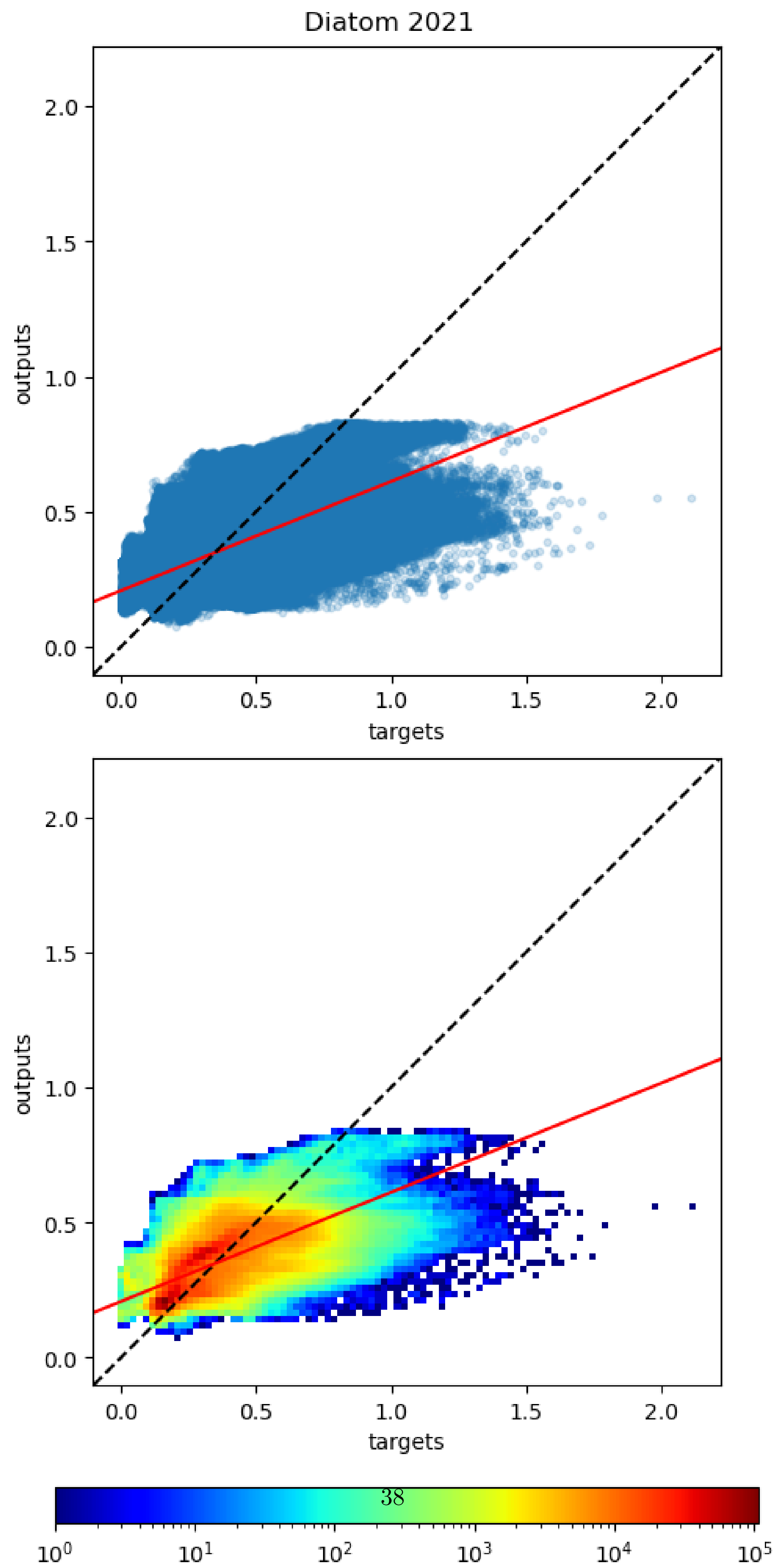
```
Gathering days for year 2019

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.27
The correlation coefficient is: 0.451
 The mean square error is: 0.02517
```


Diatom 2019
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^0
10^1
10^2
10^3
10^4

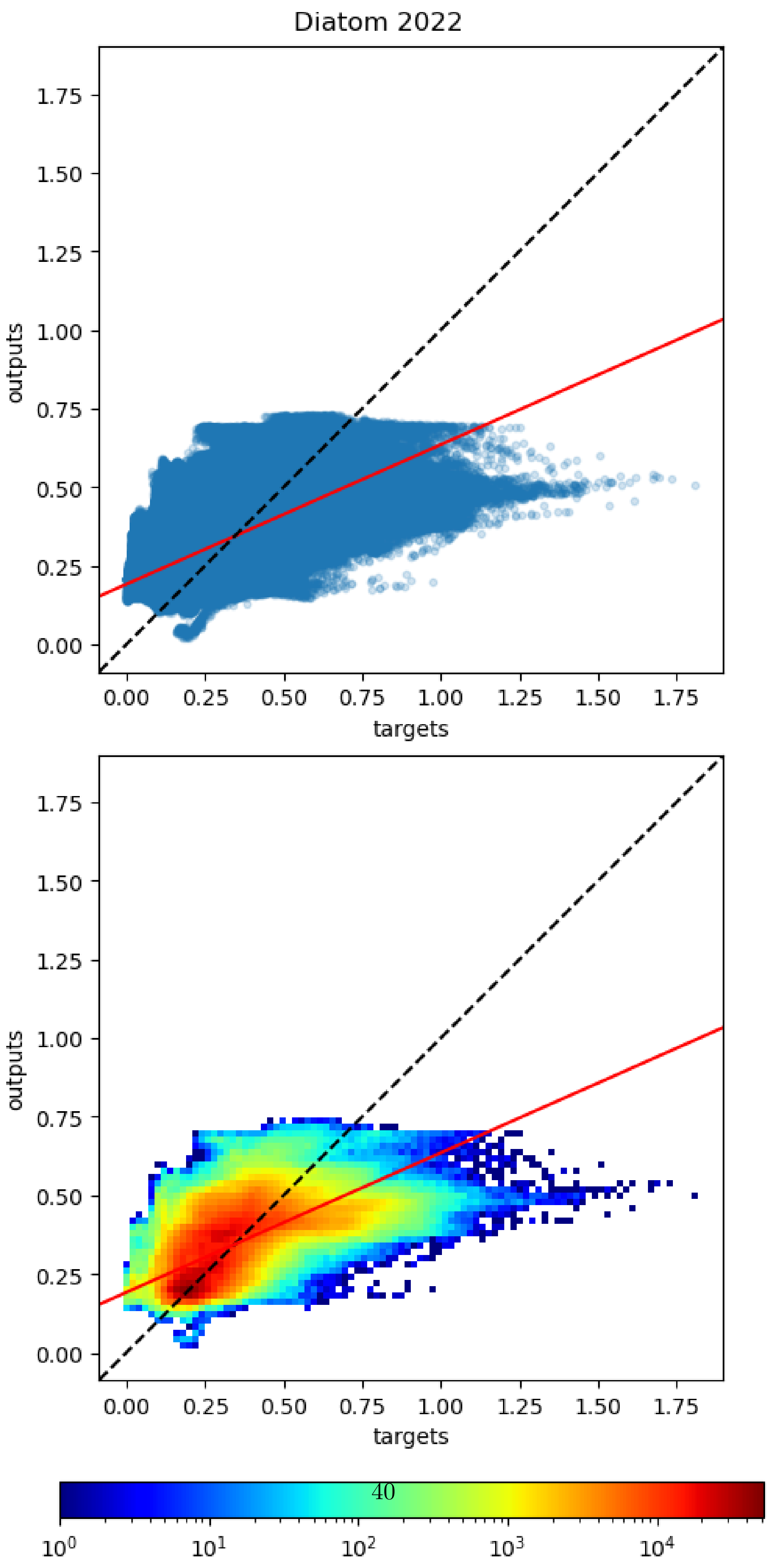
```
Gathering days for year 2020

  0%|          | 0/76 [00:00<?, ?it/s]

The amount of data points is 3532404
The slope of the best fitting line is  0.234
The correlation coefficient is: 0.462
 The mean square error is: 0.03299
```


Diatom 2020
outputs
targets
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
outputs
targets
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
10^0 10^1 10^2 10^3 10^4

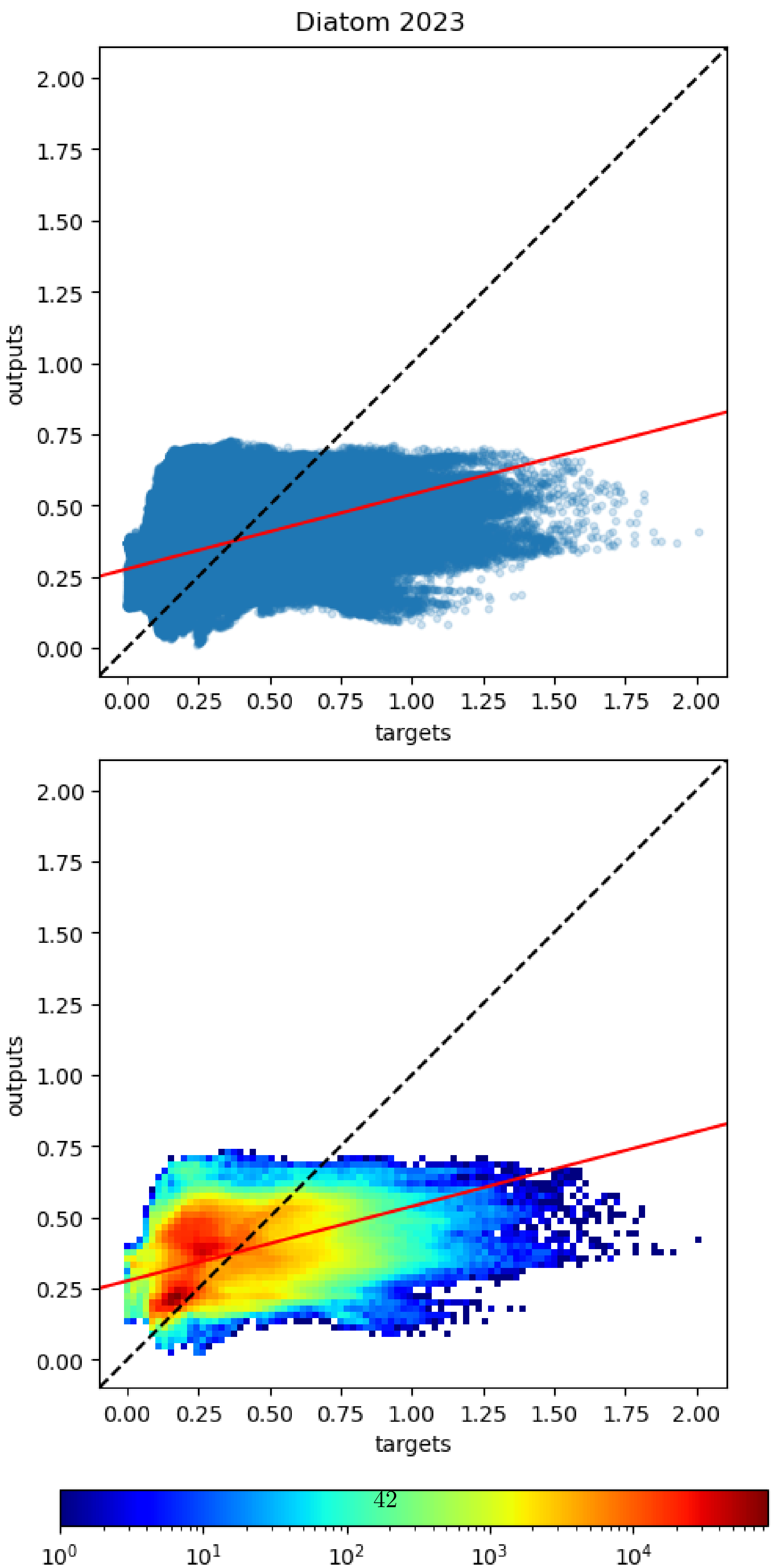
```
Gathering days for year 2021

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.404
The correlation coefficient is: 0.642
 The mean square error is: 0.01819
```

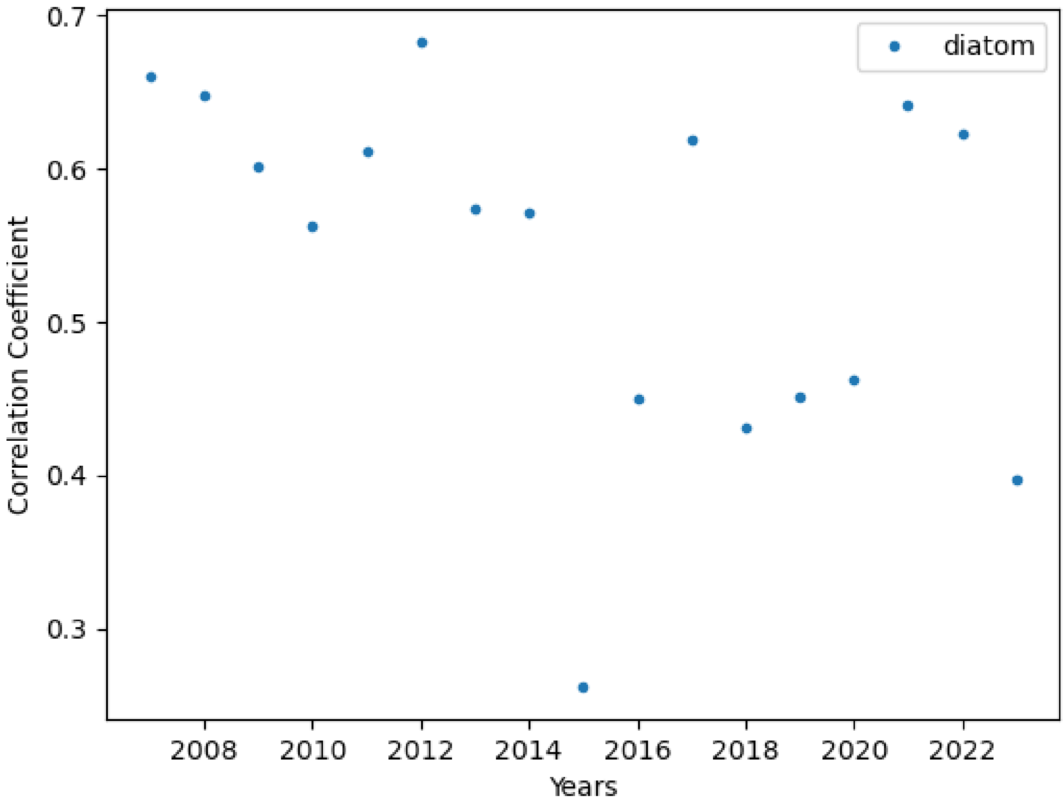

Diatom 2021
outputs
targets
0.0
0.5
1.0
1.5
2.0
10^0
10^1
10^2
10^3
10^4
10^5

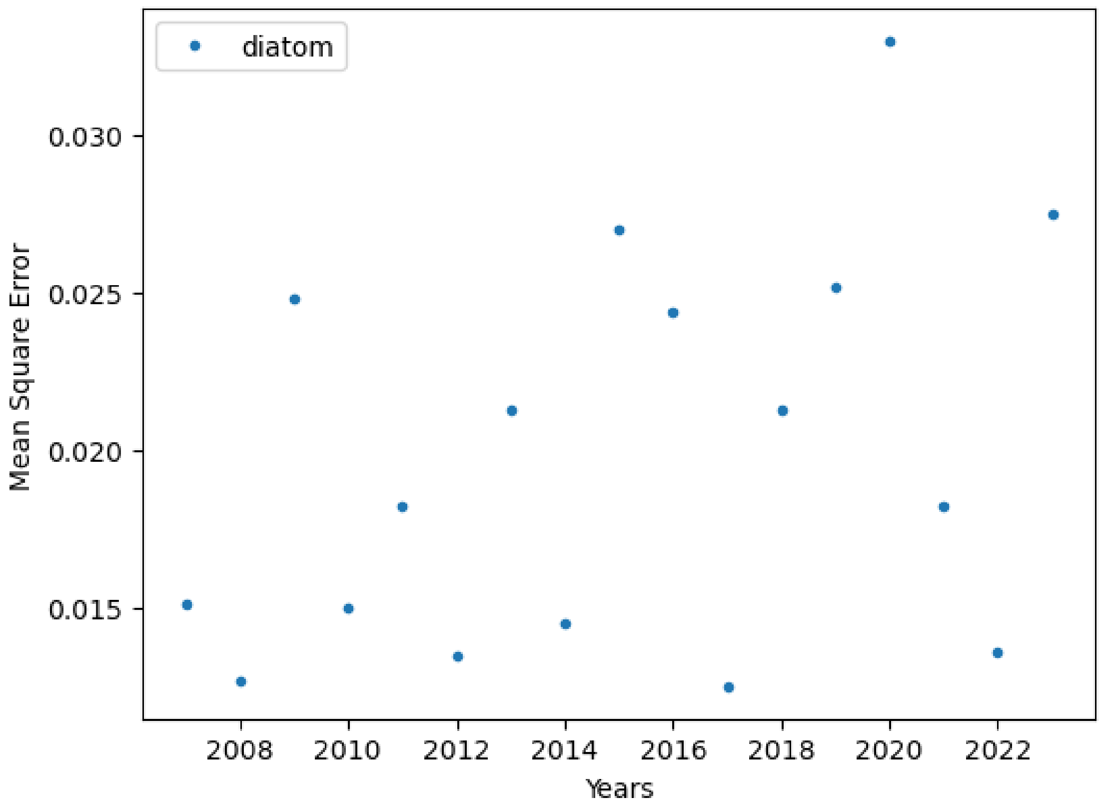
```
Gathering days for year 2022

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.444
The correlation coefficient is: 0.623
 The mean square error is: 0.01355
```

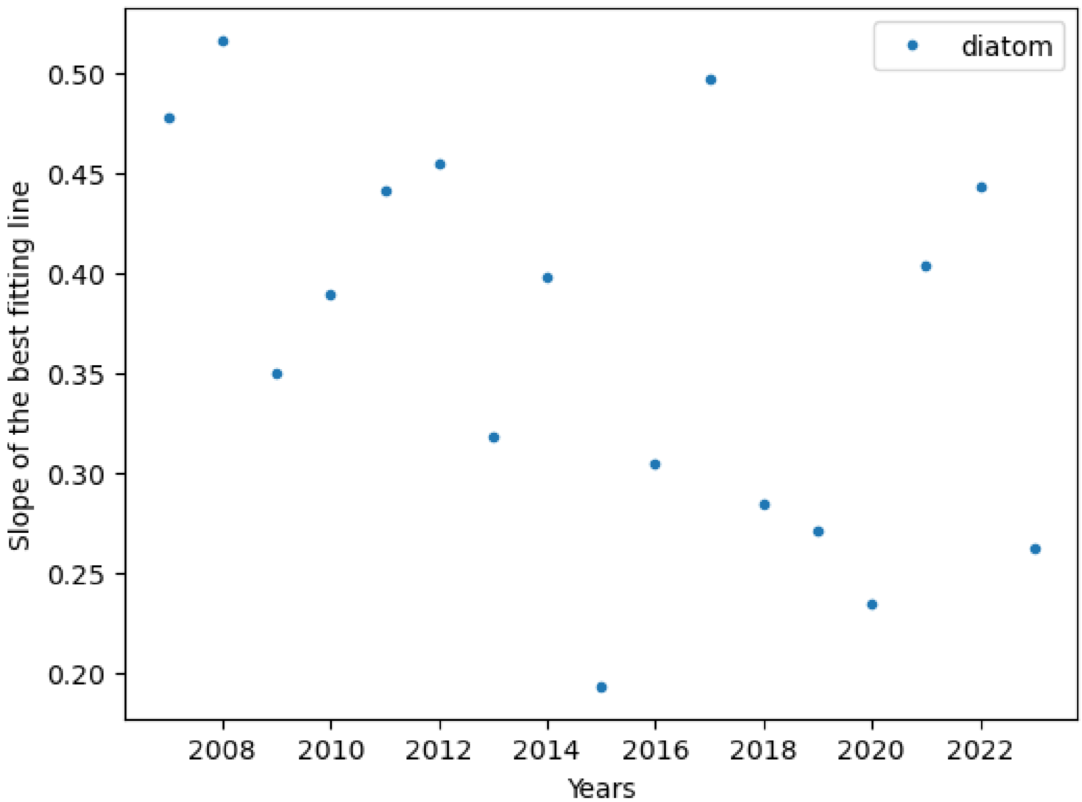

Diatom 2022
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
10^0
10^1
10^2
10^3
10^4

```
Gathering days for year 2023

  0%|          | 0/75 [00:00<?, ?it/s]

The amount of data points is 3485925
The slope of the best fitting line is  0.262
The correlation coefficient is: 0.397
 The mean square error is: 0.02754
```


Diatom 2023
outputs
targets
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
10^0
10^1
10^2
10^3
10^4


0.7
0.6
0.5
0.4
0.3
Correlation Coefficient
diatom
2008
2010
2012
2014
2016
2018
2020
2022
Years


diatom
0.030
0.025
0.020
0.015
Mean Square Error
2008
2010
2012
2014
2016
2018
2020
2022
Years

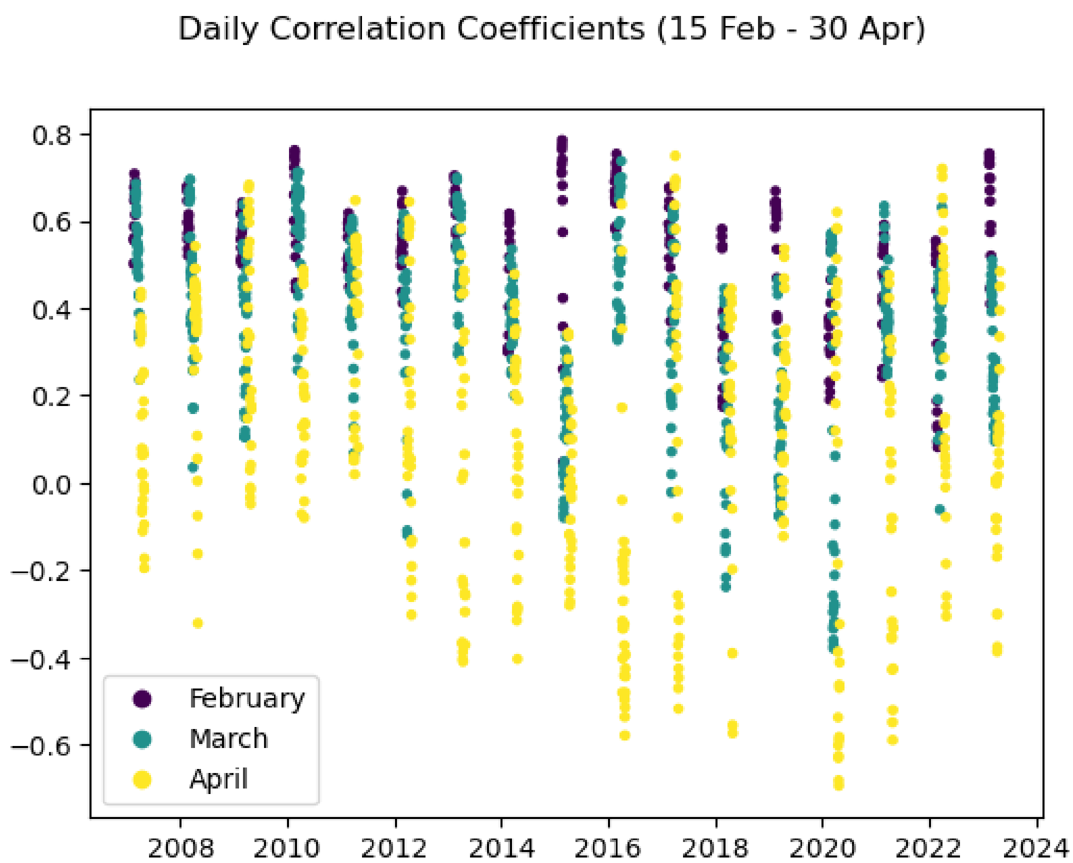


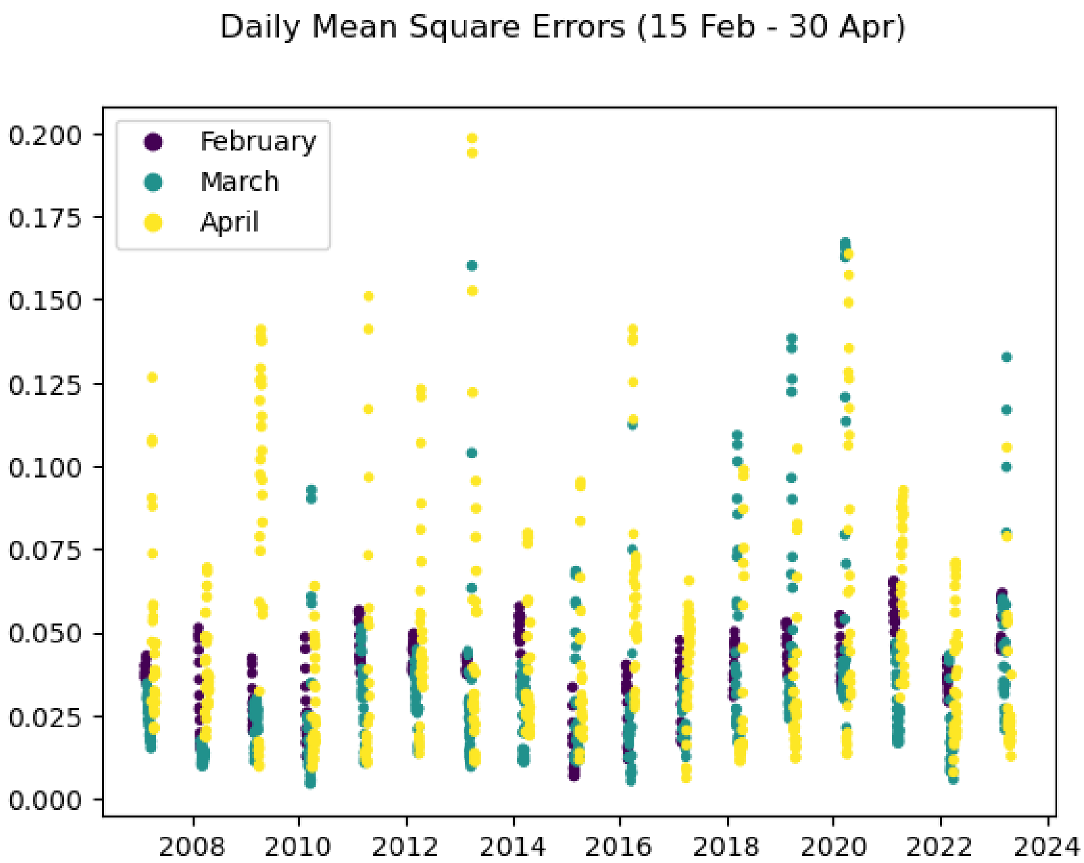
1.10 Other Years (Daily)

[ ]:
```
dates = np.array([])
r_all2 = np.array([])
rms_all2 = np.array([])
slope_all2 = np.array([])

folders = [x for x in path if ((x[2:5]=='mar' or x[2:5]=='apr' or (x[2:
↪5]=='feb' and x[0:2]>'14')) and (x[5:7]<'24'))]
indx_dates=(np.argsort(pd.to_datetime(folders, format="%d%b%y")))
folders = [folders[i] for i in indx_dates]

for i in tqdm(folders, colour = 'green'):

    date, temp_i1, temp_i2, saline_i1, saline_i2, diat_i, =␣
↪datasets_preparation()

    drivers = np.stack([np.ravel(temp_i1), np.ravel(temp_i2), np.
↪ravel(saline_i1), np.ravel(saline_i2)])
```

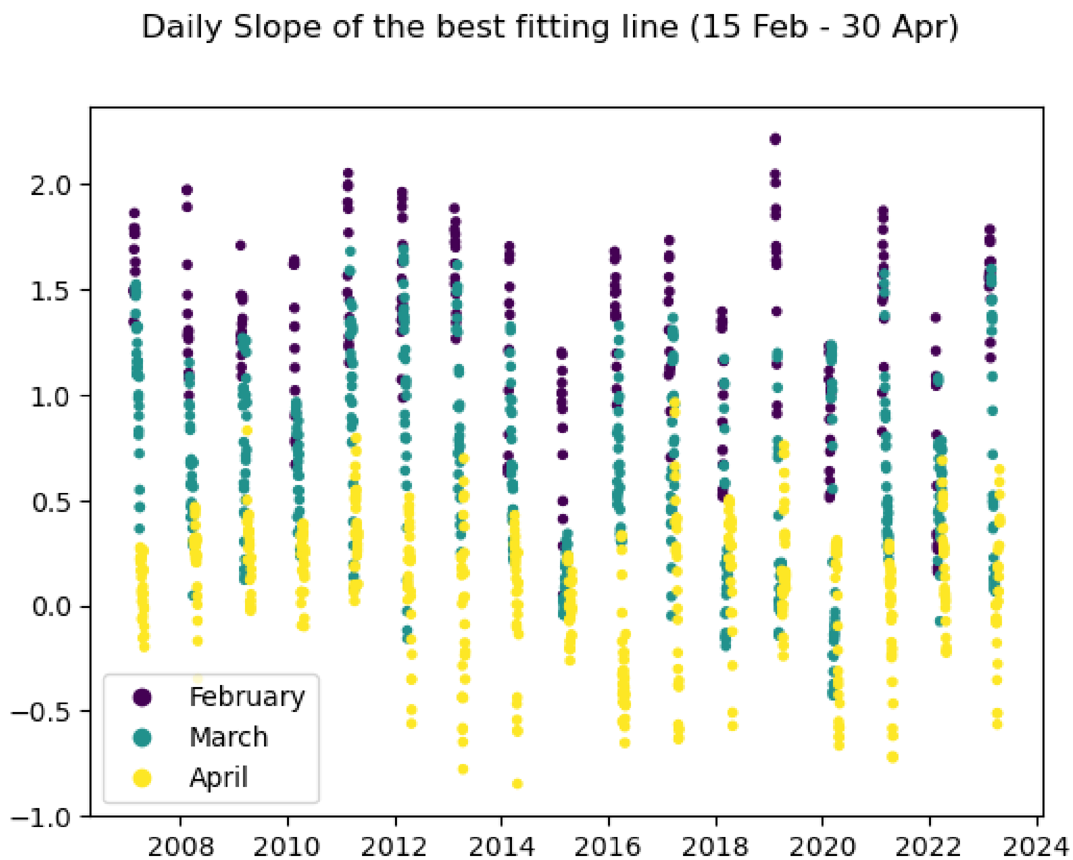
```
    indx = np.where(~np.isnan(drivers).any(axis=0))
    drivers = drivers[:,indx[0]]

    diat = np.ravel(diat_i)
    diat = diat[indx[0]]

    r, rms, m = regressor4(drivers, diat, 'Diatom')

    dates = np.append(dates,date.date)
    r_all2 = np.append(r_all2,r)
    rms_all2 = np.append(rms_all2,rms)
    slope_all2 = np.append(slope_all2,m)

plotting2(r_all2, 'Correlation Coefficients')
plotting2(rms_all2, 'Mean Square Errors')
plotting2(slope_all2, 'Slope of the best fitting line')
```

```
  0%|          | 0/1279 [00:00<?, ?it/s]
```





Daily Mean Square Errors (15 Feb - 30 Apr)
February
March
April
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
2008
2010
2012
2014
2016
2018
2020
2022
2024



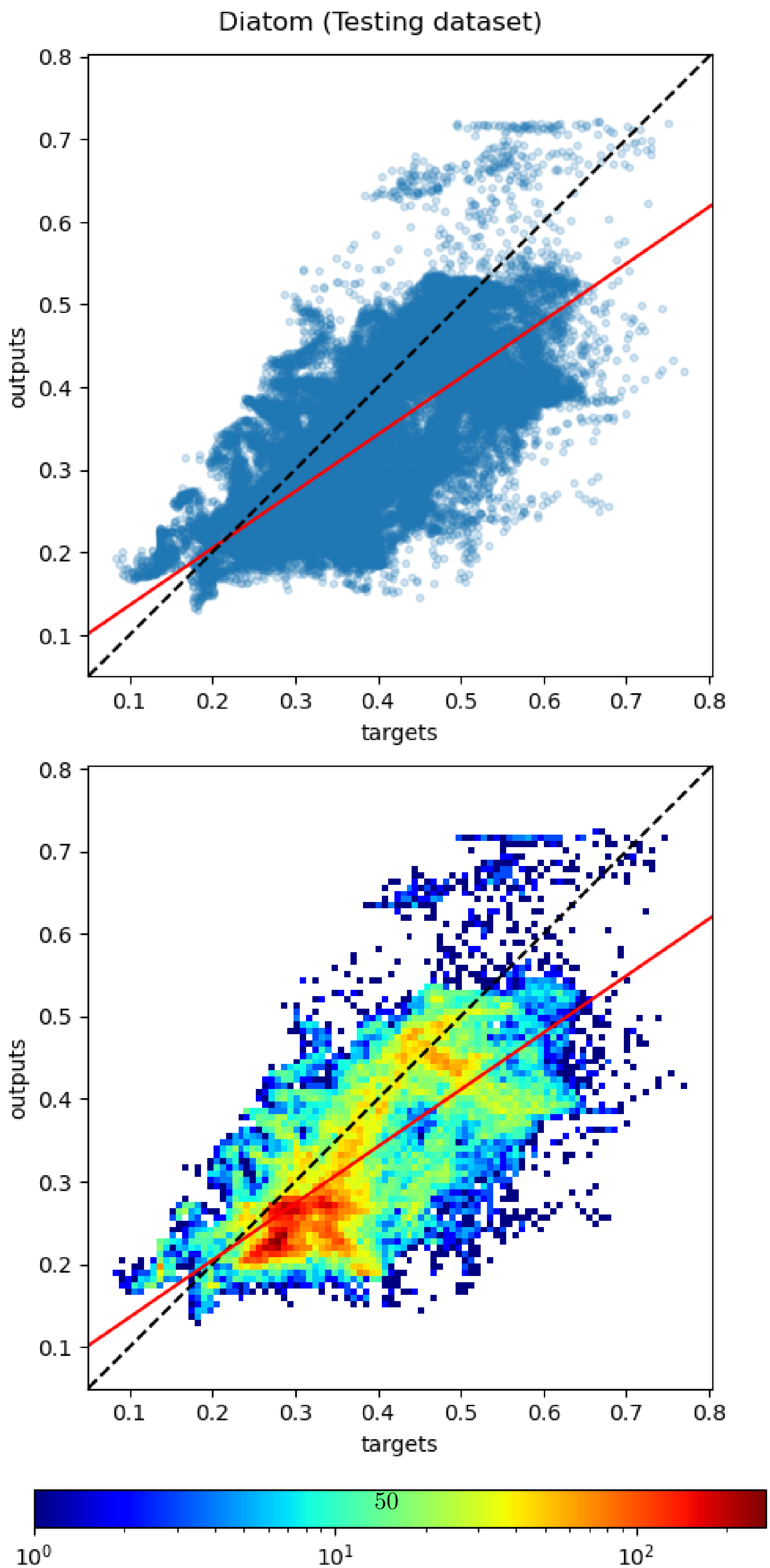
2 Daily Maps

[ ]:
```
maps = random.sample(folders,10)

for i in tqdm(maps):

    date, temp_i1, temp_i2, saline_i1, saline_i2, diat_i, =␣
 ↪datasets_preparation()

    drivers = np.stack([np.ravel(temp_i1), np.ravel(temp_i2), np.
 ↪ravel(saline_i1), np.ravel(saline_i2)])
    indx = np.where(~np.isnan(drivers).any(axis=0))
    drivers = drivers[:,indx[0]]

    diat = np.ravel(diat_i)
    diat = diat[indx[0]]
```

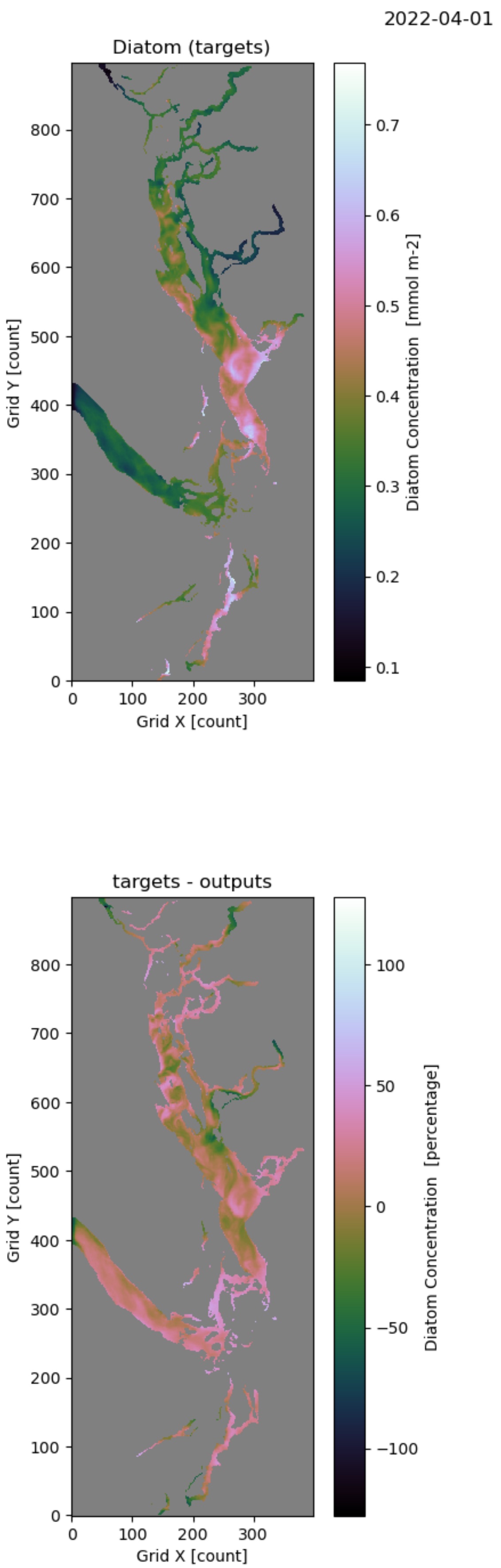
```
regressor3(drivers, diat, 'Diatom')
```

```
  0%|          | 0/10 [00:00<?, ?it/s]
```

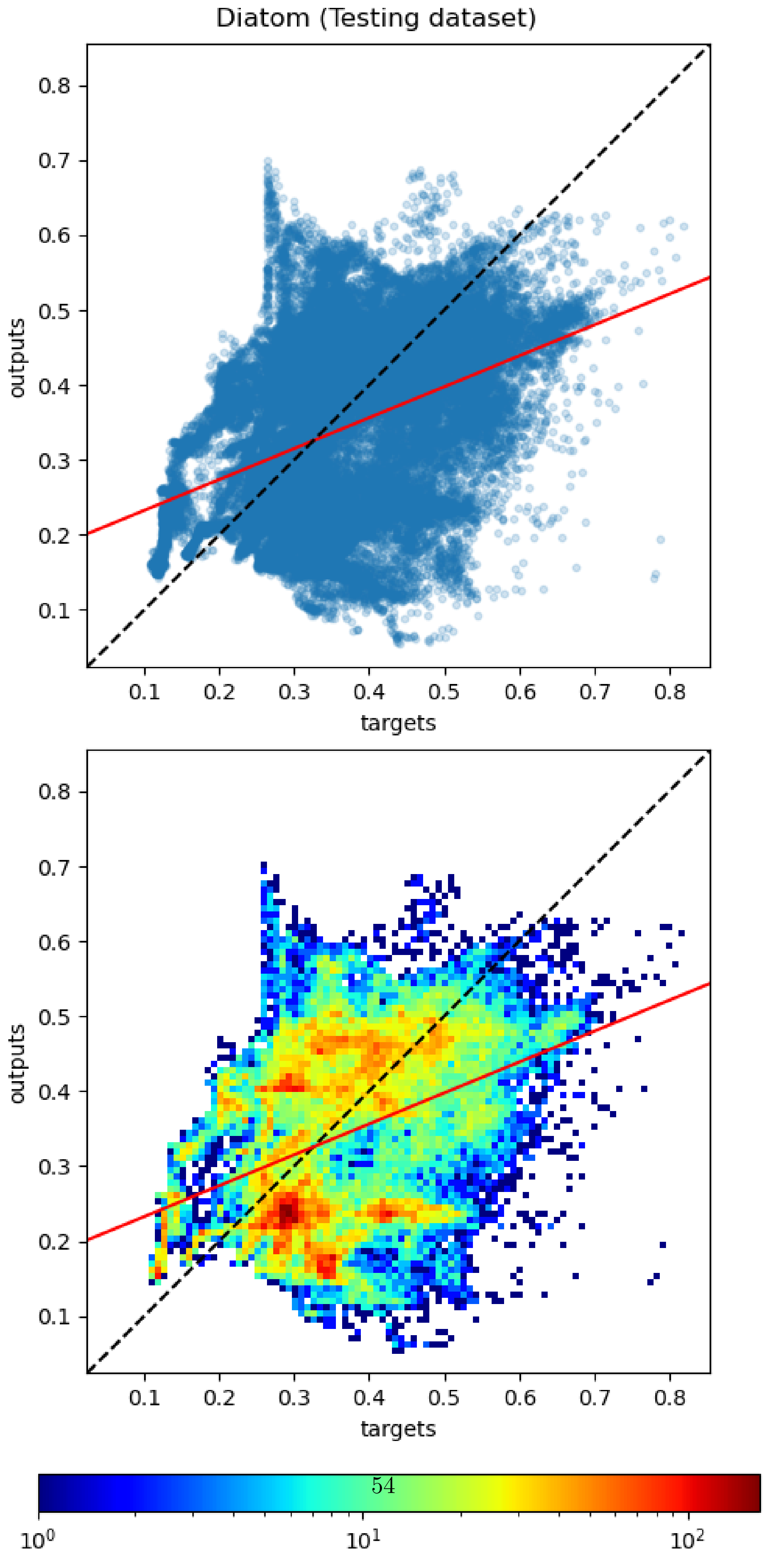
```
The amount of data points is 46479
The slope of the best fitting line is  0.688
The correlation coefficient is: 0.719
 The mean square error is: 0.00805
```

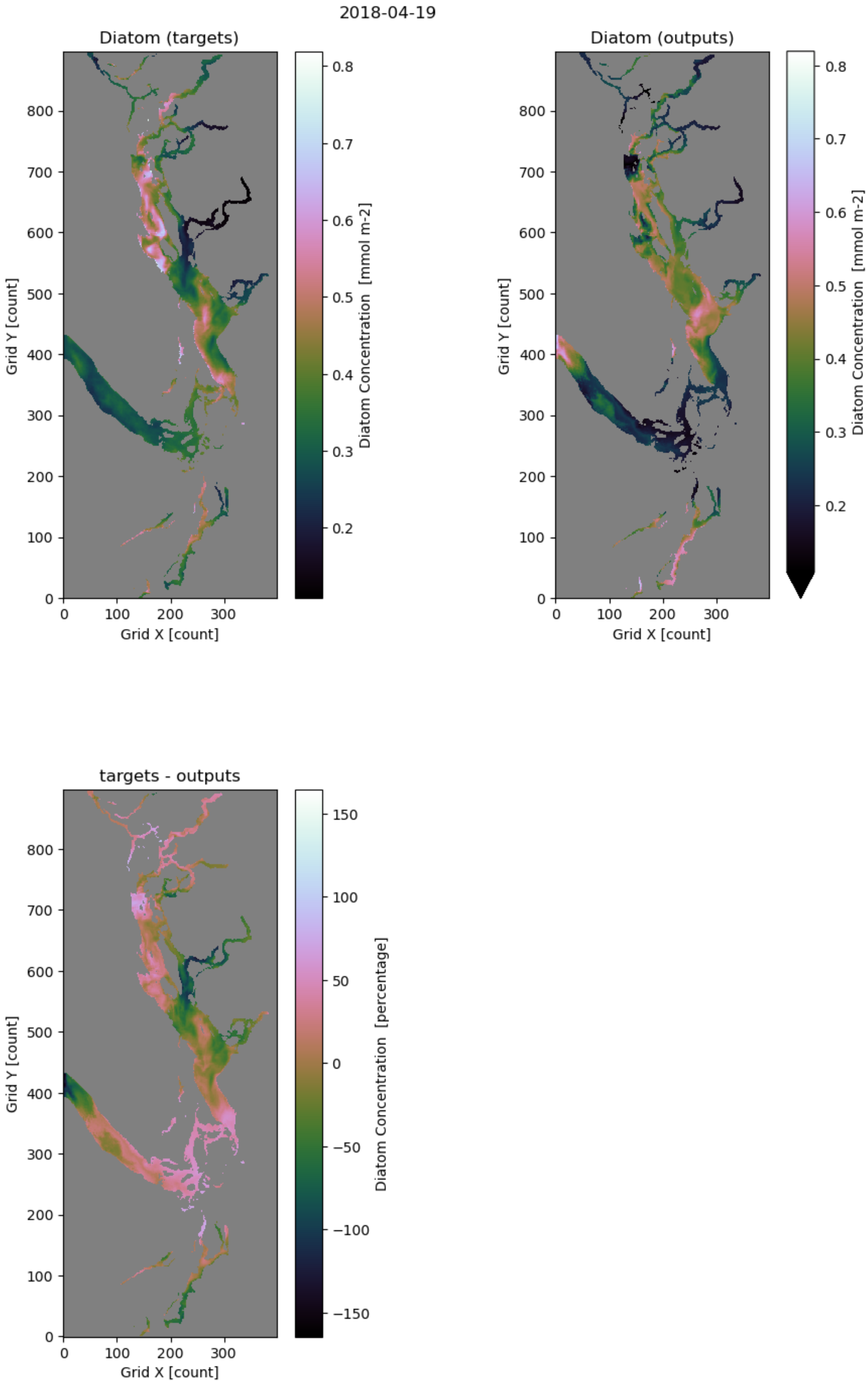


Diatom (Testing dataset)
outputs
targets
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
10^0
10^1
10^2

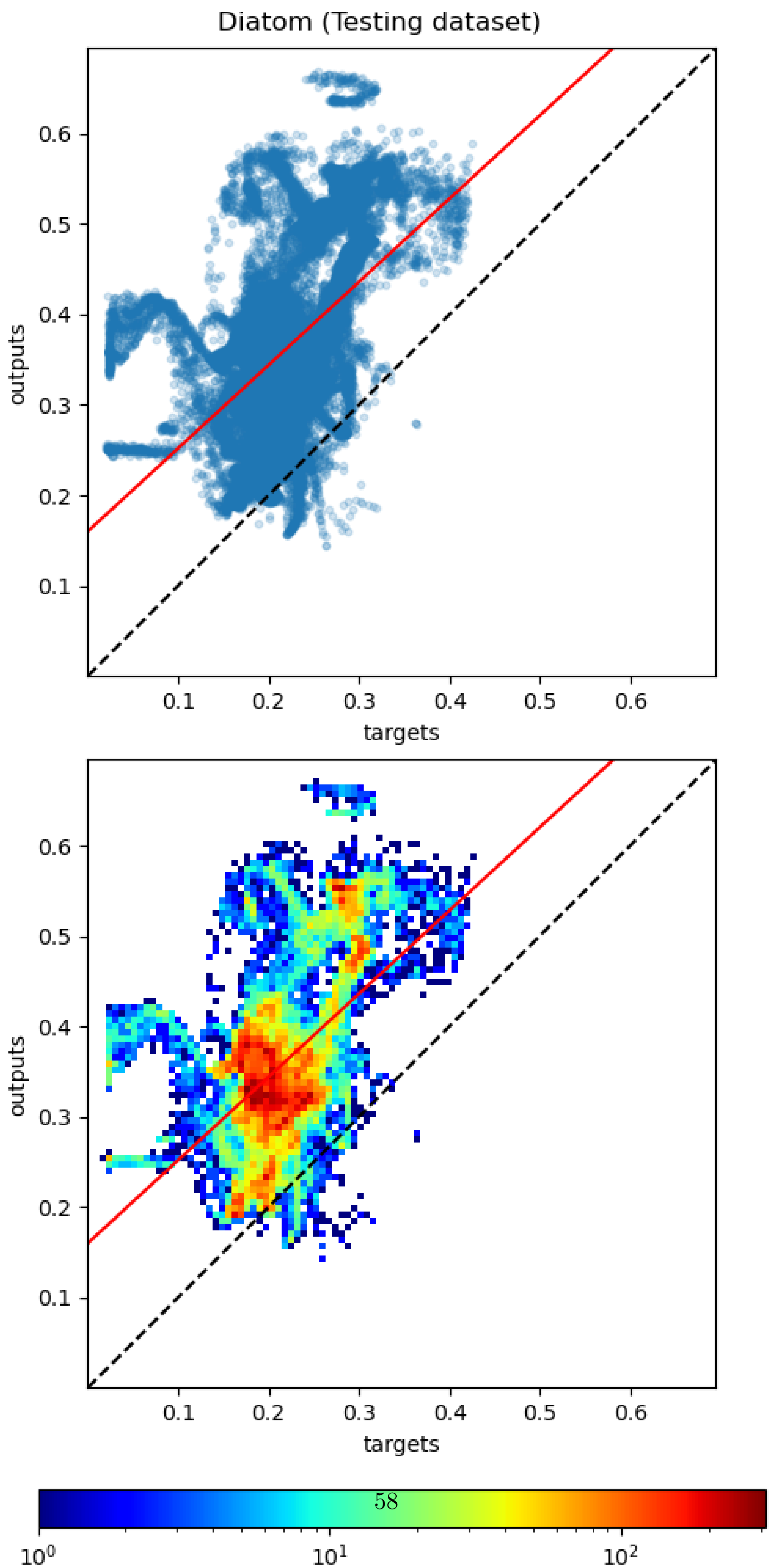

2022-04-01
Diatom (targets)
Diatom (outputs)
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
0
100
200
300
400
500
600
700
800
0.1
0.2
0.3
0.4
0.5
0.6
0.7
targets - outputs
Diatom Concentration [percentage]
−100
−50
50
100

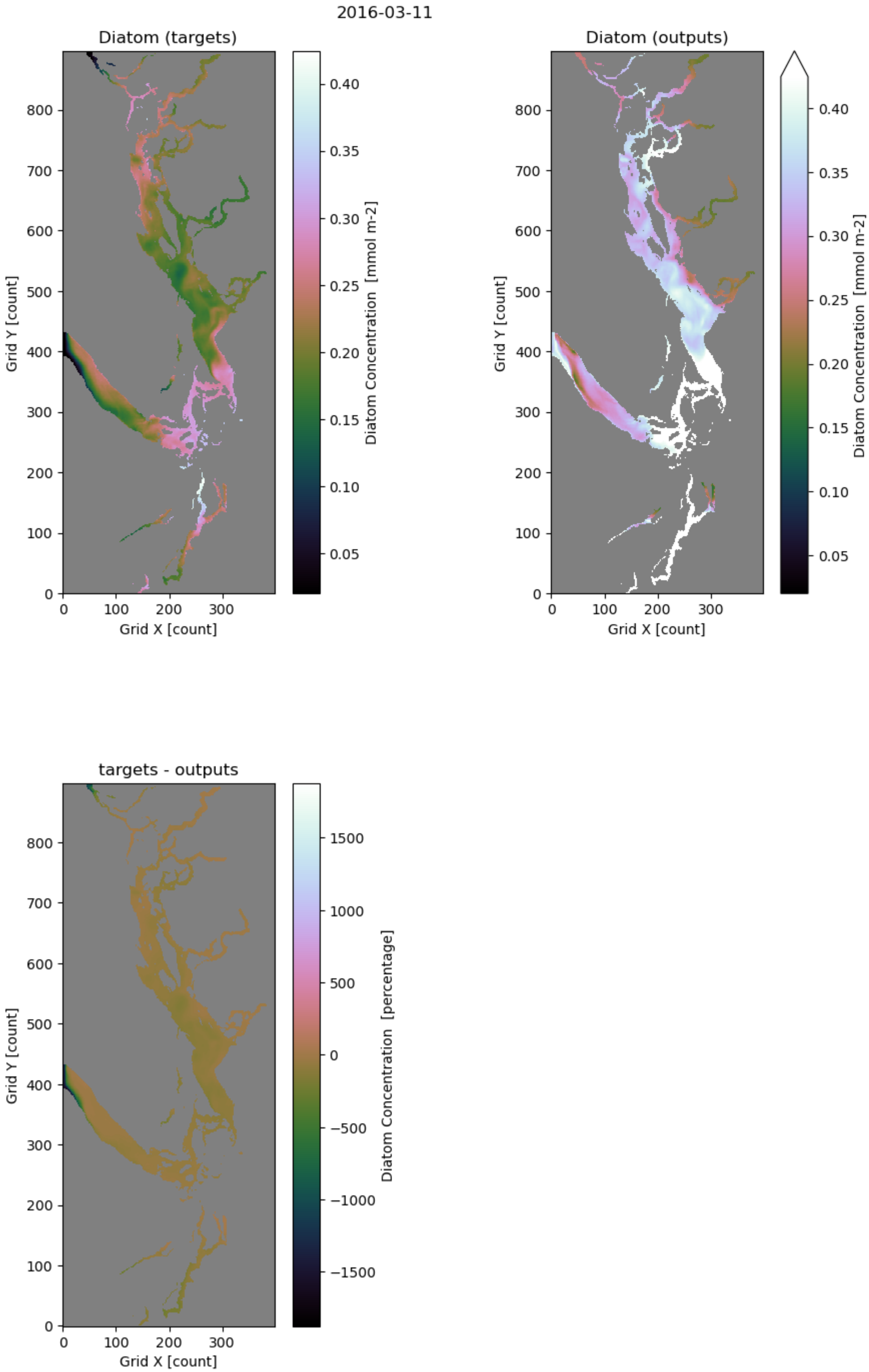
```
The amount of data points is 46479
The slope of the best fitting line is  0.412
The correlation coefficient is: 0.381
 The mean square error is: 0.01675
```


Diatom (Testing dataset)
outputs
targets
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
outputs
targets
10^0
10^1
10^2


2018-04-19
Diatom (targets)
Diatom (outputs)
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
targets - outputs
Diatom Concentration [percentage]

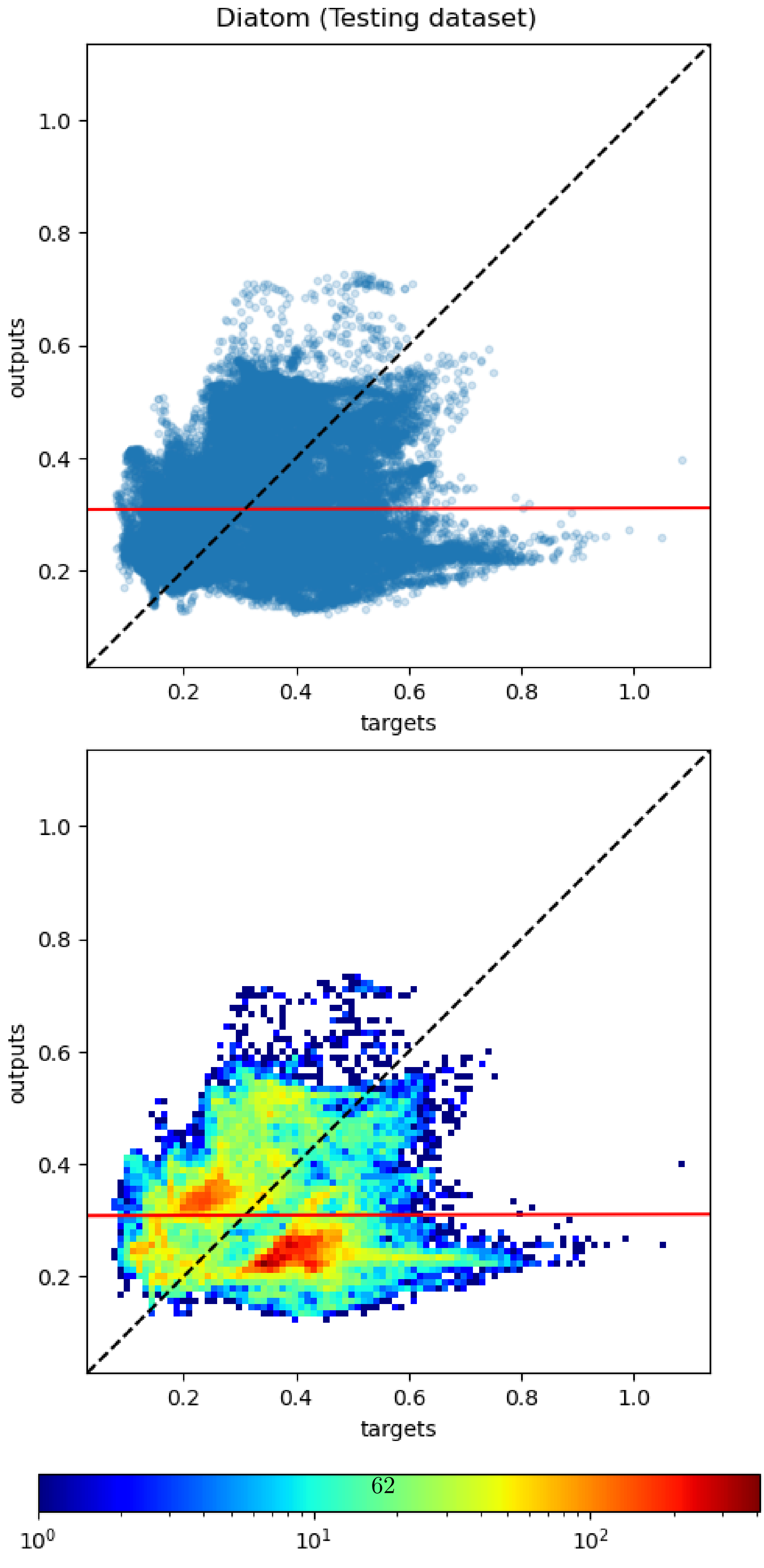
```
The amount of data points is 46479
The slope of the best fitting line is  0.92
The correlation coefficient is: 0.51
 The mean square error is: 0.02663
```

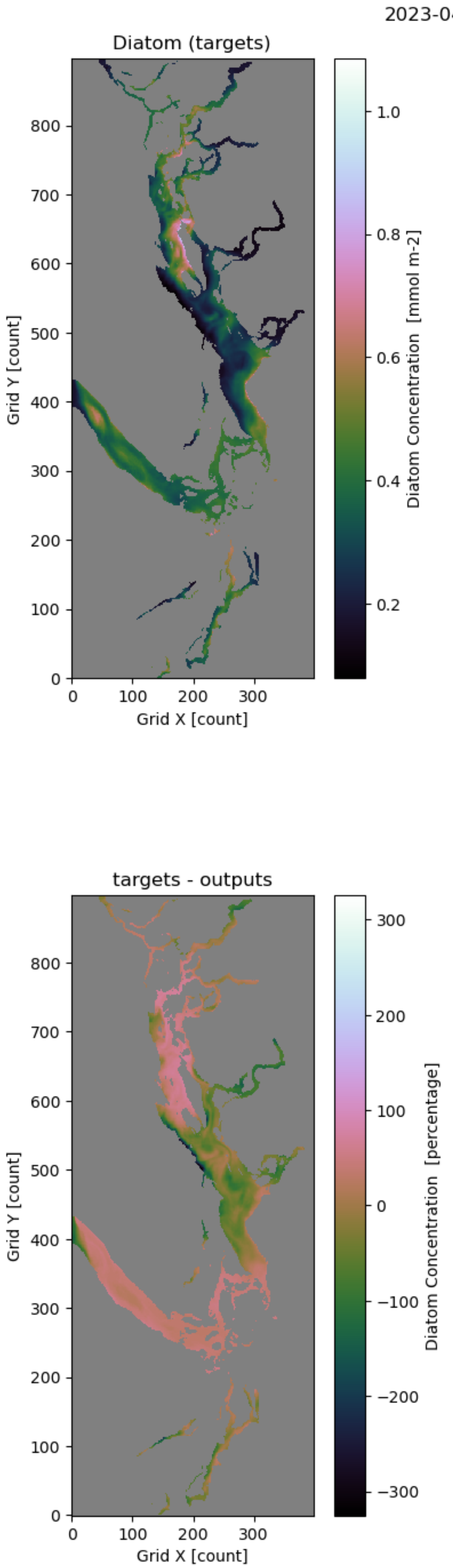

Diatom (Testing dataset)
outputs
targets
0.1
0.2
0.3
0.4
0.5
0.6
outputs
targets
0.1
0.2
0.3
0.4
0.5
0.6
10^0
10^1
10^2



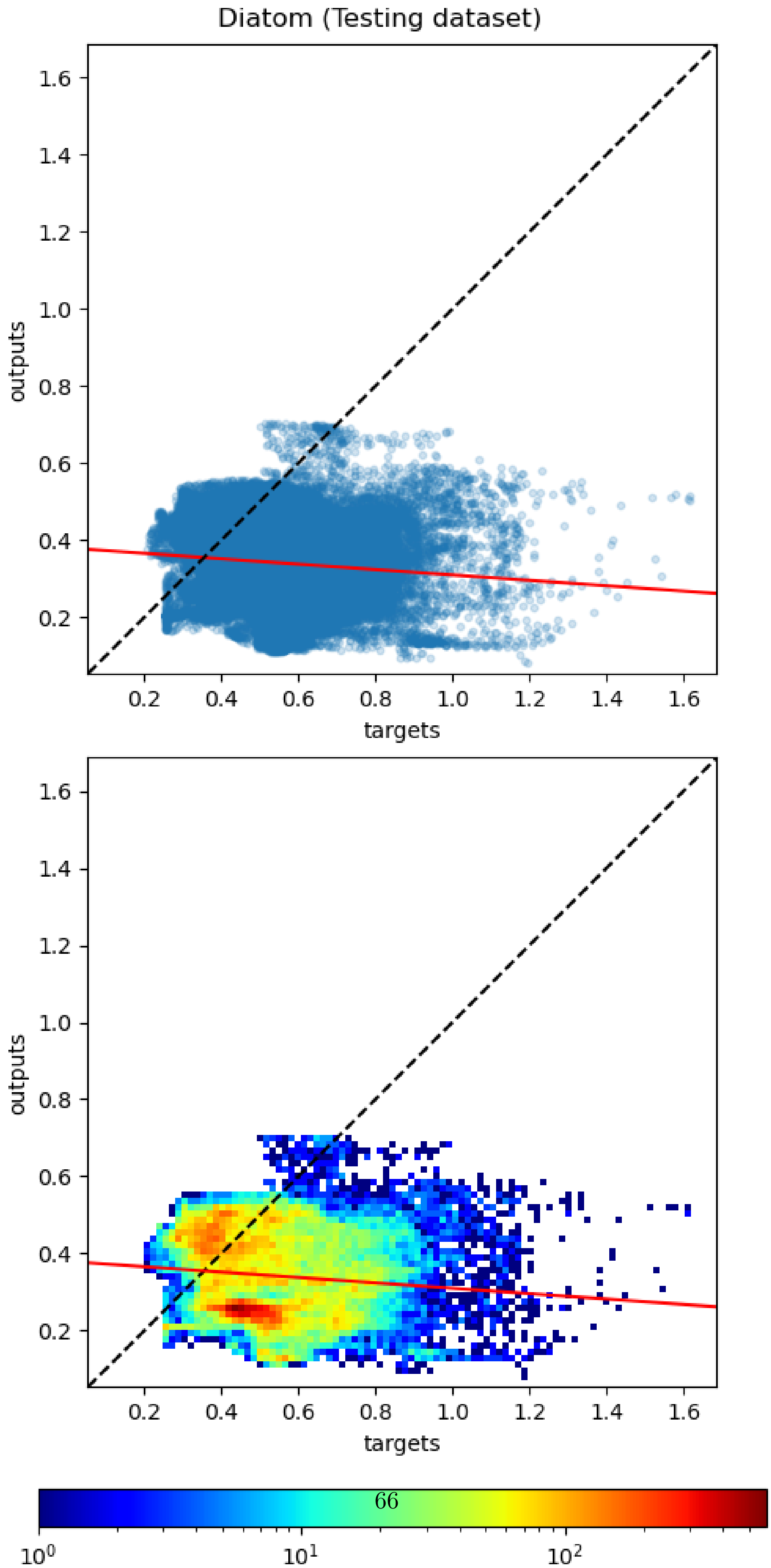
2016-03-11
Diatom (targets)
Diatom (outputs)
targets - outputs
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
Diatom Concentration [percentage]
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
1500
1000
500
0
−500
−1000
−1500

```
The amount of data points is 46479
The slope of the best fitting line is  0.002
The correlation coefficient is: 0.003
 The mean square error is: 0.02723
```

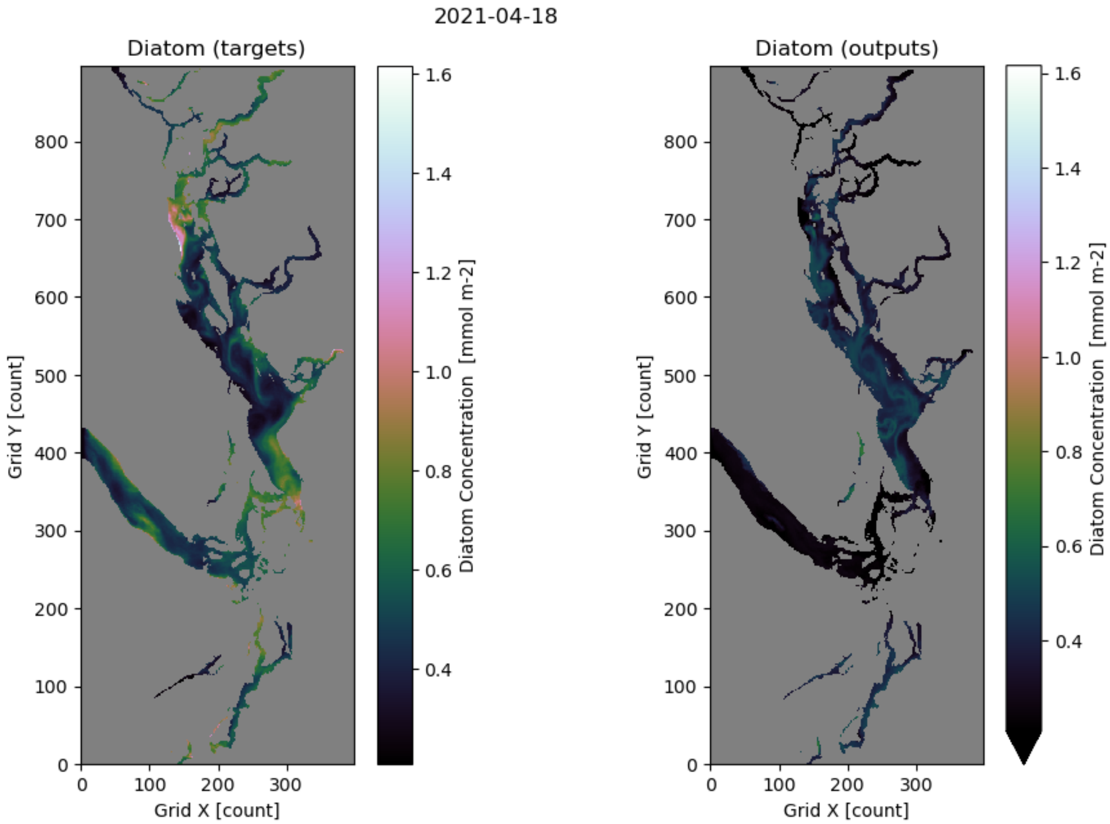

Diatom (Testing dataset)
outputs
targets
0.2
0.4
0.6
0.8
1.0
10^0
10^1
10^2

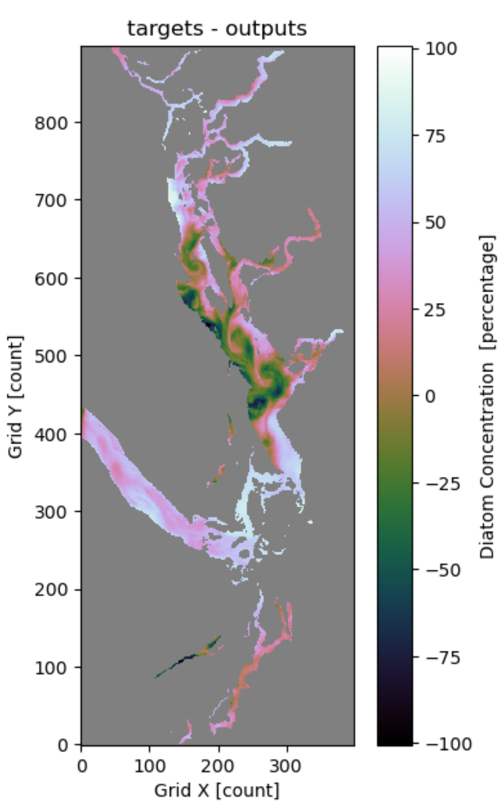

2023-04-07
Diatom (targets)
Diatom (outputs)
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
targets - outputs
Diatom Concentration [percentage]

```
The amount of data points is 46479
The slope of the best fitting line is  -0.07
The correlation coefficient is: -0.105
 The mean square error is: 0.07627
```

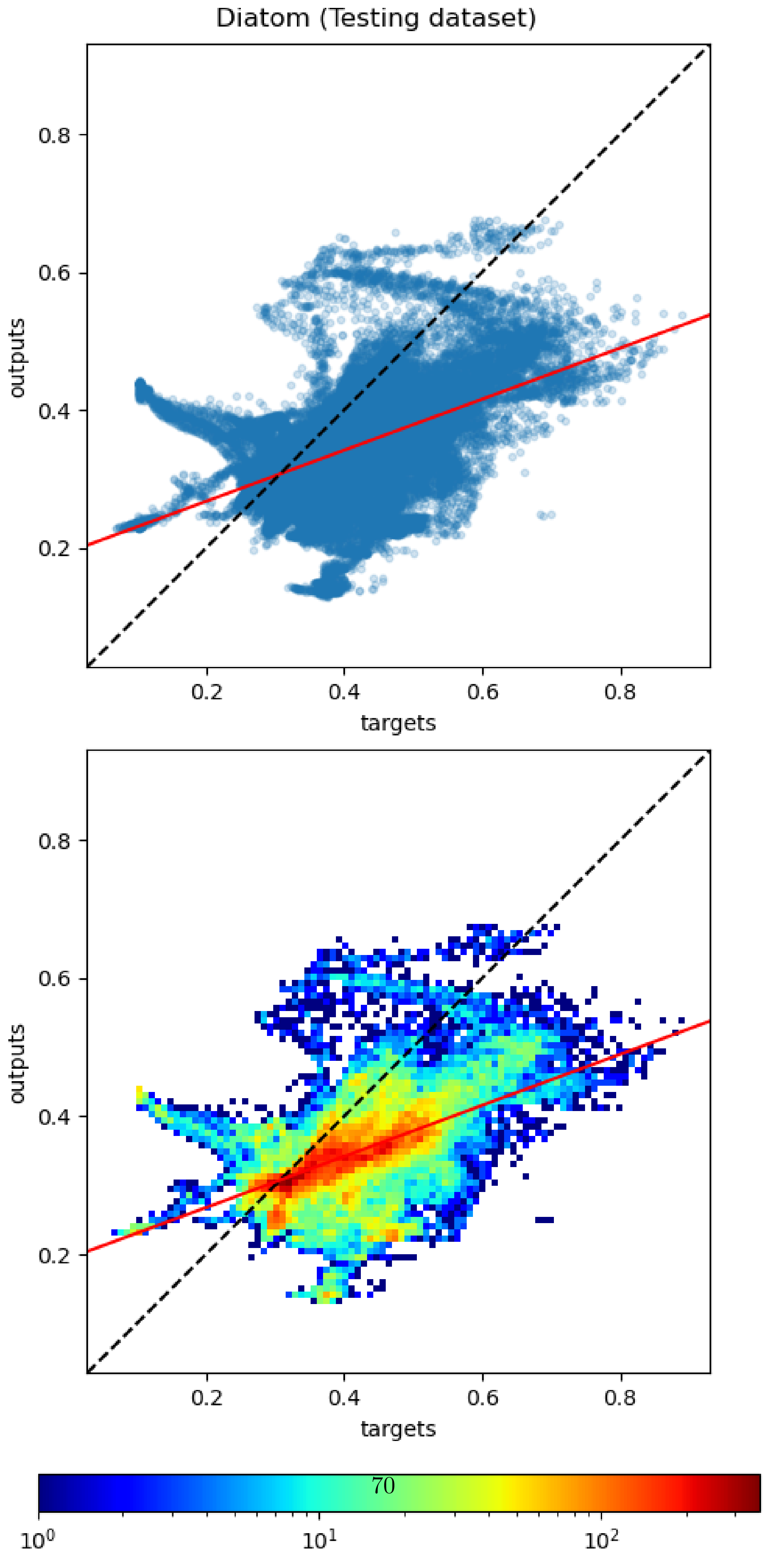


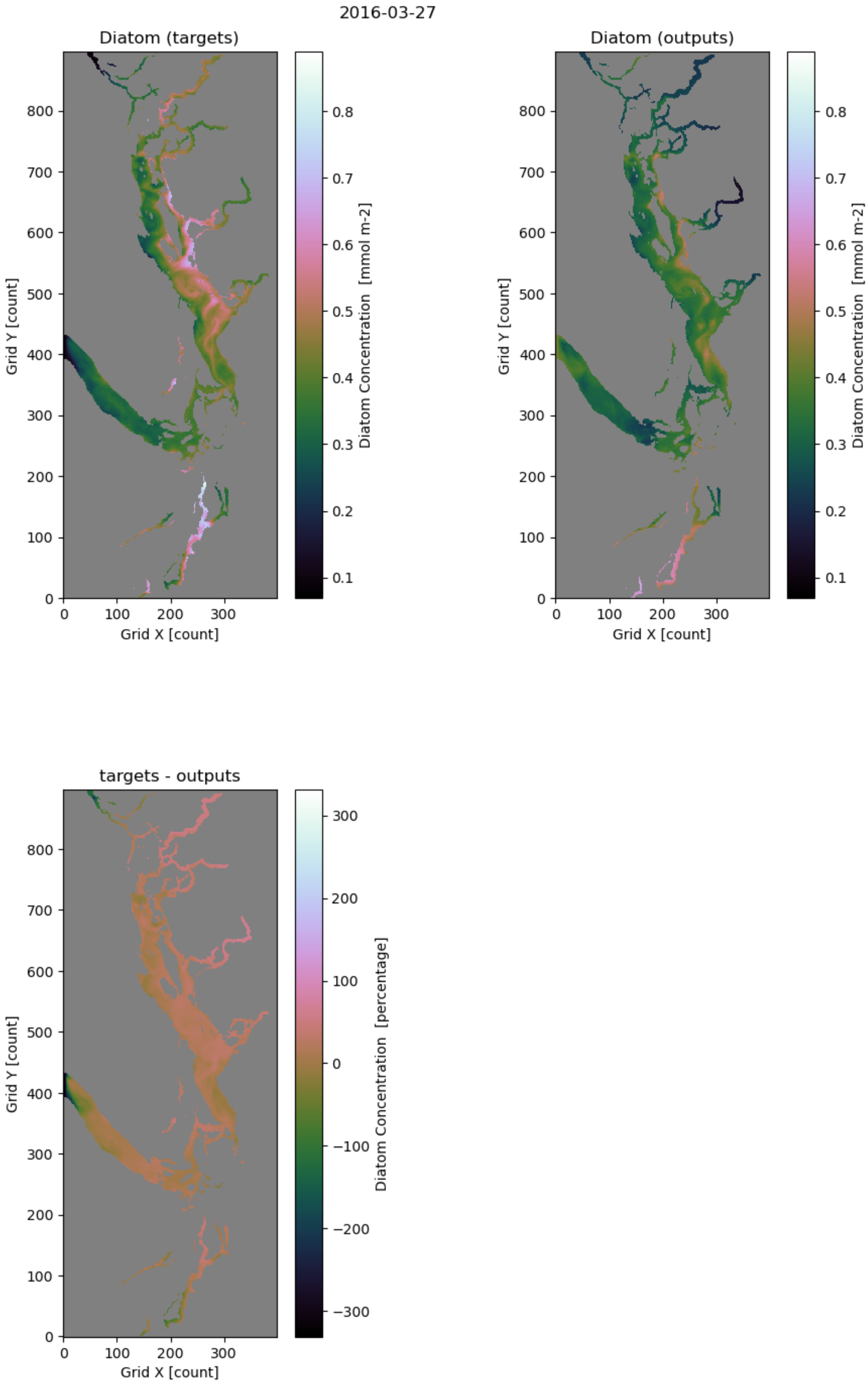
Diatom (Testing dataset)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
outputs
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
targets
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
outputs
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
targets
66
10^0
10^1
10^2


2021-04-18
Diatom (targets)
Diatom (outputs)
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
0
100
200
300
400
500
600
700
800
0.4
0.6
0.8
1.0
1.2
1.4
1.6

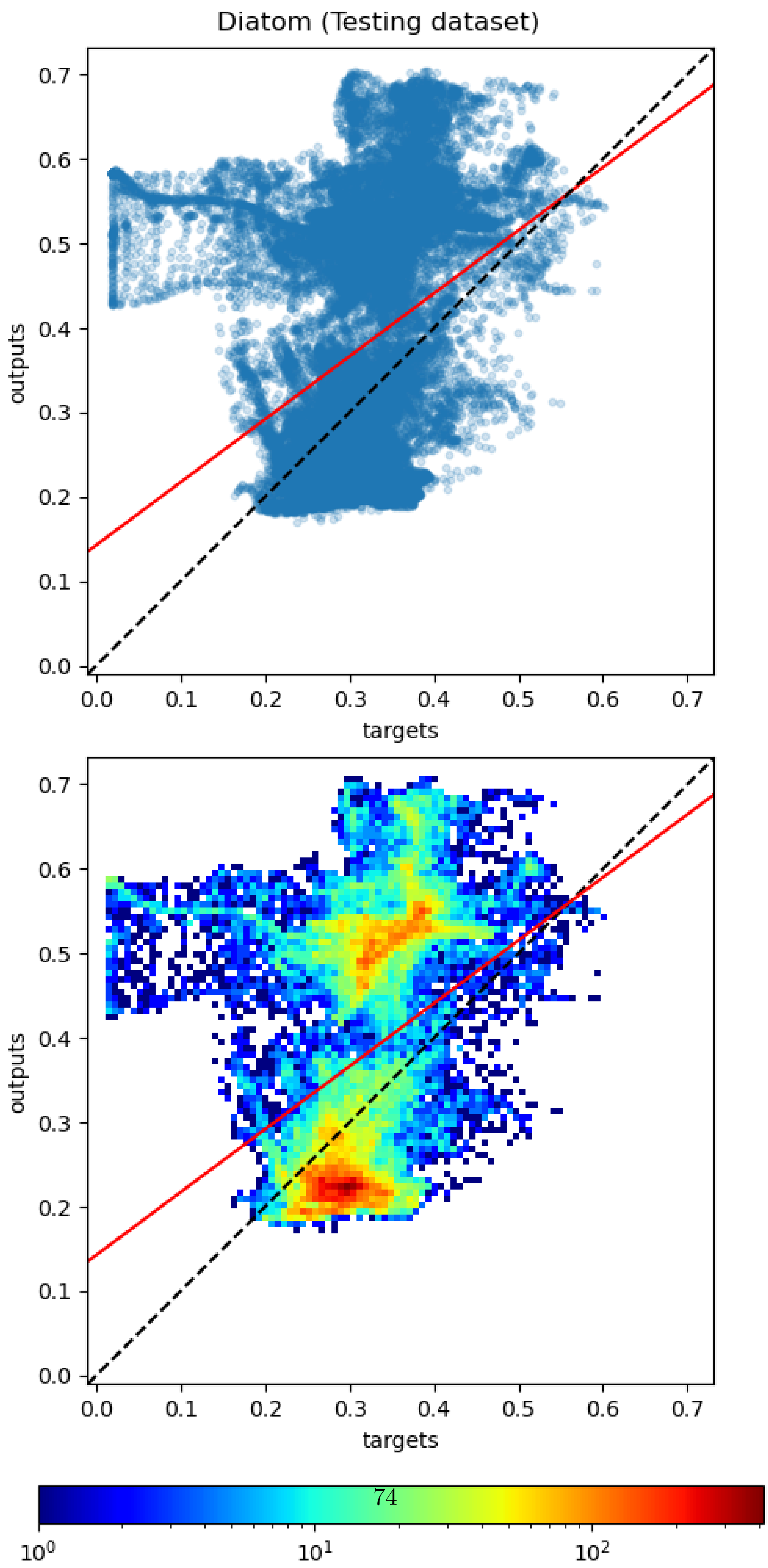

targets - outputs
Grid Y [count]
Grid X [count]
Diatom Concentration [percentage]
100
75
50
25
0
−25
−50
−75
−100

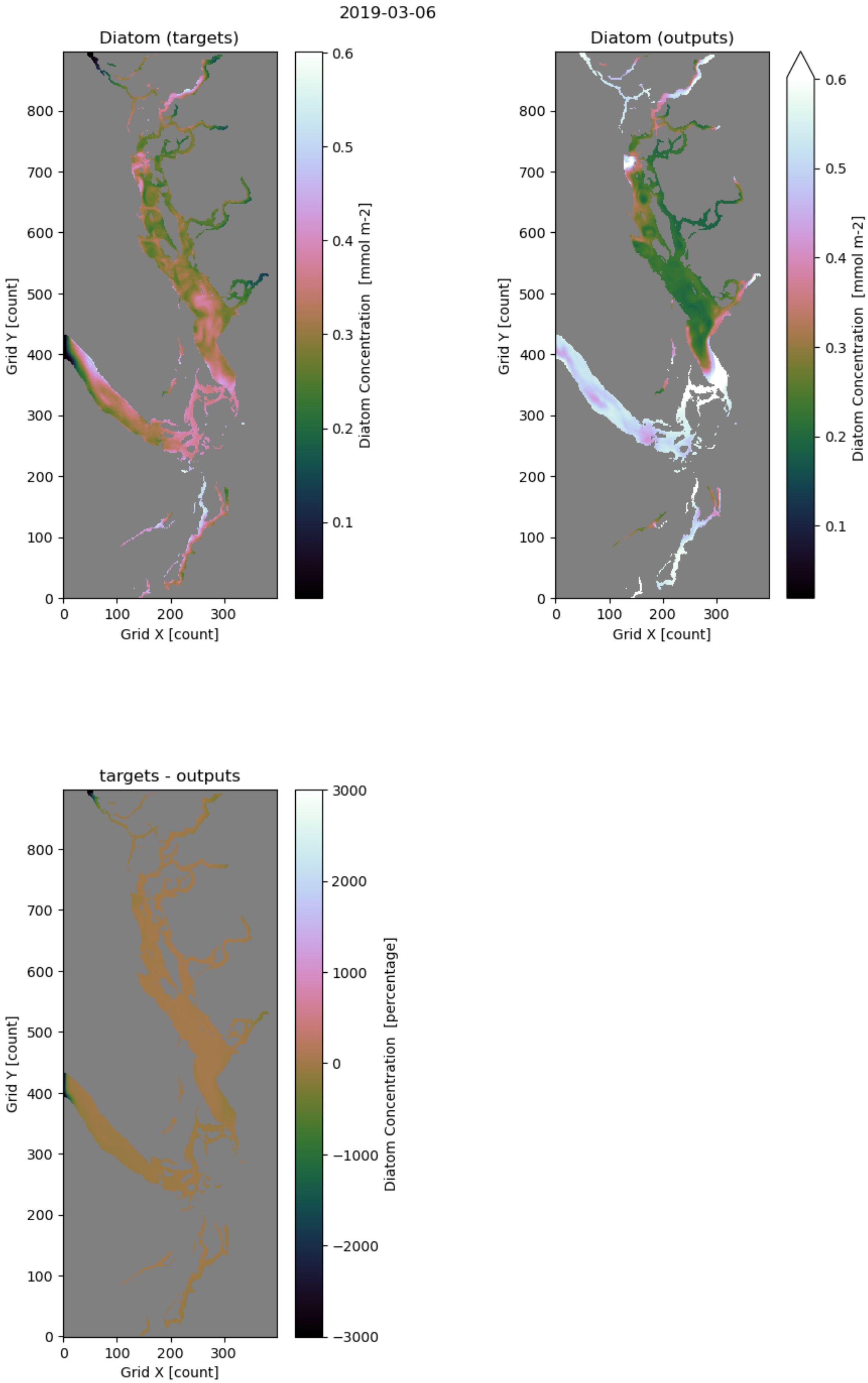
```
The amount of data points is 46479
The slope of the best fitting line is  0.37
The correlation coefficient is: 0.506
 The mean square error is: 0.01297
```


Diatom (Testing dataset)
outputs
targets
0.2
0.4
0.6
0.8
10^0
10^1
10^2


2016-03-27
Diatom (targets)
Diatom (outputs)
targets - outputs
Grid Y [count]
Grid X [count]
0
100
200
300
400
500
600
700
800
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Diatom Concentration [mmol m-2]
−300
−200
−100
0
100
200
300
Diatom Concentration [percentage]

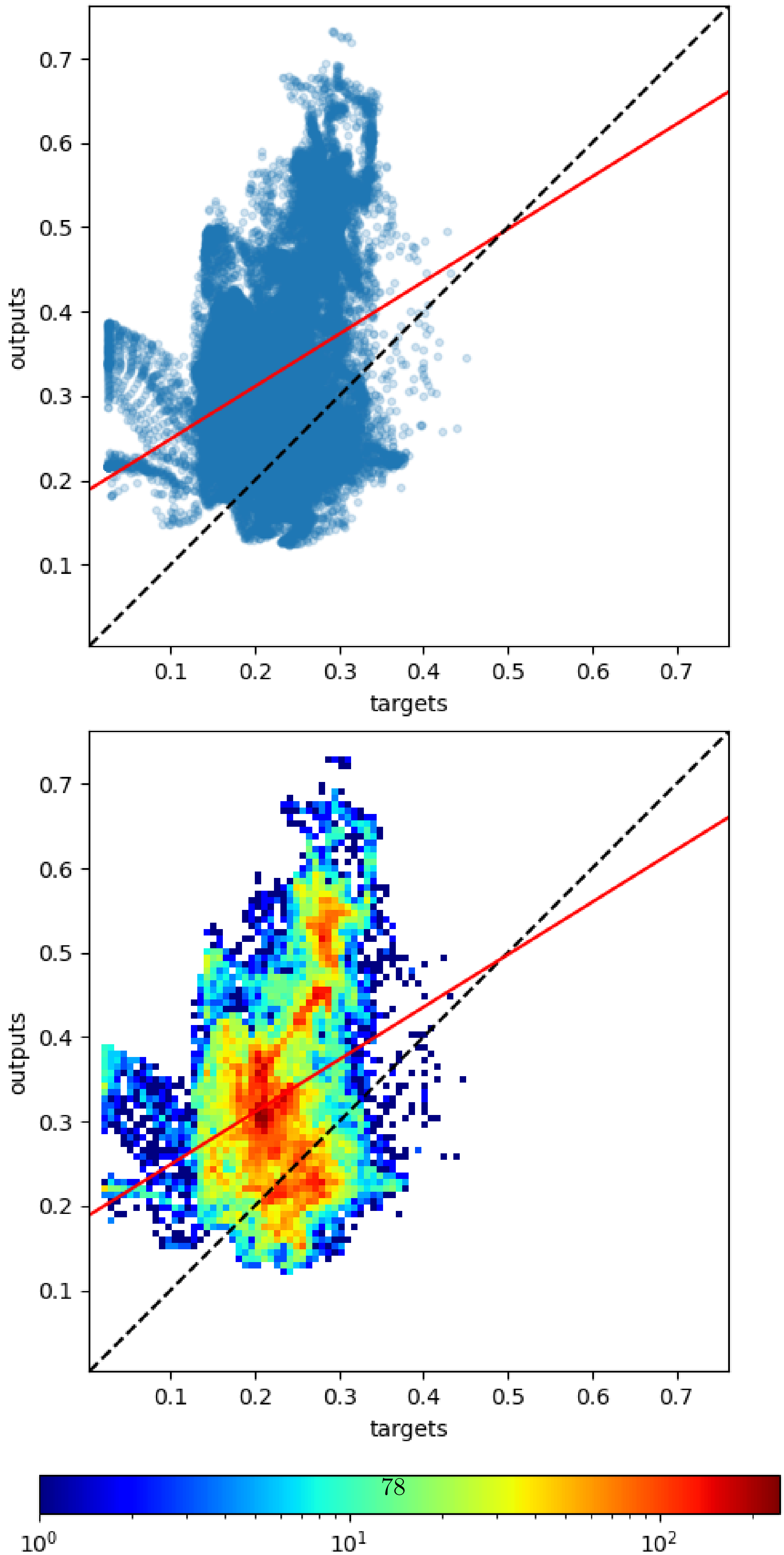
```
The amount of data points is 46479
The slope of the best fitting line is  0.746
The correlation coefficient is: 0.341
 The mean square error is: 0.02413
```

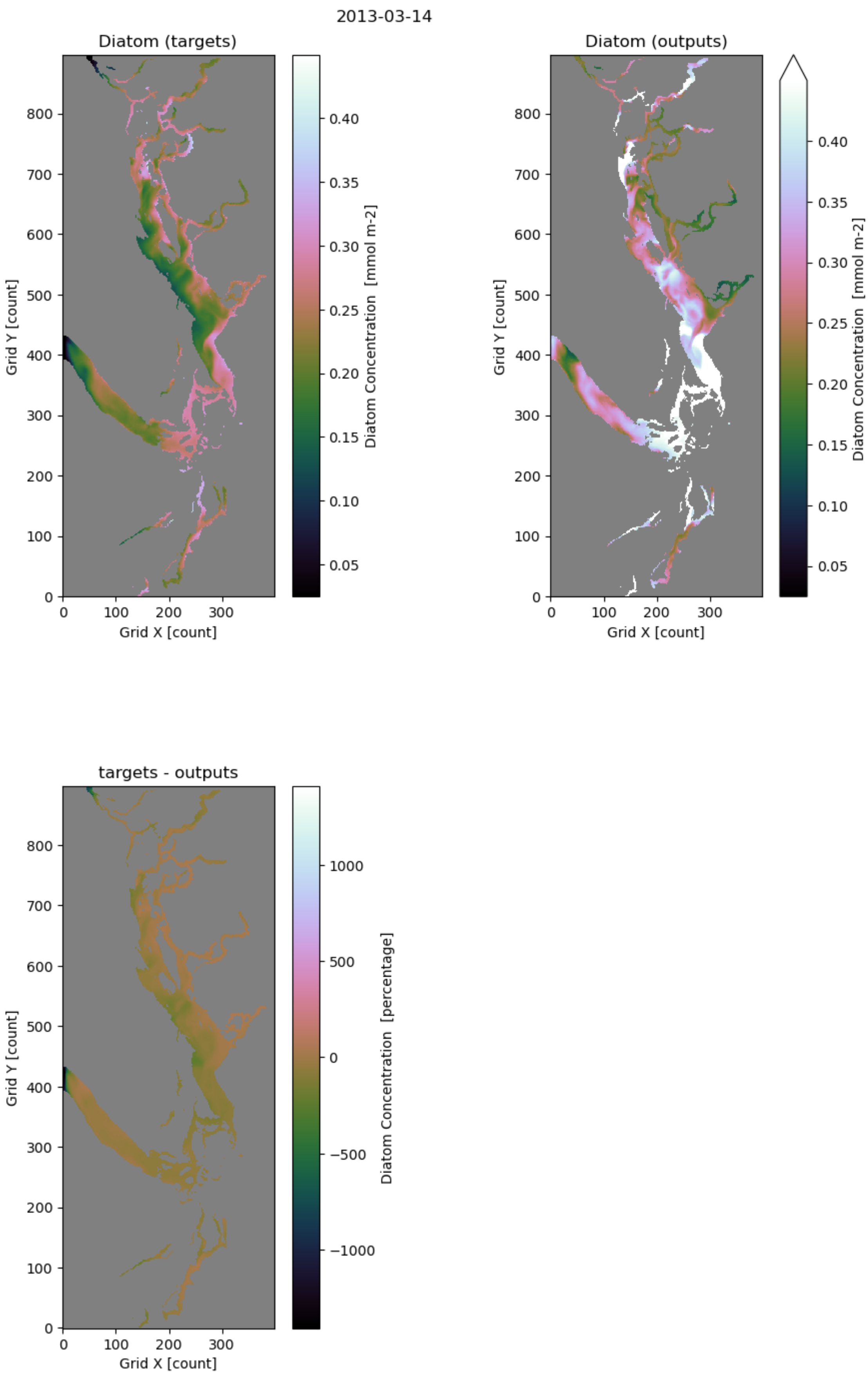

Diatom (Testing dataset)
outputs
targets
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
outputs
targets
10^0
10^1
10^2


2019-03-06
Diatom (targets)
Diatom (outputs)
targets - outputs
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
Diatom Concentration [percentage]

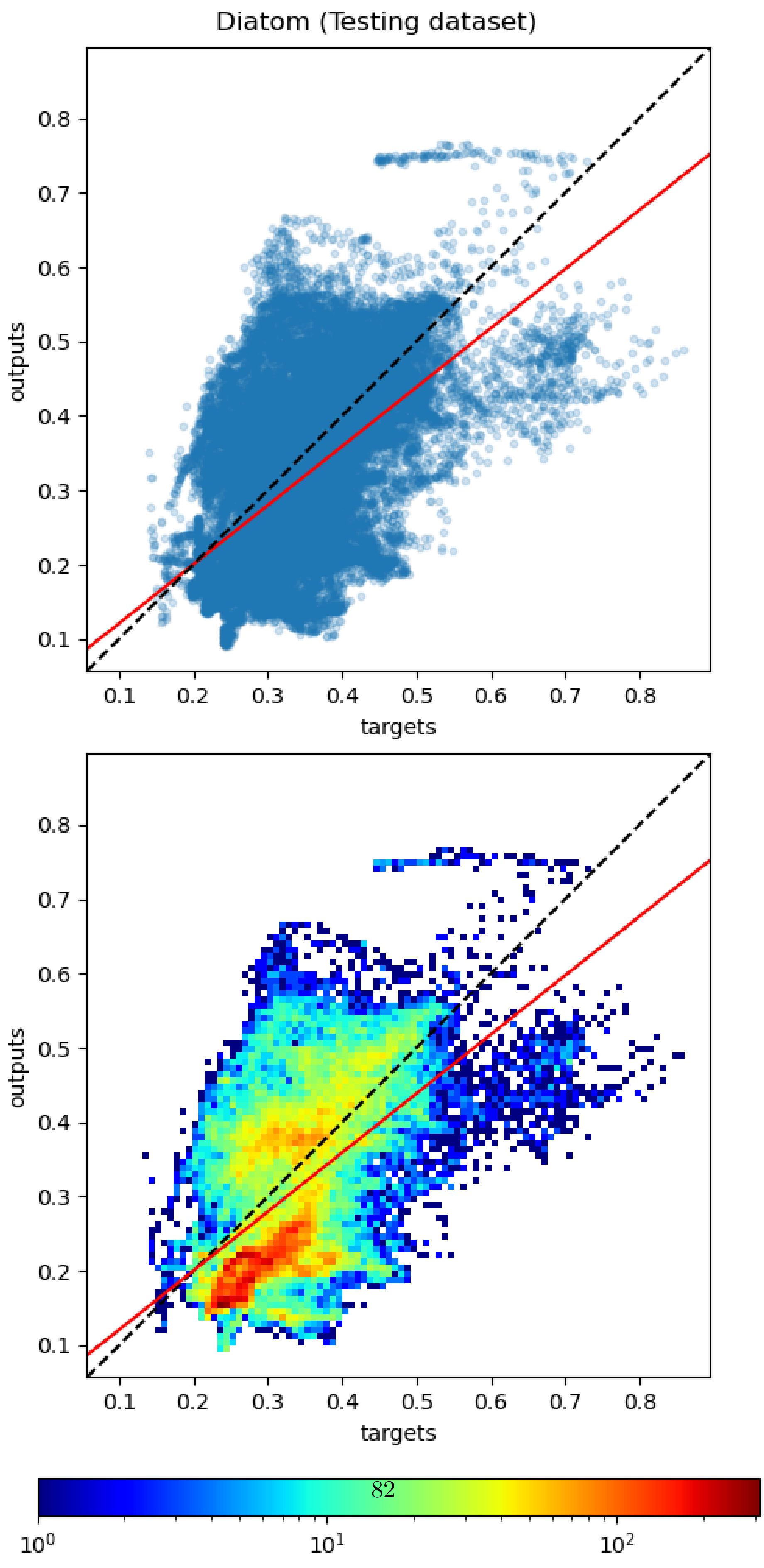
```
The amount of data points is 46479
The slope of the best fitting line is  0.622
The correlation coefficient is: 0.293
 The mean square error is: 0.02107
```

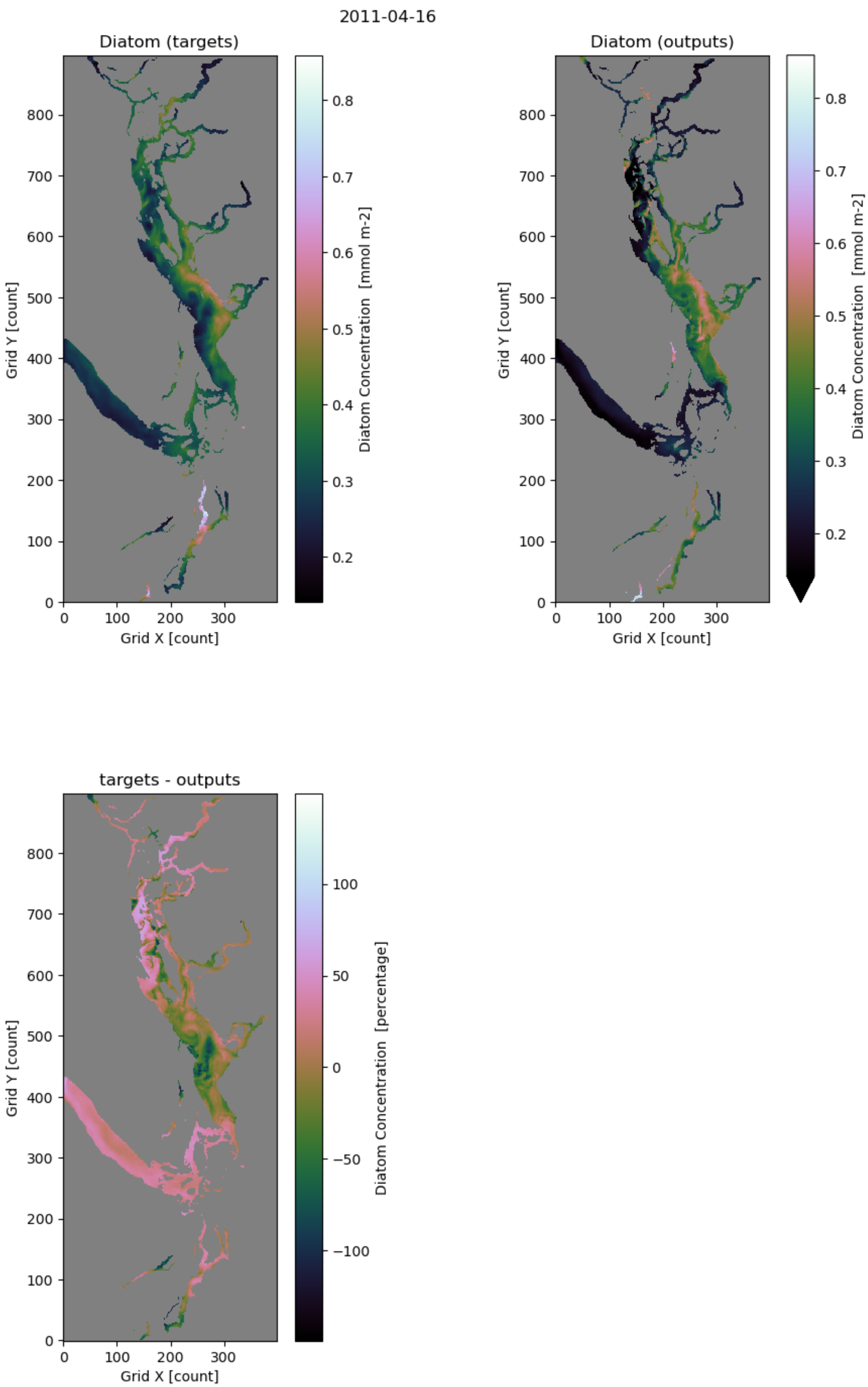
Diatom (Testing dataset)




2013-03-14
Diatom (targets)
Diatom (outputs)
targets - outputs
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
Diatom Concentration [percentage]
0
100
200
300
400
500
600
700
800
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
−1000
−500
500
1000

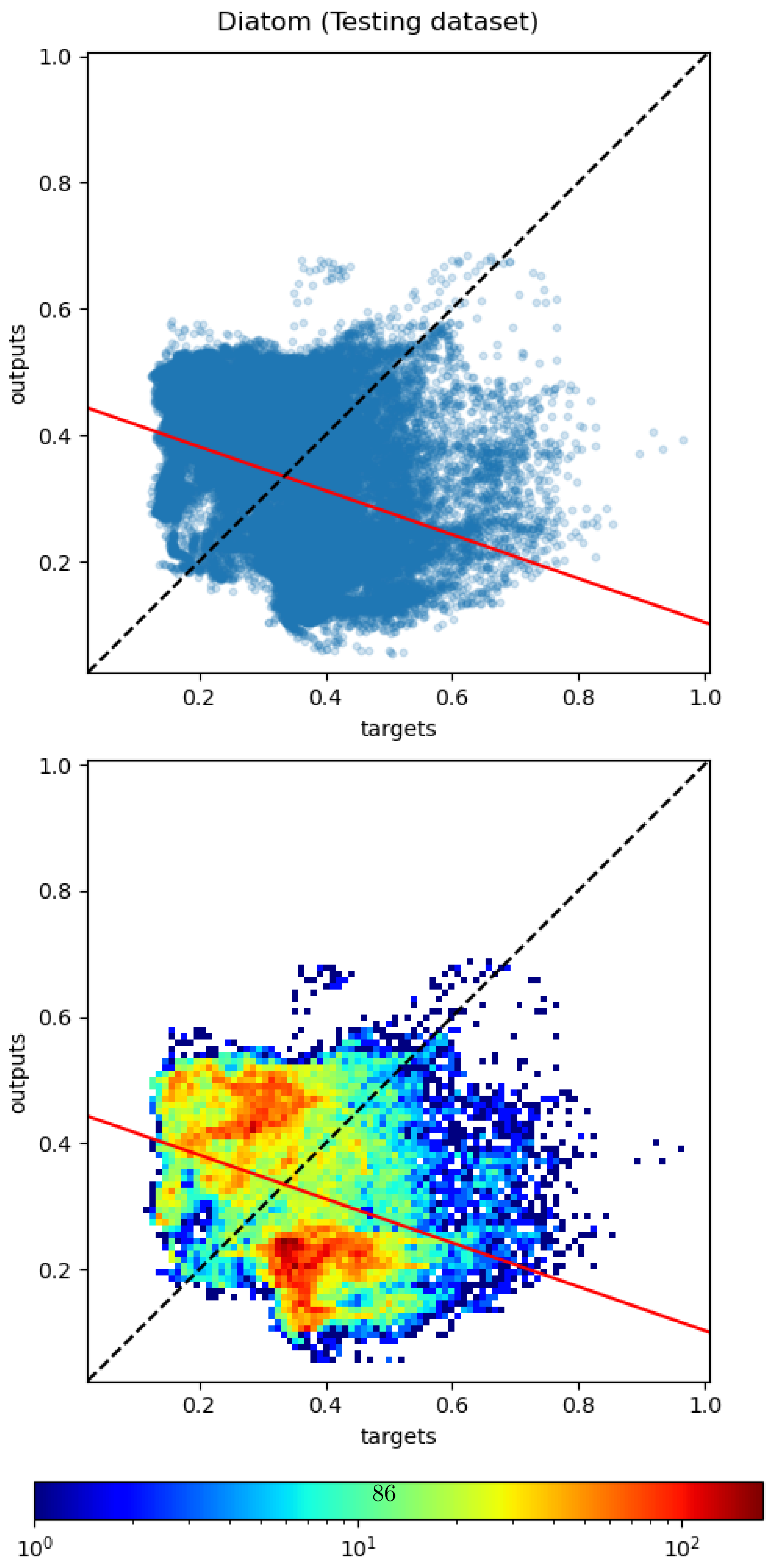
```
The amount of data points is 46479
The slope of the best fitting line is  0.794
The correlation coefficient is: 0.562
 The mean square error is: 0.01073
```

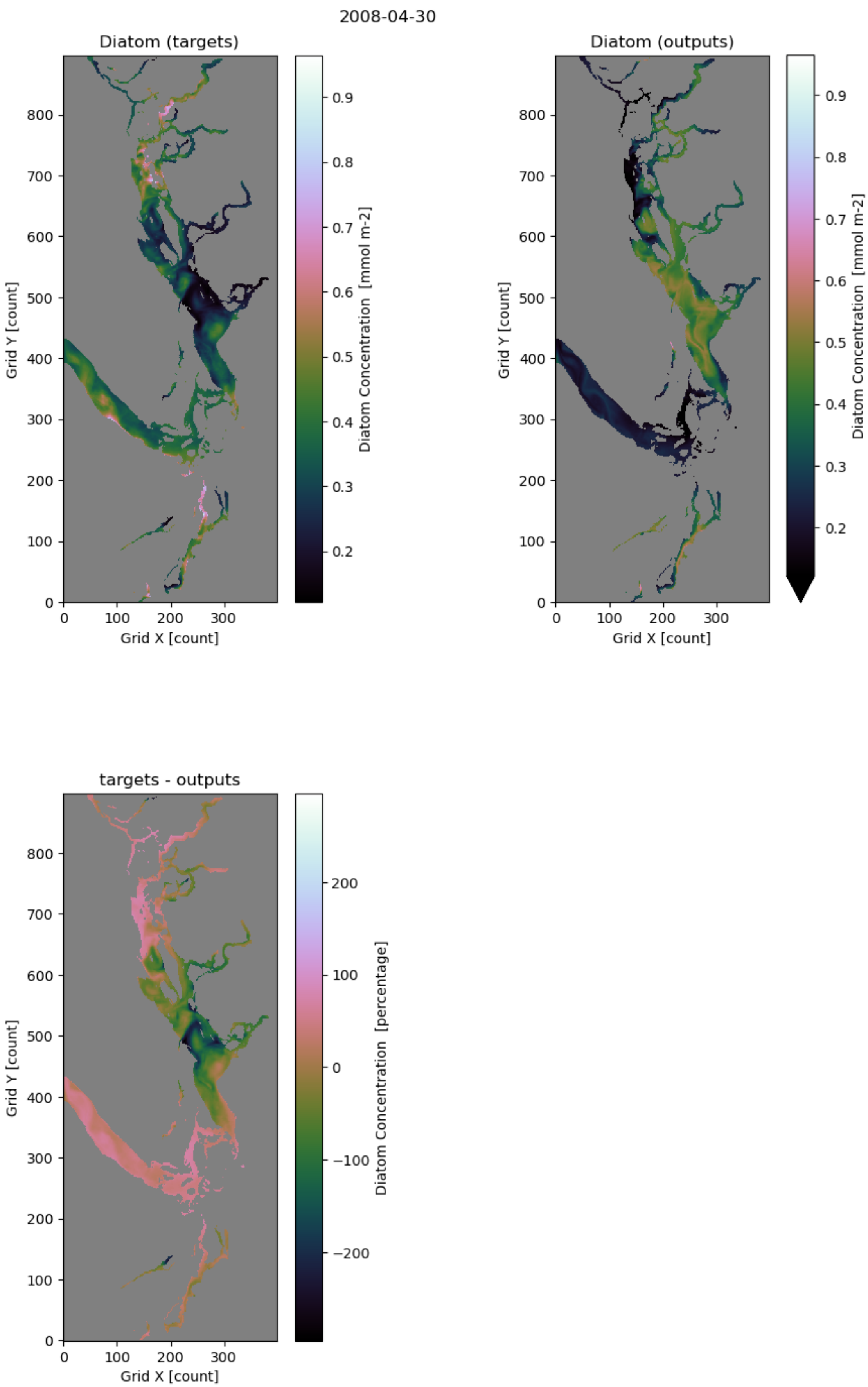

Diatom (Testing dataset)
outputs
targets
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
outputs
targets
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
10^0 10^1 10^2



2011-04-16
Diatom (targets)
Diatom (outputs)
targets - outputs
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
Diatom Concentration [percentage]

```
The amount of data points is 46479
The slope of the best fitting line is  -0.348
The correlation coefficient is: -0.322
 The mean square error is: 0.03655
```


Diatom (Testing dataset)
outputs
targets
0.2
0.4
0.6
0.8
1.0
outputs
targets
0.2
0.4
0.6
0.8
1.0
10^0
10^1
10^2


2008-04-30
Diatom (targets)
Diatom (outputs)
Grid Y [count]
Grid X [count]
Diatom Concentration [mmol m-2]
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
800
700
600
500
400
300
200
100
0
targets - outputs
Diatom Concentration [percentage]
200
100
0
−100
−200

[ ]:
```
dill.dump_session('test2.db')
```

[ ]:
```

```