reg_year_r_2014

January 31, 2024

0.1 Importing

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

from sklearn.ensemble import BaggingRegressor
from sklearn.tree import ExtraTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

from sklearn.metrics import mean_squared_error as mse

import os
```

0.2 Datasets Preparation (Training)

[ ]:
```
def datasets_preparation ():

    # Dataset and date
    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)

    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 (temp_i1, temp_i2, saline_i1, saline_i2, flag_i)
```

0.3 Regressor

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

    inputs = inputs.transpose()

    # Regressor
    scale = preprocessing.StandardScaler()
    inputs2 = scale.fit_transform(inputs)
    X_train, X_test, y_train, y_test = train_test_split(inputs2, targets)

    extra_tree = ExtraTreeRegressor(criterion='poisson')
    regr = BaggingRegressor(extra_tree, n_estimators=10, max_features=4).
↪fit(X_train, y_train)

    outputs_test = regr.predict(X_test)

    m = scatter_plot(y_test, outputs_test, variable_name + ' (Testing dataset)')
    r = np.round(np.corrcoef(y_test, outputs_test)[0][1],3)
    rms = np.round(mse(y_test, outputs_test),4)


    return (r, rms, m, 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()

    plt.scatter(targets,outputs, alpha = 0.2, s = 10)
    plt.xlabel('targets')
    plt.ylabel('outputs')

    lims = [
        np.min([ax.get_xlim(), ax.get_ylim()]),  # min of both axes
        np.max([ax.get_xlim(), ax.get_ylim()]),  # max of both axes
    ]

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

    ax.set_aspect('equal')
    ax.set_xlim(lims)
    ax.set_ylim(lims)

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

    fig.suptitle(str(year) + ', ' + 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 Regressor 2

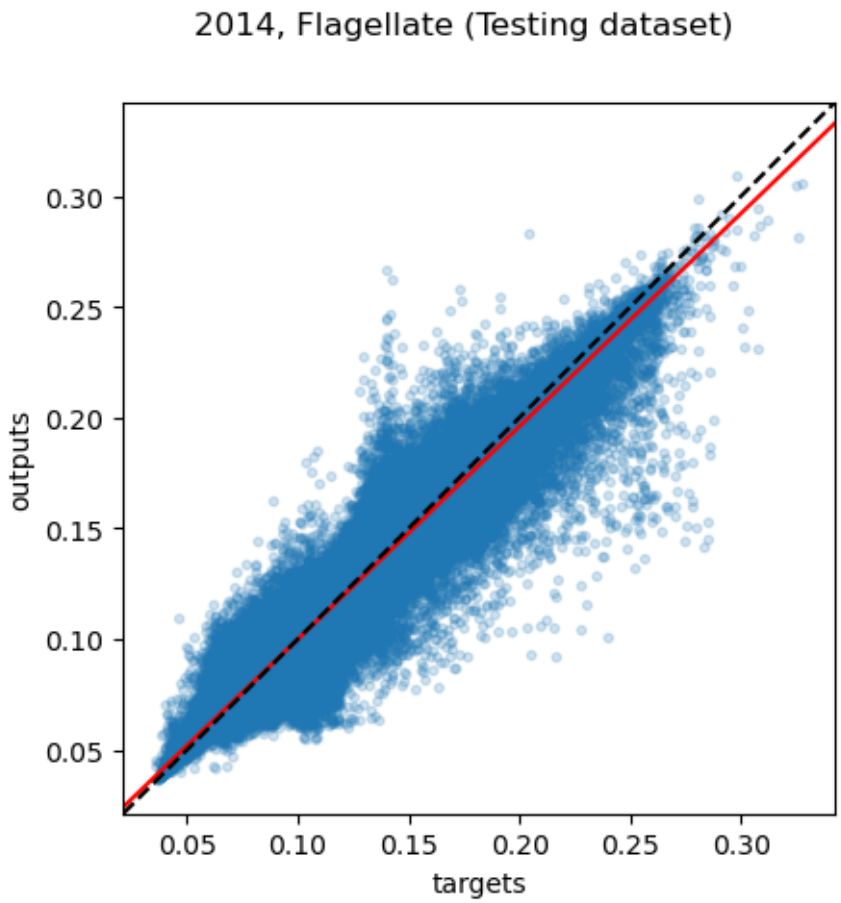
[ ]:
```
def regressor2 (inputs, targets, variable_name):

    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 + ' (Testing␣
 ↪dataset)')
    r = np.round(np.corrcoef(targets, outputs_test)[0][1],3)
    rms = np.round(mse(targets, outputs_test),4)

    return (r, rms, m)
```

1.4 Training of 2007

[ ]:
```
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()

year = 2014

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([[],[],[],[]])
flag_all = np.array([])

print ('Gathering days for year ' + str(year))

for i in folders:

    temp_i1, temp_i2, saline_i1, saline_i2, flag_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)

    flag = np.ravel(flag_i)
    flag = flag[indx[0]]
    flag_all = np.concatenate((flag_all,flag))

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

r, rms, m, regr = regressor(drivers_all, flag_all, 'Flagellate')
```

```
Gathering days for year 2014
Done gathering, building the prediction model


The amount of data points is 871482
The slope of the best fitting line is  0.962
The correlation coefficient is: 0.983
 The mean square error is: 3e-05
```

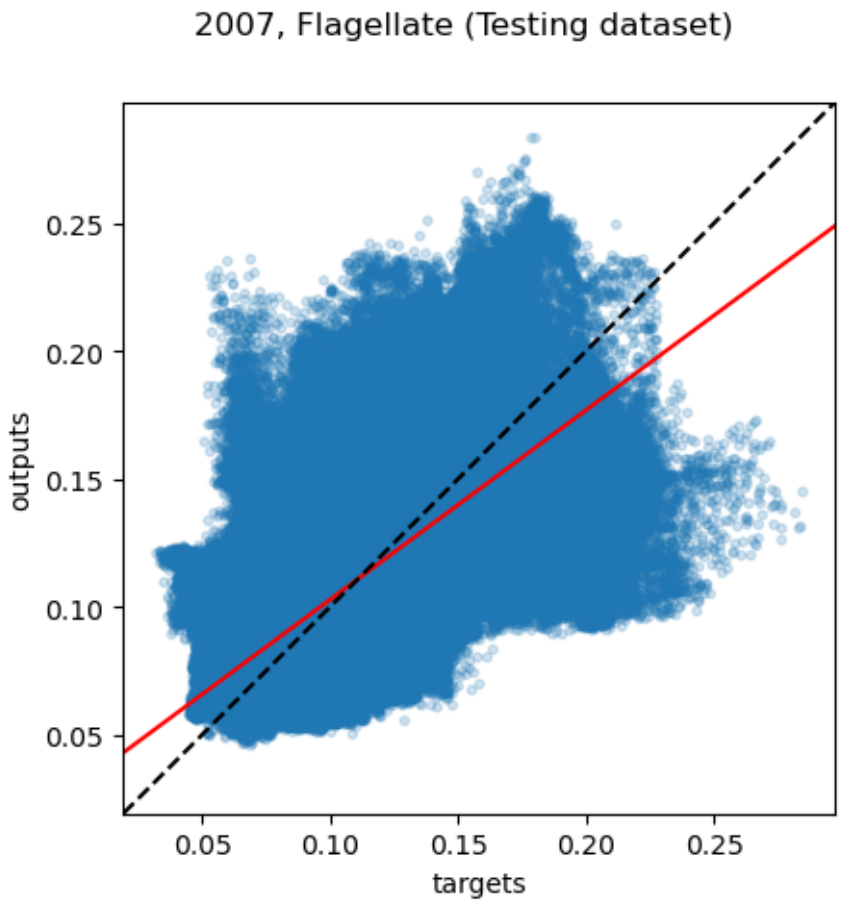


1.5 Other Years

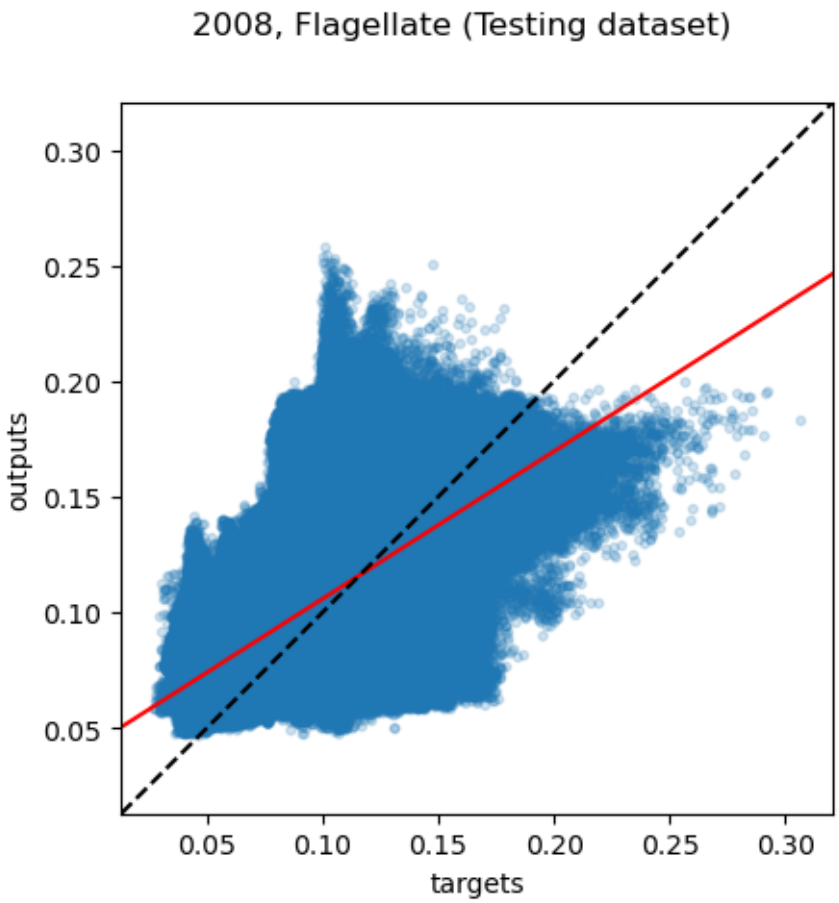
[ ]:
```
years = range (2007,2024)

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([[],[],[],[]])
    flag_all = np.array([])

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

        temp_i1, temp_i2, saline_i1, saline_i2, flag_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)

        flag = np.ravel(flag_i)
        flag = flag[indx[0]]
        flag_all = np.concatenate((flag_all,flag))

    r, rms, m = regressor2(drivers_all, flag_all, 'Flagellate')
    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')
```

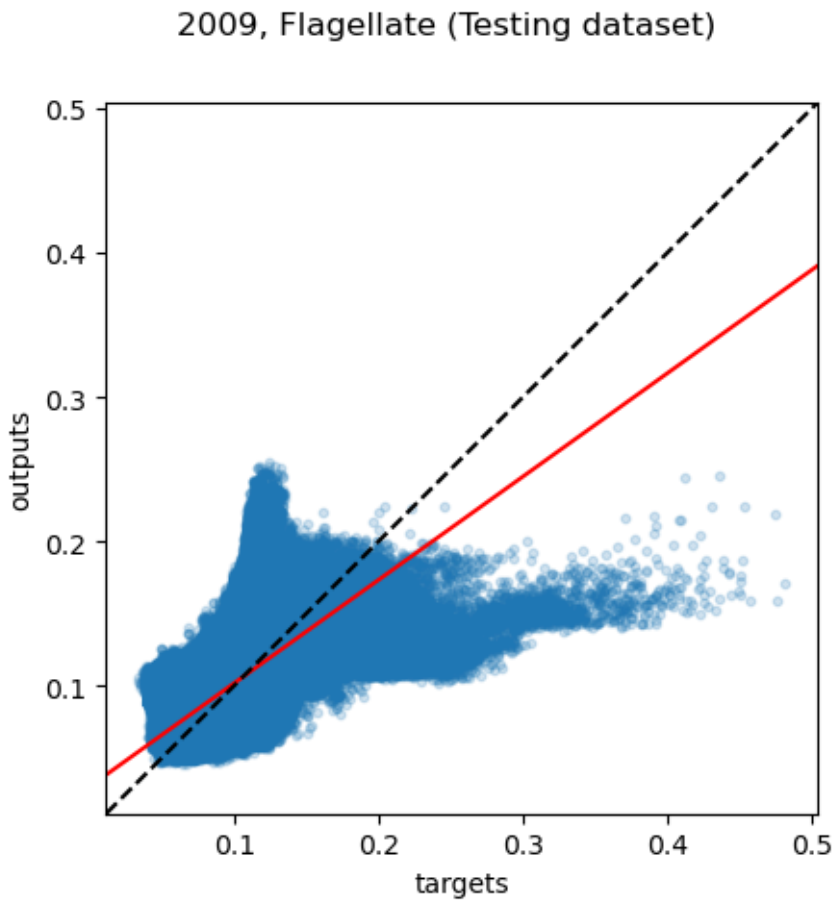
```
Gathering days for year 2007

The amount of data points is 3485925
The slope of the best fitting line is  0.741
The correlation coefficient is: 0.788
 The mean square error is: 0.00033
```



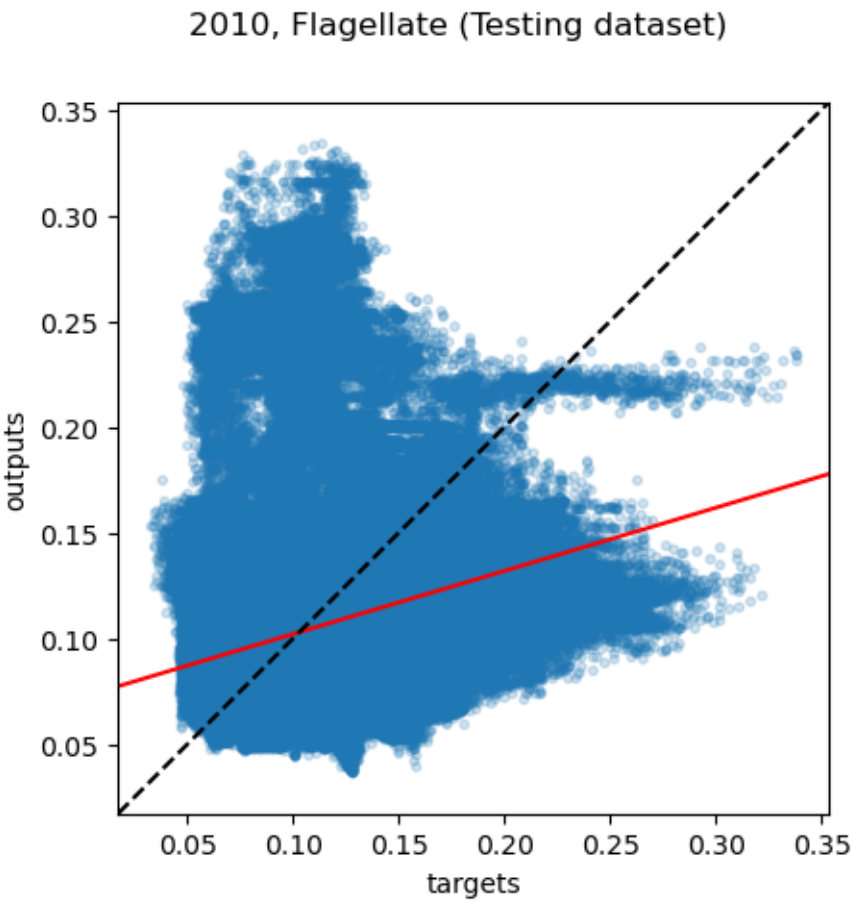

```
Gathering days for year 2008
The amount of data points is 3532404
The slope of the best fitting line is  0.638
The correlation coefficient is: 0.649
 The mean square error is: 0.00065
```



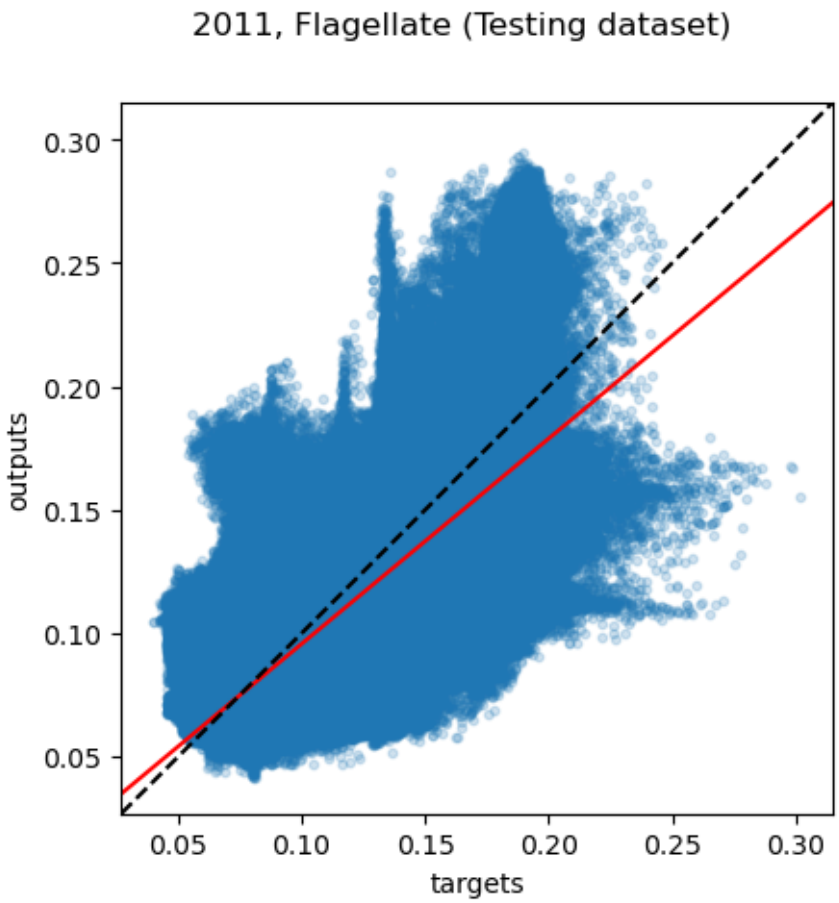

```
Gathering days for year 2009
The amount of data points is 3485925
The slope of the best fitting line is  0.716
The correlation coefficient is: 0.749
 The mean square error is: 0.00038
```



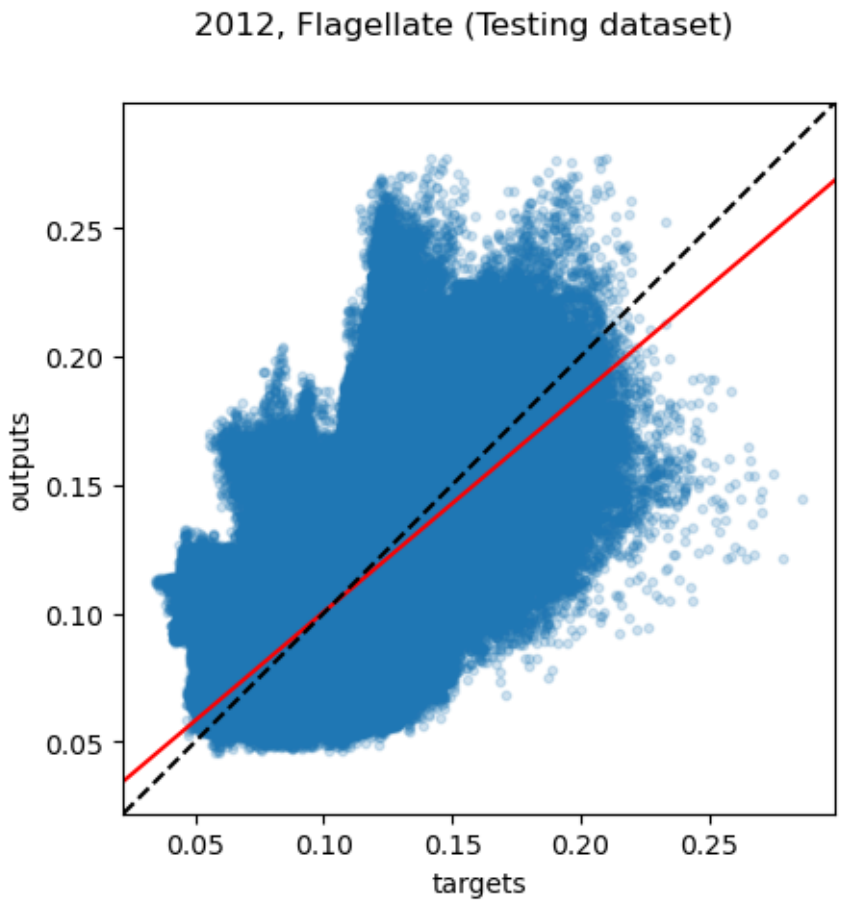

```
Gathering days for year 2010
The amount of data points is 3485925
The slope of the best fitting line is  0.299
The correlation coefficient is: 0.381
 The mean square error is: 0.00078
```



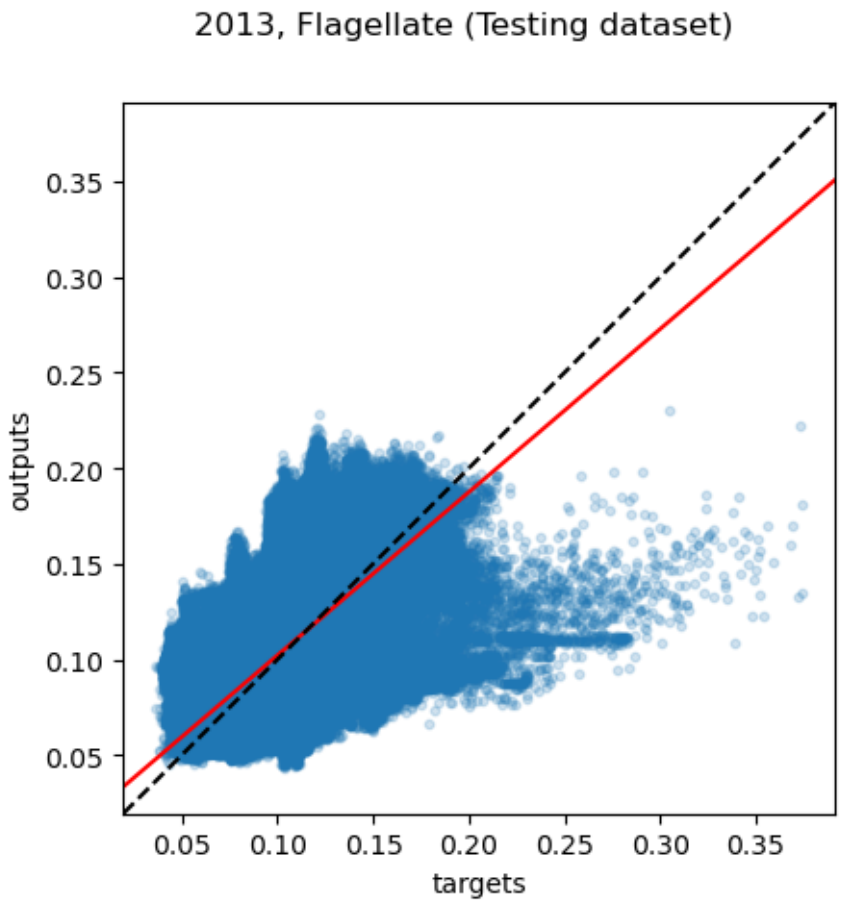

```
Gathering days for year 2011
The amount of data points is 3485925
The slope of the best fitting line is  0.832
The correlation coefficient is: 0.773
 The mean square error is: 0.00042
```



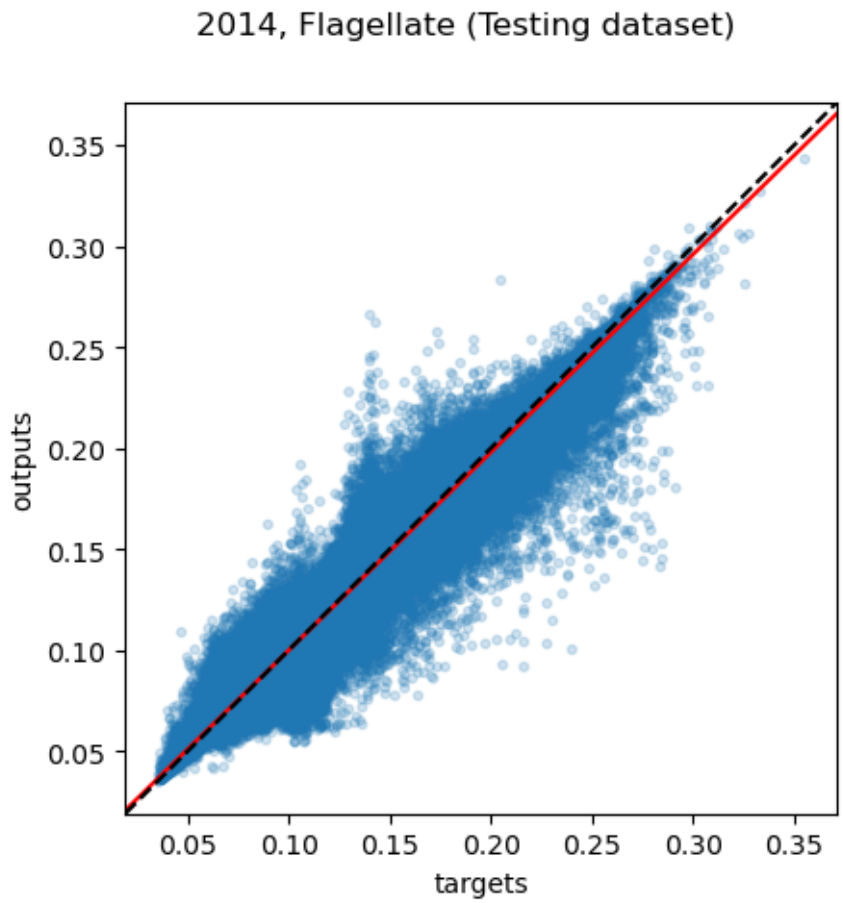

```
Gathering days for year 2012
The amount of data points is 3532404
The slope of the best fitting line is  0.845
The correlation coefficient is: 0.77
 The mean square error is: 0.00037
```



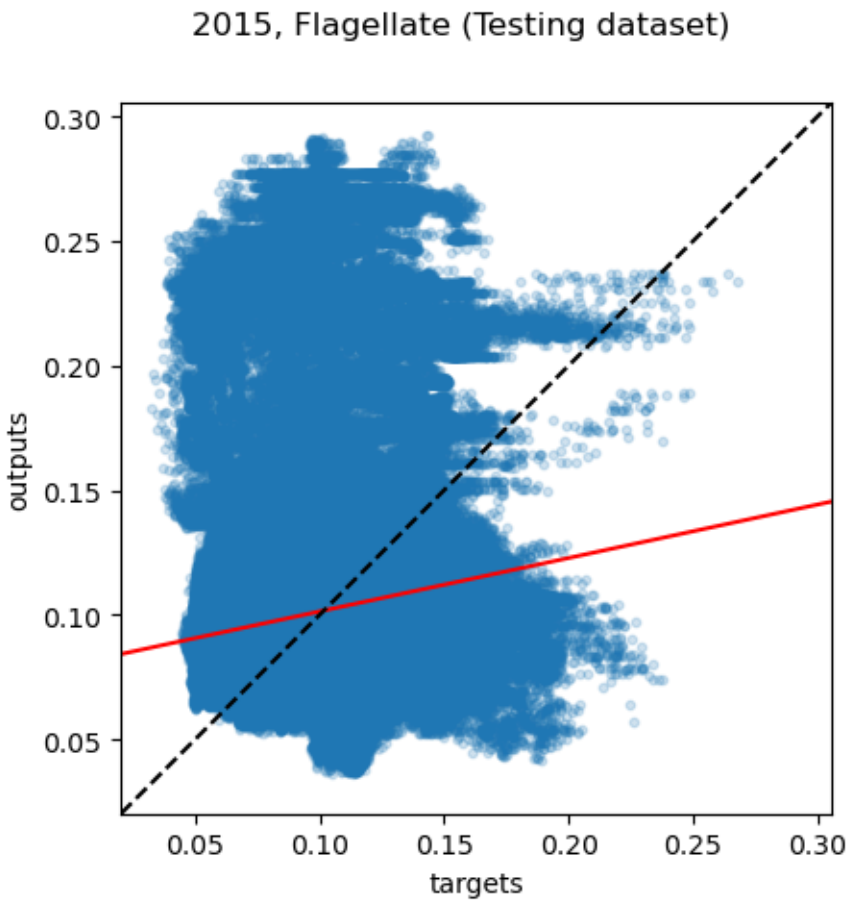

```
Gathering days for year 2013
The amount of data points is 3485925
The slope of the best fitting line is  0.853
The correlation coefficient is: 0.741
 The mean square error is: 0.00037
```



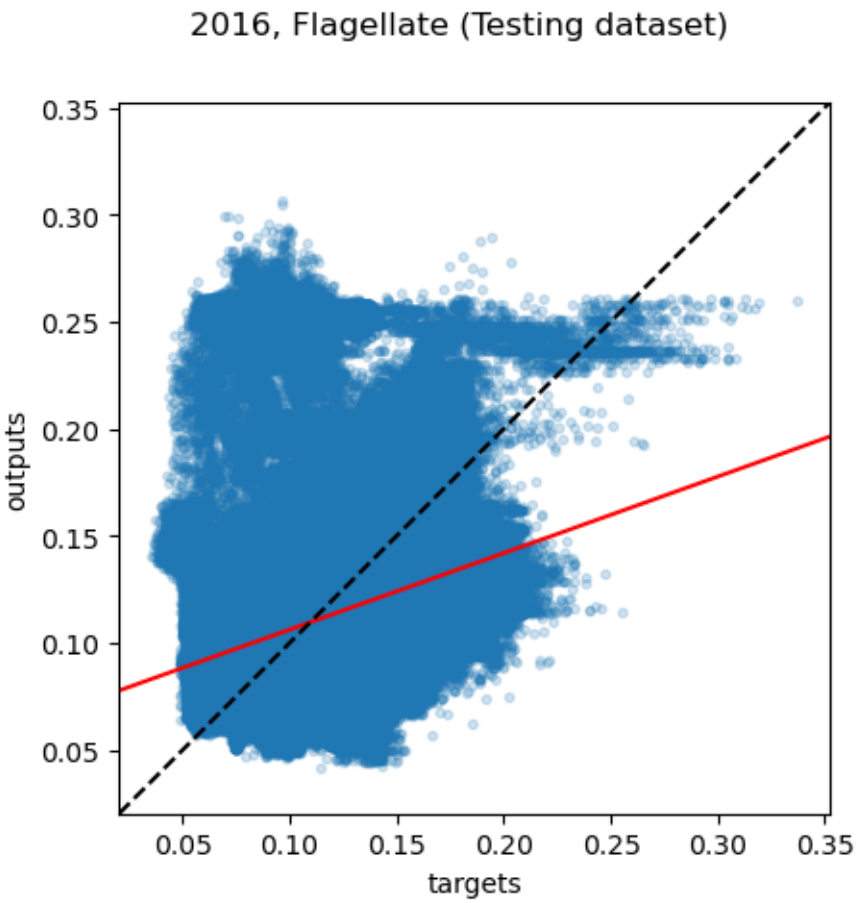

```
Gathering days for year 2014
The amount of data points is 3485925
The slope of the best fitting line is  0.98
The correlation coefficient is: 0.993
 The mean square error is: 1e-05
```



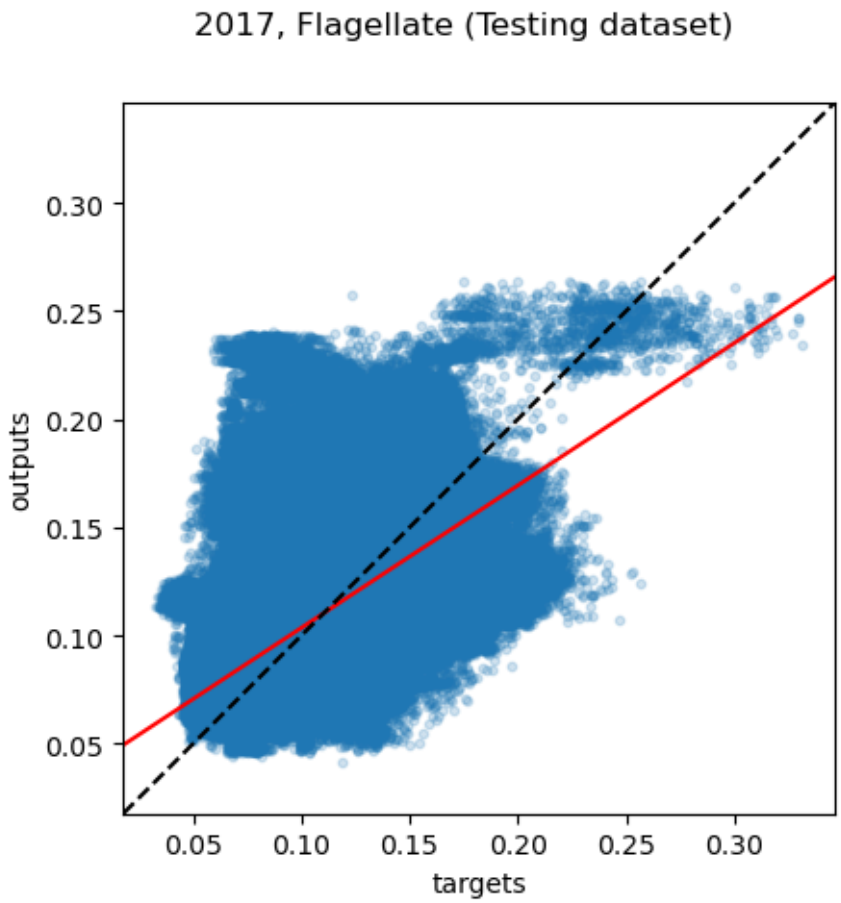

```
Gathering days for year 2015
The amount of data points is 3485925
The slope of the best fitting line is  0.214
The correlation coefficient is: 0.195
 The mean square error is: 0.00093
```



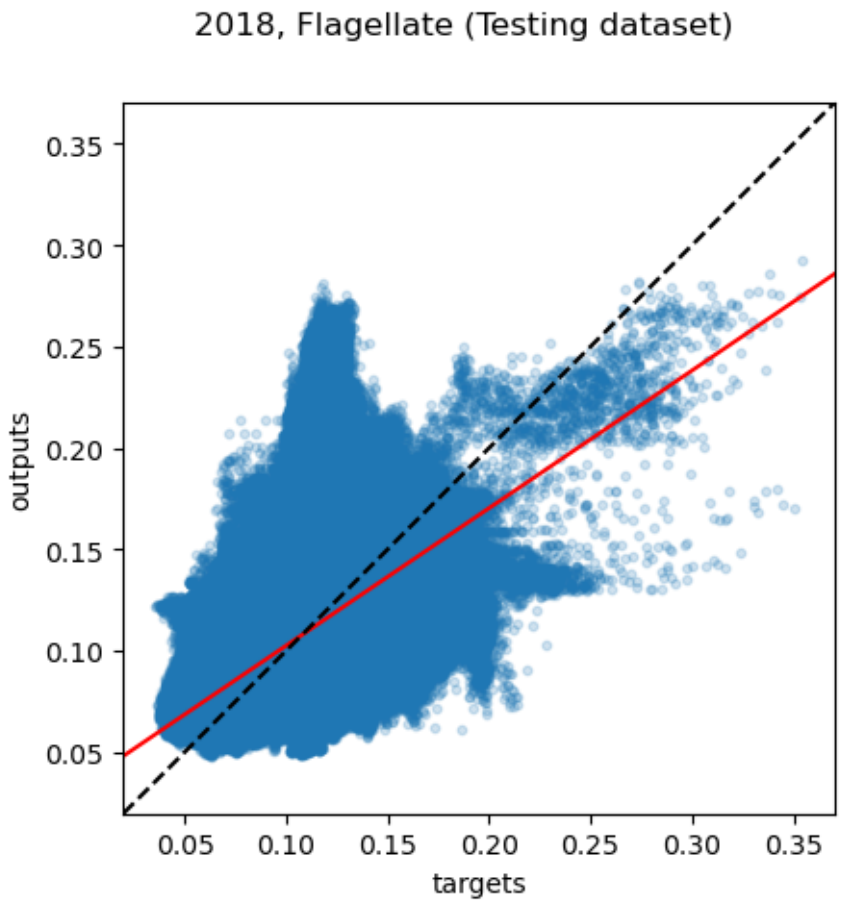

```
Gathering days for year 2016
The amount of data points is 3532404
The slope of the best fitting line is  0.357
The correlation coefficient is: 0.391
 The mean square error is: 0.00075
```



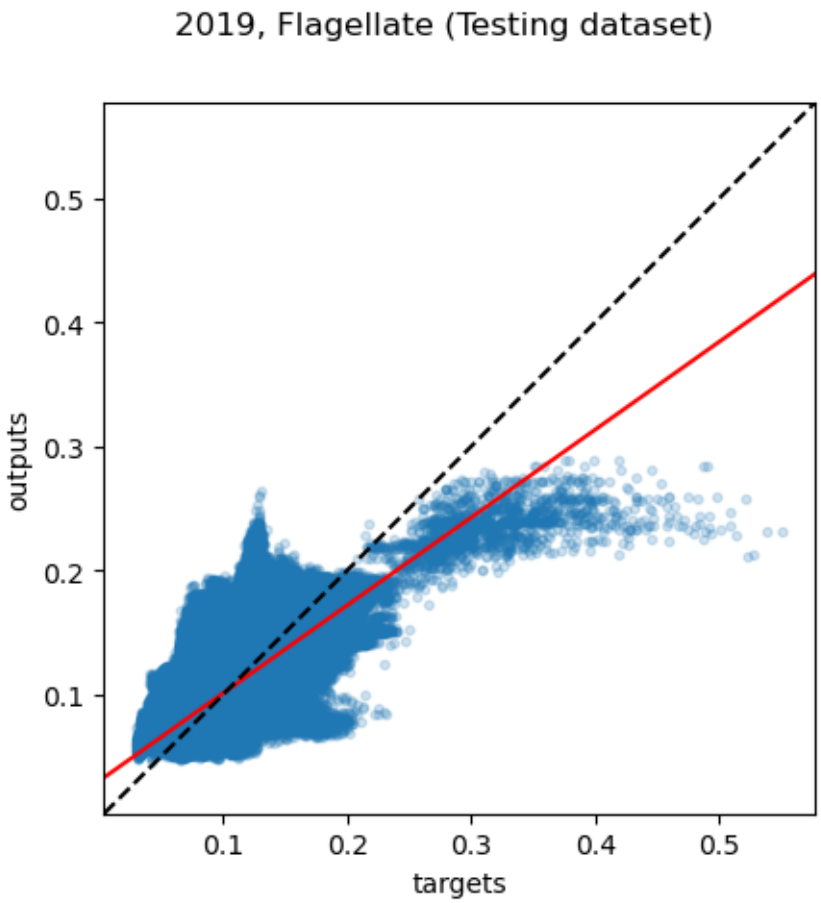

```
Gathering days for year 2017
The amount of data points is 3485925
The slope of the best fitting line is  0.658
The correlation coefficient is: 0.699
 The mean square error is: 0.0004
```



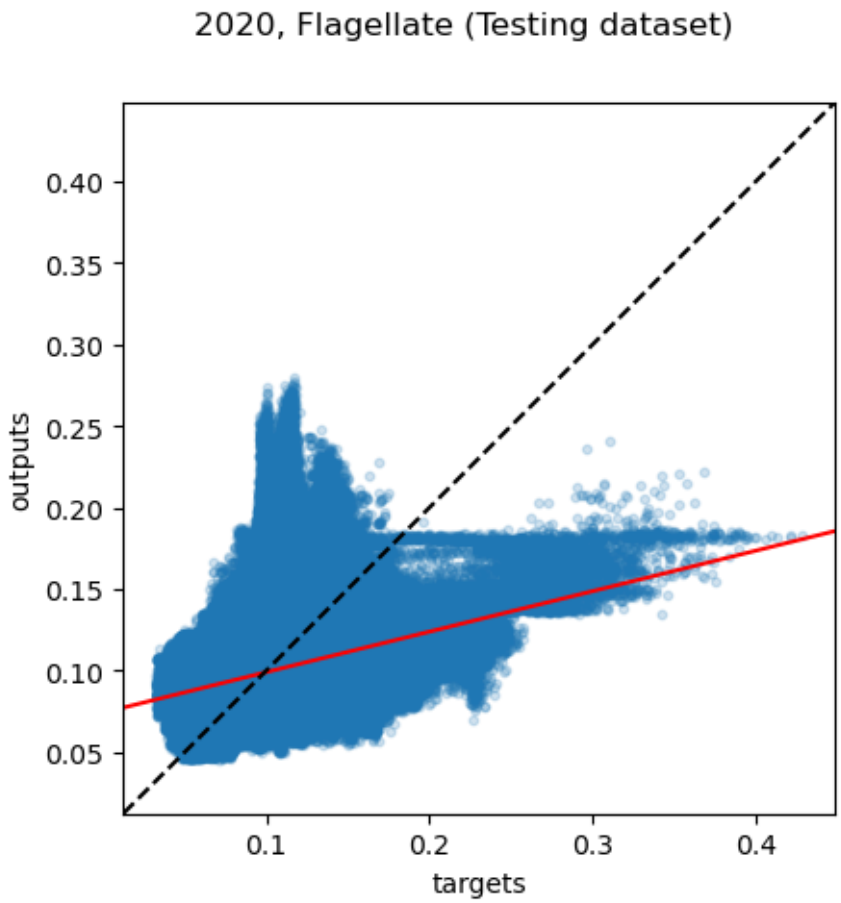

```
Gathering days for year 2018
The amount of data points is 3485925
The slope of the best fitting line is  0.678
The correlation coefficient is: 0.608
 The mean square error is: 0.00062
```



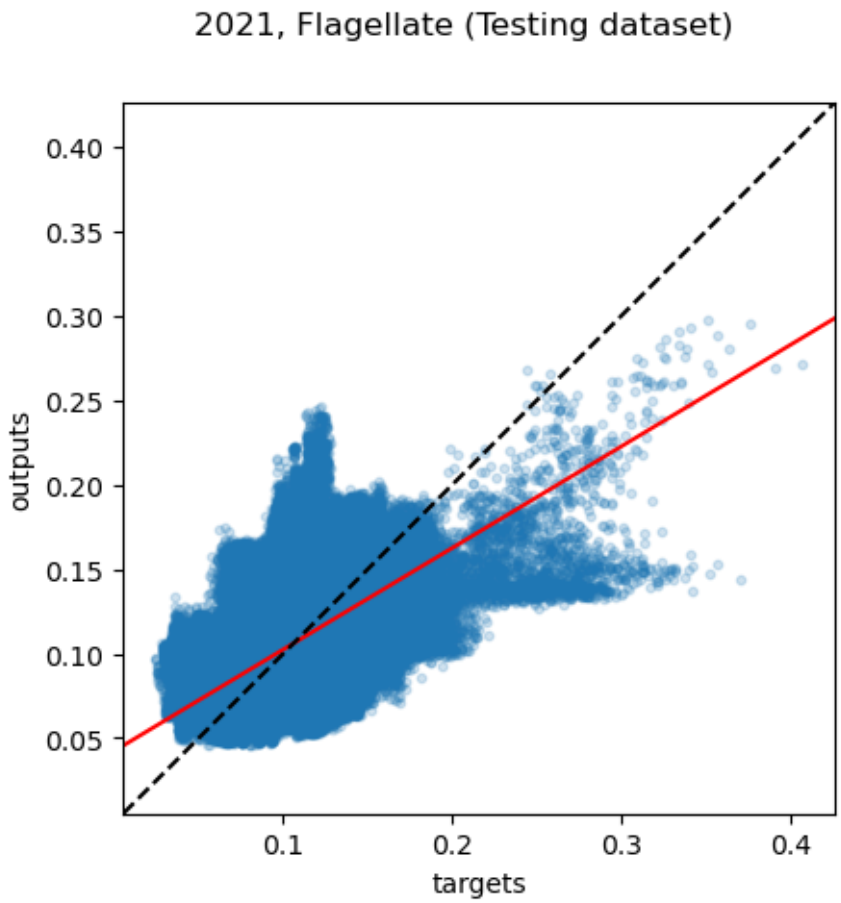

```
Gathering days for year 2019
The amount of data points is 3485925
The slope of the best fitting line is  0.708
The correlation coefficient is: 0.744
 The mean square error is: 0.00035
```



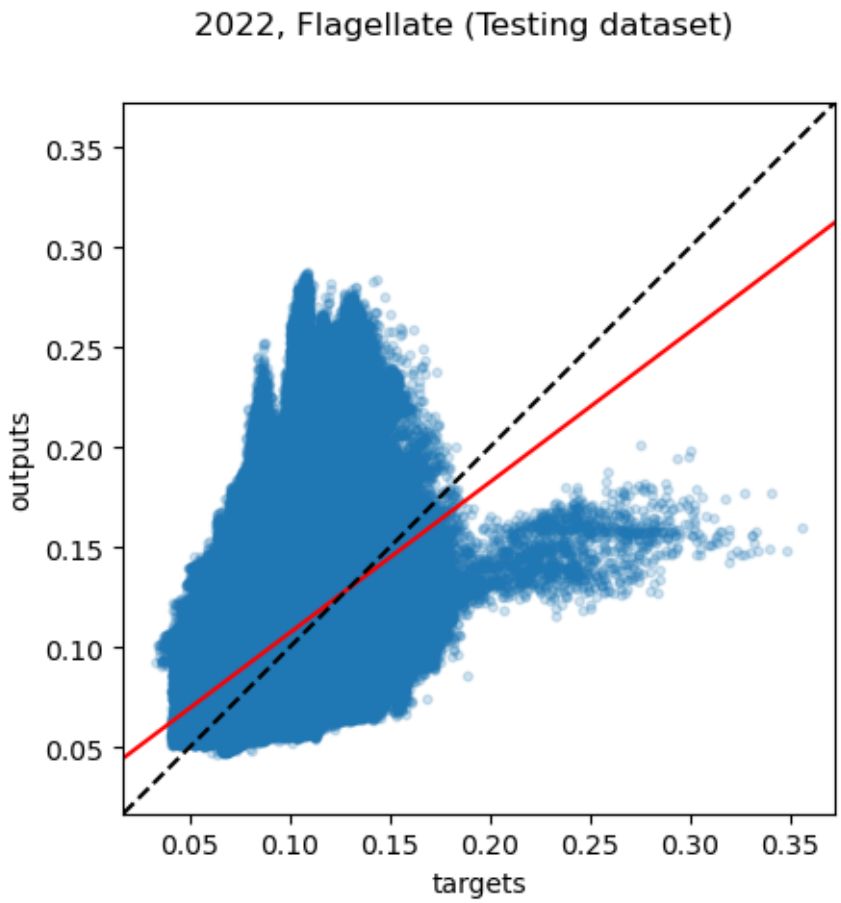

```
Gathering days for year 2020
The amount of data points is 3532404
The slope of the best fitting line is  0.248
The correlation coefficient is: 0.314
 The mean square error is: 0.0013
```



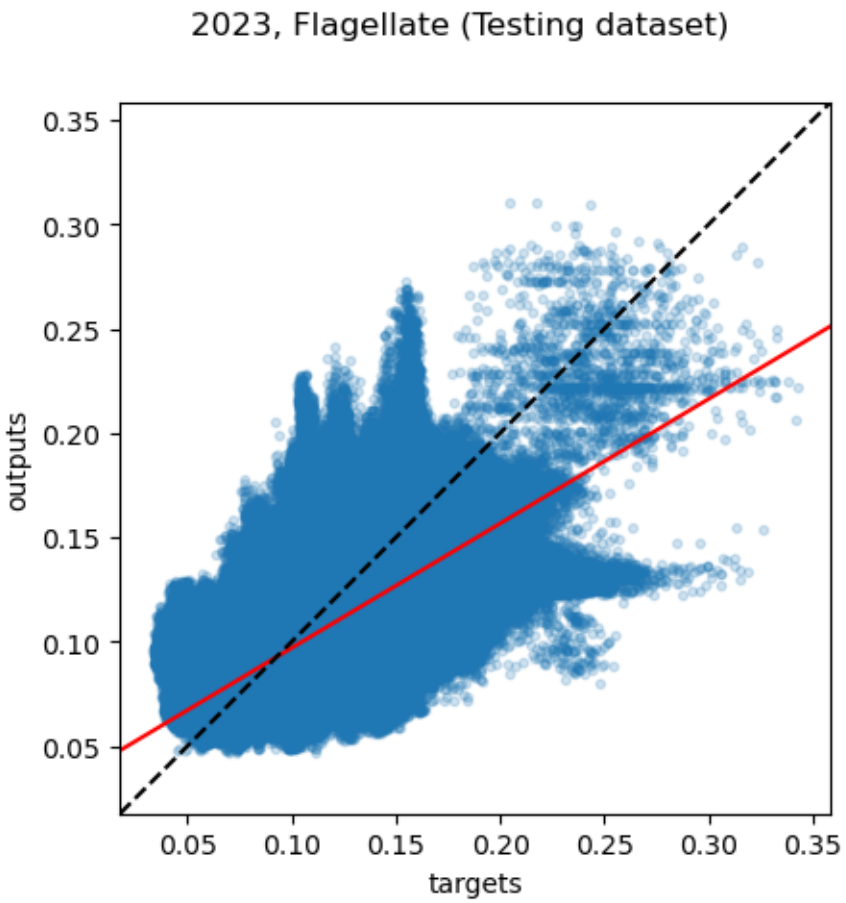

```
Gathering days for year 2021
The amount of data points is 3485925
The slope of the best fitting line is  0.601
The correlation coefficient is: 0.672
 The mean square error is: 0.00046
```

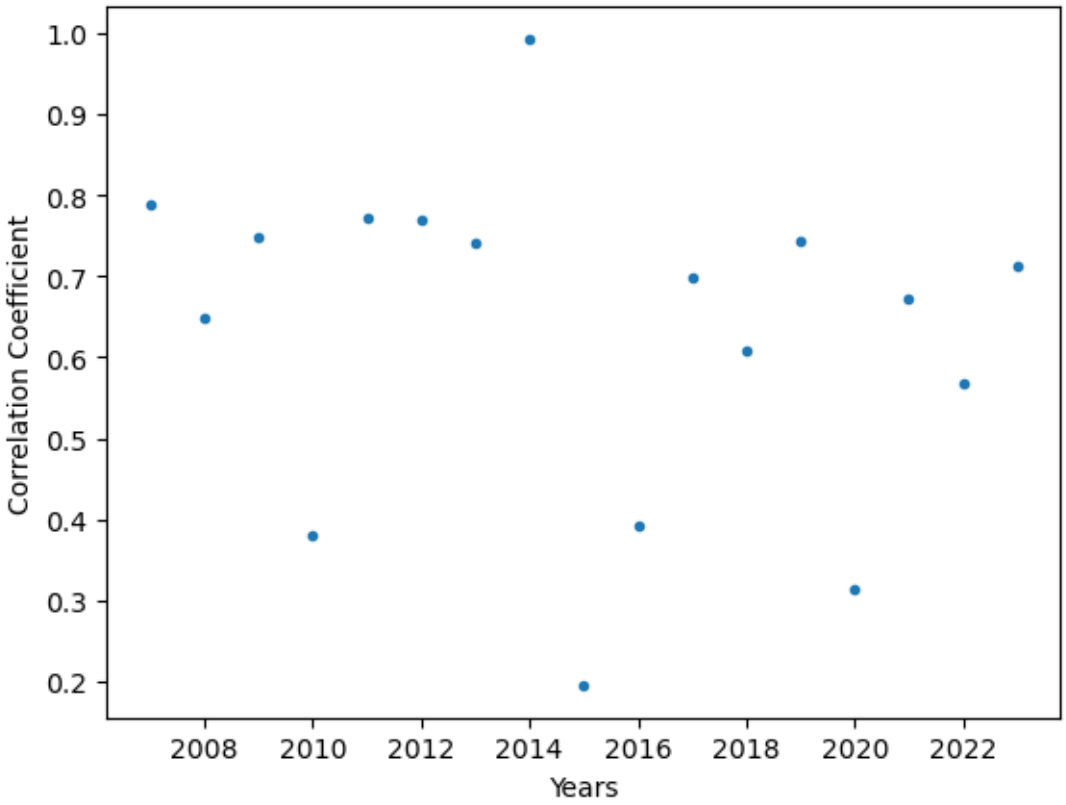


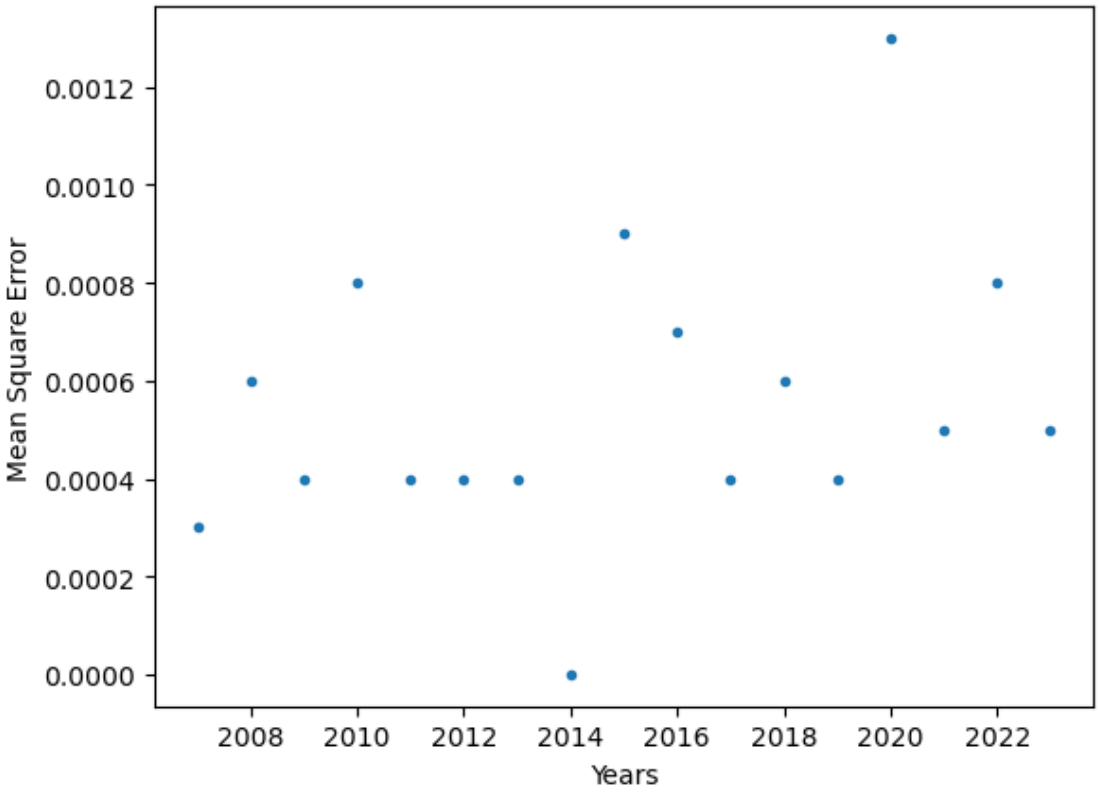

```
Gathering days for year 2022
The amount of data points is 3485925
The slope of the best fitting line is  0.753
The correlation coefficient is: 0.568
 The mean square error is: 0.00076
```

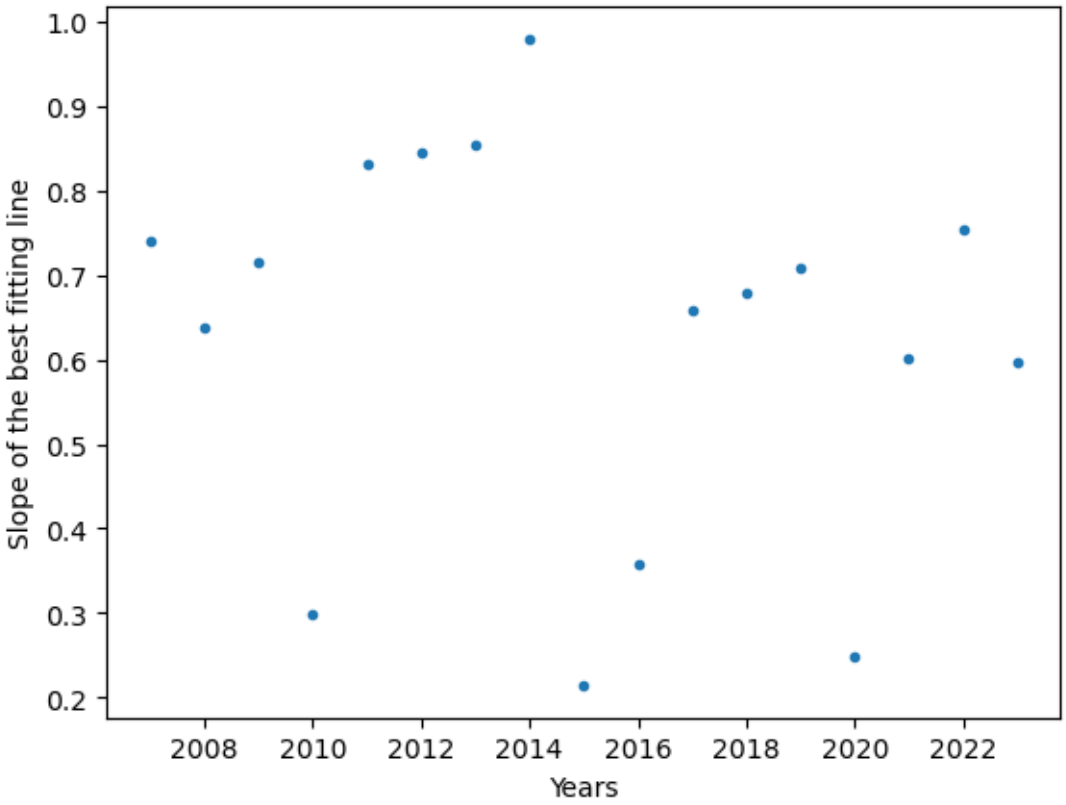



```
Gathering days for year 2023
The amount of data points is 3485925
The slope of the best fitting line is  0.597
The correlation coefficient is: 0.712
 The mean square error is: 0.00052
```


2023, Flagellate (Testing dataset)
outputs
targets
0.05
0.10
0.15
0.20
0.25
0.30
0.35


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Correlation Coefficient
2008
2010
2012
2014
2016
2018
2020
2022
Years


Mean Square Error
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
2008
2010
2012
2014
2016
2018
2020
2022
Years


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Slope of the best fitting line
2008
2010
2012
2014
2016
2018
2020
2022
Years