In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
```

In [2]:
```
df = pd.read_csv("CH06PR09.txt", sep='\s+', header=None, names = ['Y', 'X1', 'X2', 'X3'])
df.head()
```

Out[2]:

	Y	X1	X2	X3
0	4264	305657	7.17	0
1	4496	328476	6.20	0
2	4317	317164	4.61	0
3	4292	366745	7.02	0
4	4945	265518	8.61	1

In [3]:
```
X1 = df['X1']
X2 = df['X2']
X3 = df['X3']
Y = df['Y']
n = len(X1)
X = df.iloc[:,1:4]
X = sm.add_constant(X)
X.head()
```

Out[3]:

	const	X1	X2	X3
0	1.0	305657	7.17	0
1	1.0	328476	6.20	0
2	1.0	317164	4.61	0
3	1.0	366745	7.02	0
4	1.0	265518	8.61	1

a. Obtain the studentized deleted residuals and identify any outlying Y observations. Use the Bonferroni outlier test procedure with α = .05. State the decision rule and conclusion.

In [4]:
```
#a
X_trans = np.transpose(X)
X_transX = X_trans.dot(X)
X_transY = X_trans.dot(Y)
Invert = np.linalg.pinv(X_transX)
bb = Invert.dot(X_transY)
Y_hat = X.dot(bb)
J = np.ones((len(X1), len(X1)))
b_trans = np.transpose(bb)
resid = Y - Y_hat
Y_trans = np.transpose(Y)
Y_transY = Y_trans.dot(Y)
Y_transJ = Y_trans.dot(J)
YJY = Y_transJ.dot(Y)
b_transXY = b_trans.dot(X_transY)
SSTO = Y_transY - ((1/len(X1))*YJY)
SSE = Y_transY - b_transXY
MSE = SSE / (len(X1)-4)
print(bb)
```

```
[ 4.14988724e+03  7.87080357e-04 -1.31660218e+01  6.23554481e+02]
```

In [7]:
```
ti = resid * (((len(X1) - 4 - 1)/(SSE*(1-h)-(resid**2)))**(1/2))
ti
```

Out[7]:
```
0    -0.224087
1     1.225490
2    -0.170589
3    -0.384653
4     0.590792
5     0.196124
6    -1.112209
7    -1.205293
8    -0.973171
9     2.036518
10    0.934595
11   -0.237756
12   -0.415163
13   -1.575636
14    0.161777
15   -0.945855
16    1.275712
17   -0.539465
18    0.766954
19   -1.306883
20   -0.378669
21    0.093487
22   -0.861994
23    0.582044
24    0.347378
25   -0.164277
26    0.553953
27   -0.416946
28    0.432226
29   -0.163818
30   -0.987935
31   -1.997667
32    1.584030
33    1.700417
34   -1.666863
35   -0.485481
36   -0.987260
37    2.118786
38   -0.774014
39    2.178272
40    0.205354
41    0.596602
42    0.885566
43    0.432470
44    0.276805
45   -0.566903
46   -1.046822
47   -0.234437
48    0.019097
49    1.630205
50   -1.374705
51    0.452790
dtype: float64
```

In [8]:
```
# Bonferroni outlier test procedure: t(1 − α/2n; n − p − 1)
n = len(X1)
p = 4
alpha = 0.05
t = stats.t.ppf(1 - alpha/(2*n), n - p - 1)
t
```

Out[8]:
```
3.5230801924879818
```

In [9]:
```
for i in range(52):
    if ti[i] > t:
        print(b[i])
else:
    print('|ti|  always lower than t')
```

```
|ti|  always lower than t
```

Conclusion a: Since |ti| always lower than t, we conclude that there is no outlier case.

b. Obtain the diagonal element of the hat matrix. Identify any outlying X observations.

In [10]:
```
from statsmodels.stats.outliers_influence import OLSInfluence
model = sm.OLS(Y, X, data = df)
results = model.fit()
results.summary()
test_class = OLSInfluence(results)
dir(test_class)
h = test_class.hat_matrix_diag
aa = []
```

```
D:\DS\lib\site-packages\statsmodels\base\model.py:127: ValueWarning: unknown kwargs ['data']
  warnings.warn(msg, ValueWarning)
```

In [11]:
```
for i in h:
    aa.append(i)
print(aa)
```

```
[0.022584972983419285, 0.06179962851795298, 0.21887725560625504, 0.052973224067106764, 0.20632818139275616, 0.027122124957388097, 0.02861964085078189, 0.056352638079748496, 0.03
0.04017169201929552, 0.04826901220735319, 0.030116336059819268, 0.04977032683051244, 0.027611338685290004, 0.06047245505016327, 0.037564476227039595, 0.2554249265491989, 0.03
32496523937486, 0.051049346263582596, 0.025617582395154888, 0.024918809352843, 0.19360471980053706, 0.25771994606470194, 0.05677233325138955, 0.07959048797955831, 0.056133
007076587485, 0.021894405105021122, 0.026972796456825505, 0.06097480605274354, 0.03684680871145793, 0.041746583591 6911, 0.0363400753622005, 0.09602318070259686, 0.0419329222
1342251, 0.025178373267151125, 0.046210567553352414, 0.0662222546807916, 0.03108517317171193, 0.03204566211164102, 0.049032485077924876, 0.03210501778122068, 0.04373193469233
92, 0.12395570934162886, 0.286858607403551, 0.22002363060530608, 0.11050576768481896, 0.031594264538048394, 0.06494376502143251, 0.281776644377407, 0.0244669219027156, 0.0342
0197309076067, 0.10278141584657355, 0.027540926920099337]
```

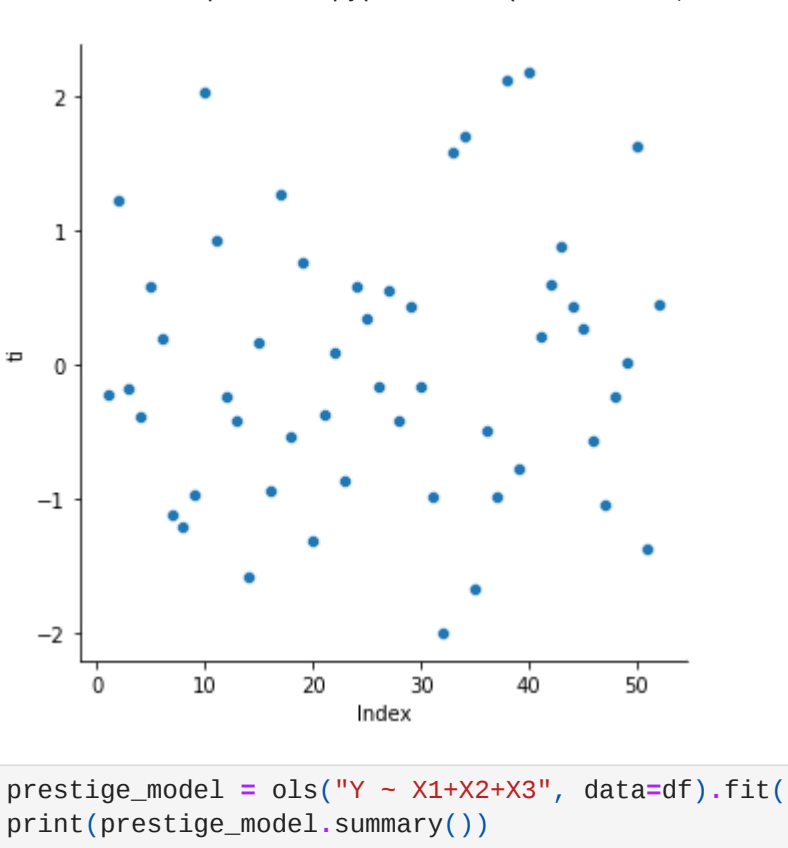
In [14]:
```
b = np.arange(1,53)
df2 = pd.DataFrame({'Index':b, 'H_matrix':h, 'ti':ti})
df2.head()
```

Out[14]:

	Index	H_matrix	ti
0	1	0.022585	-0.224087
1	2	0.061800	1.225490
2	3	0.218877	-0.170589
3	4	0.052973	-0.384653
4	5	0.206328	0.590792

In [15]:
```
sns.relplot(x='Index', y='ti', data=df2)
plt.show
```

Out[15]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



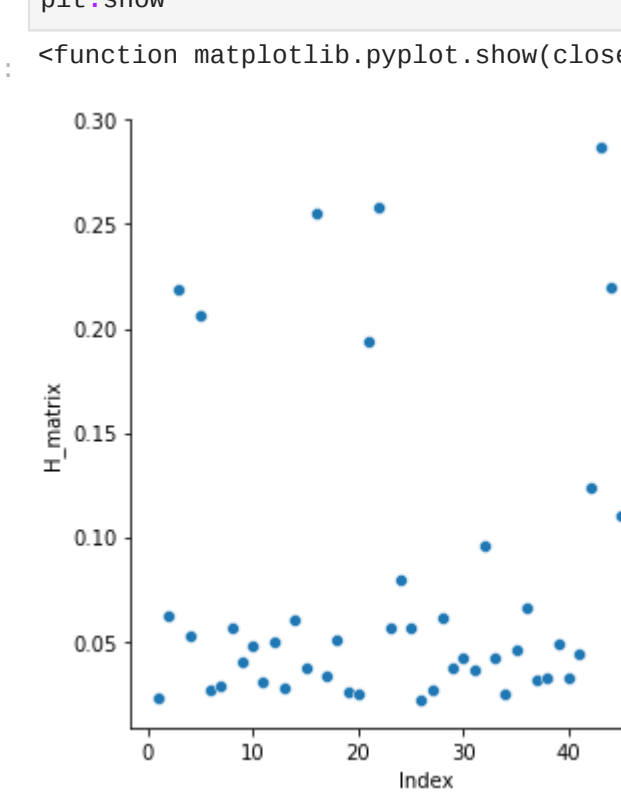
In [16]:
```
prestige_model = ols("Y ~ X1+X2+X3", data=df).fit()
print(prestige_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      Y   R-squared:                       0.688
Model:                            OLS   Adj. R-squared:                  0.669
Method:                 Least Squares   F-statistic:                     35.34
Date:                Mon, 05 Dec 2022   Prob (F-statistic):           3.32e-12
Time:                        22:14:13   Log-Likelihood:                -329.88
No. Observations:                  52   AIC:                             667.8
Df Residuals:                      48   BIC:                             675.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   4149.8872    195.565     21.220      0.000    3756.677    4543.098
X1             0.0008      0.000      2.159      0.036    5.41e-05       0.002
X2           -13.1660     23.092     -0.570      0.571     -59.595      33.263
X3           623.5545     62.641      9.954      0.000     497.606     749.503
==============================================================================
Omnibus:                        1.532   Durbin-Watson:                   2.298
Prob(Omnibus):                  0.465   Jarque-Bera (JB):                1.504
Skew:                           0.332   Prob(JB):                        0.471
Kurtosis:                       2.496   Cond. No.                     3.04e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.04e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [17]:
```
sns.relplot(x='Index', y='H_matrix', data=df2)
plt.show
```

Out[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



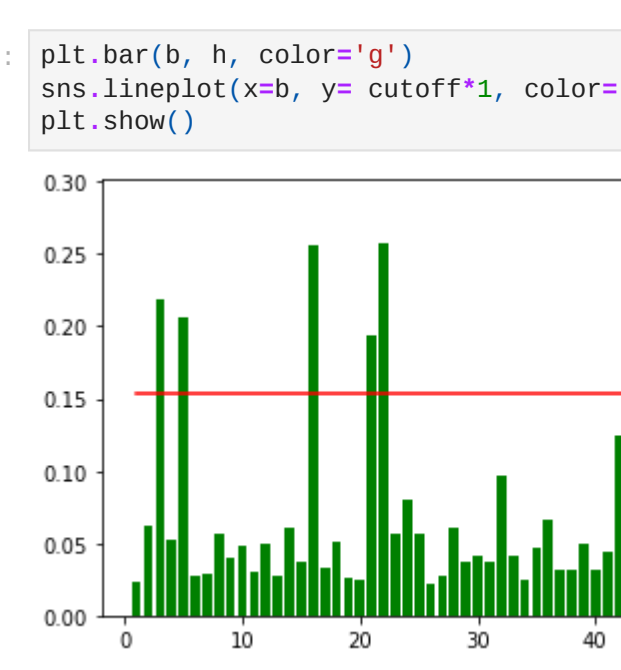
In [18]:
```
mean_hatval = 4/52
cutoff = 2 * mean_hatval
print(mean_hatval, cutoff)
```

```
0.07692307692307693 0.15384615384615385
```

In [19]:
```
for i in range(52):
    if aa[i] > cutoff:
        print(b[i])
```

```
3
5
16
21
22
43
44
48
```

In [21]:
```
plt.bar(b, h, color='g')
sns.lineplot(x=b, y= cutoff*1, color='red')
plt.show()
```



Conclusion: Based on the diagonal element of the hat matrix, the observations in position 3, 5, 16, 21, 22, 43, 44, 48 are outlying X observations.

c Management wishes to predict the total labor hours required to handle the next

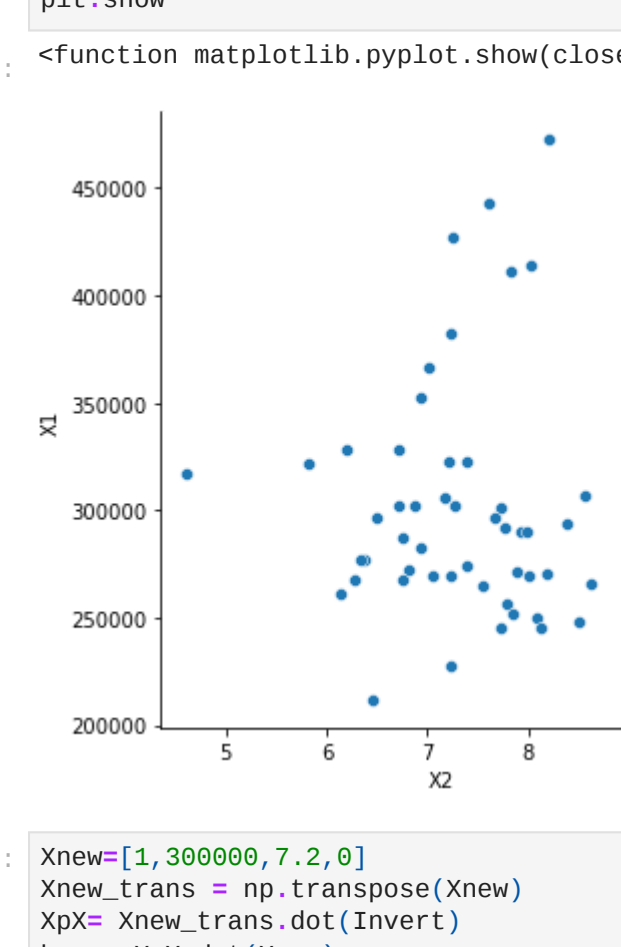
shipment containing X1 = 300,000 cases whose indirect costs of the total hours is X2 = 7.2 and X3 = 0 (no holiday in week). Construct a scatter plot of X2 against X1 and determine visually whether this prediction involves an extrapolation beyond the range of the data. Also, use (10.29) to determine whether an extrapolation is involved. Do your conclusions from the two methods agree?

In [22]:
```
b0 = bb[0]
b1 = bb[1]
b2 = bb[2]
b3 = bb[3]
Y_pred = b0 + (b1*300000) + (b2*7.2) + (b3*0)
print(Y_pred)
```

```
4291.215989085412
```

In [23]:
```
sns.relplot(x='X2', y='X1', data=df)
plt.show
```

Out[23]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [24]:
```
Xnew=[1,300000,7.2,0]
Xnew_trans = np.transpose(Xnew)
XpX= Xnew_trans.dot(Invert)
hnew= XpX.dot(Xnew)
print(hnew)
```

```
0.022217277721309638
```

Conclusion c: hnew, new is less than mean leverage value(0.022<0.077). So, hnew,new is well within the range of leverage values hii for the cases in the data set, no extrapolation is involved.

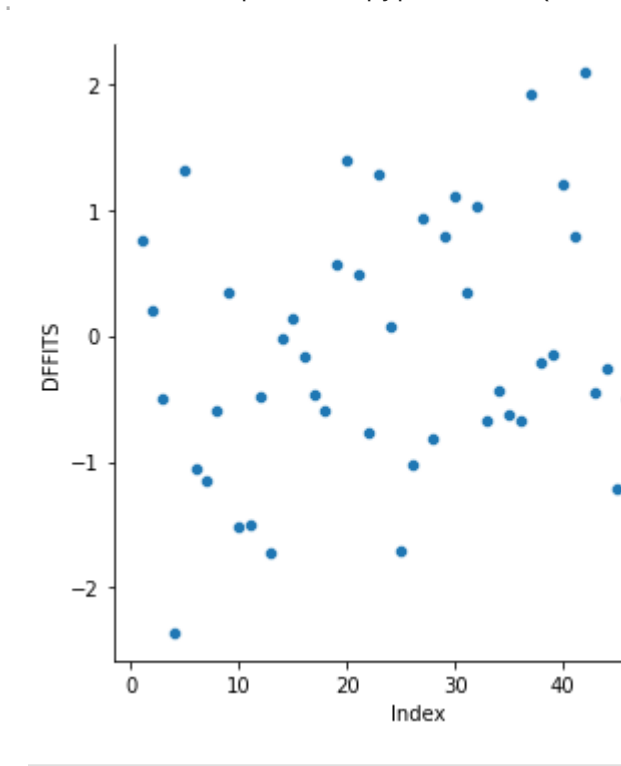
d. Cases 16, 22, 43, and 48 appear to be outlying X observations, and cases 10, 32, 38, and 40 appear to be outlying Y observations. Obtain the DFFITS, DFBETAS, and Cook's distance values for each of these cases to assess their influence. What do you conclude?

In [25]:
```
dffits = ti * ((h / (1 - h))**(1/2))
```

In [26]:
```
df2['DFFITS'] = pd.Series(np.random.randn(len(X1)), index=df2.index)
```

In [27]:
```
sns.relplot(x='Index', y='DFFITS', data=df2)
plt.show
```

Out[27]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [28]:
```
cutoff1 = 2 * np.sqrt(mean_hatval)
for i in range(52):
    if abs(dffits[i]) > cutoff1:
        print(b[i])
```

```
32
43
```

In [29]:
```
plt.bar(b, dffits, color='g')
sns.lineplot(x=b, y= cutoff1*1, color='red')
sns.lineplot(x=b, y= cutoff1*(-1), color='red')
plt.show()
```

Based on DFFITS, the observations in whose position are 32 and 43 are influential values.

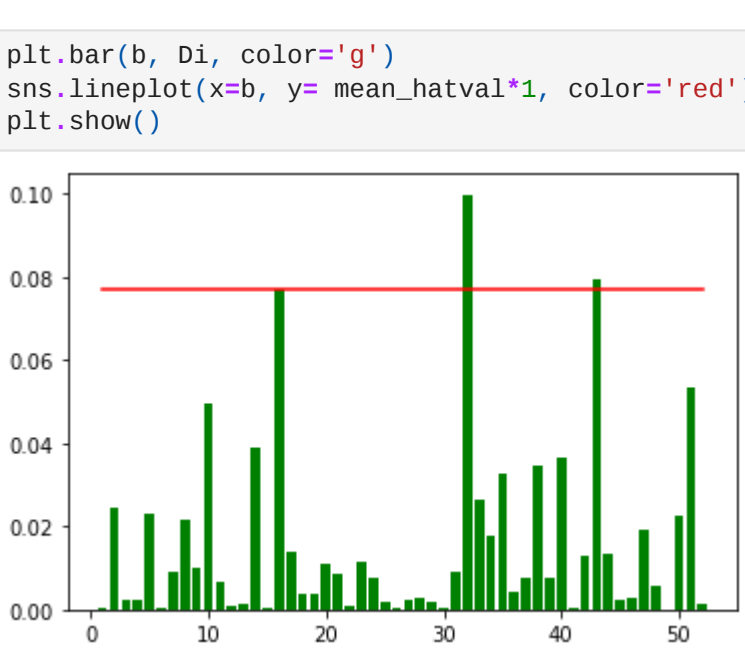
In [30]:
```
# Cook's distance
Di = ((resid**2)/(4*MSE))*(h/((1-h)**2))
Di.head(5)
```

Out[30]:
```
0    0.000296
1    0.024476
2    0.002081
3    0.002106
4    0.022996
dtype: float64
```

In [31]:
```
for i in range(52):
    if abs(Di[i]) > mean_hatval:
        print(b[i])
```

```
32
43
```

In [32]:
```
plt.bar(b, Di, color='g')
sns.lineplot(x=b, y= mean_hatval*1, color='red')
plt.show()
```



So, the observations in whose position are 32 and 43 are influential values according to Cook Distance

In [33]:
```
lm = smf.ols(formula = "Y ~ X1+X2+X3", data = df).fit()
lm.summary()
influence = lm.get_influence()
resid_student = influence.resid_studentized_external
(cooks, p) = influence.cooks_distance
(dffits, p) = influence.dffits
leverage = influence.hat_matrix_diag
```

In [34]:
```
resid_student = influence.resid_studentized_external
(cooks, p) = influence.cooks_distance
(dffits, p) = influence.dffits
leverage = influence.hat_matrix_diag
```

In [36]:
```
dfres = pd.concat([pd.Series(cooks, name = "cooks"), pd.Series(dffits, name = "dffits"), pd.Series(leverage, name = "leverage"), pd.Series(resid_student, name = "resid_student
dfres = pd.concat([df, dfres], axis = 1)
dfres.head()
```

Out[36]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
0	4264	305657	7.17	0	0.000296	-0.034063	0.022585	-0.224087
1	4496	328476	6.20	0	0.024476	0.314525	0.061800	1.225490
2	4317	317164	4.61	0	0.002081	-0.090301	0.218877	-0.170589
3	4292	366745	7.02	0	0.002106	-0.090974	0.052973	-0.384653
4	4945	265518	8.61	1	0.022996	0.301227	0.206328	0.590792

Double check

In [37]:
```
import math
dfres[abs(dfres.dffits) > 2 * math.sqrt(4 / 52)]
```

Out[37]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
31	3998	293225	9.01	0	0.099760	-0.651077	0.096023	-1.997667
42	5045	369989	9.65	1	0.079219	0.561652	0.286859	0.885566

In [38]:
```
import math
dfres[abs(dfres.cooks) > mean_hatval]
```

Out[38]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
31	3998	293225	9.01	0	0.099760	-0.651077	0.096023	-1.997667
42	5045	369989	9.65	1	0.079219	0.561652	0.286859	0.885566

In [39]:
```
# Adding DFBETAS
dfbeta = pd.concat([dfres, pd.DataFrame(influence.dfbetas, columns = ['dfb_intercept', 'dfb_X1', 'dfb_X2', 'dfb_X3'])], axis = 1)
print(dfbeta.head())
```

```
      Y      X1    X2  X3     cooks    dffits  leverage  resid_student  \
0  4264  305657  7.17   0  0.000296 -0.034063  0.022585      -0.224087
1  4496  328476  6.20   0  0.024476  0.314525  0.061800       1.225490
2  4317  317164  4.61   0  0.002081 -0.090301  0.218877      -0.170589
3  4292  366745  7.02   0  0.002106 -0.090974  0.052973      -0.384653
4  4945  265518  8.61   1  0.022996  0.301227  0.206328       0.590792

   dfb_intercept    dfb_X1    dfb_X2    dfb_X3
0      -0.007459 -0.002714  0.006206  0.010689
1       0.167778  0.104486 -0.238319 -0.040585
2      -0.068934 -0.014358  0.085368  0.020049
3       0.009435 -0.066932  0.025312  0.019826
4      -0.048128 -0.082695  0.169284  0.244168
```

In [40]:
```
dfb_intercept = dfbeta['dfb_intercept']
dfb_X1 = dfbeta['dfb_X1']
dfb_X2 = dfbeta['dfb_X2']
dfb_X3 = dfbeta['dfb_X3']
thresh = 2/np.sqrt(52)
print(thresh)
dfb_intercept[32]
```

```
0.2773500981126146
```

Out[40]:
```
-0.05242173543713732
```

In [41]:
```
import math
dfres[abs(dfb_intercept) > thresh]
```

Out[41]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
9	4560	277223	6.37	0	0.049350	0.458633	0.048269	2.036518
31	3998	293225	9.01	0	0.099760	-0.651077	0.096023	-1.997667
42	5045	369989	9.65	1	0.079219	0.561652	0.286859	0.885566

In [42]:
```
import math
dfres[abs(dfb_X1) > thresh]
```

Out[42]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
50	4186	411750	7.83	0	0.053137	-0.465284	0.102781	-1.374705

In [43]:
```
import math
dfres[abs(dfb_X2) > thresh]
```

Out[43]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
9	4560	277223	6.37	0	0.049350	0.458633	0.048269	2.036518
13	4063	306639	8.56	0	0.038751	-0.399742	0.060472	-1.575636
15	4833	321773	5.82	1	0.076895	-0.553990	0.255425	-0.945855
31	3998	293225	9.01	0	0.099760	-0.651077	0.096023	-1.997667
42	5045	369989	9.65	1	0.079219	0.561652	0.286859	0.885566

In [44]:
```
import math
dfres[abs(dfb_X3) > thresh]
```

Out[44]:

	Y	X1	X2	X3	cooks	dffits	leverage	resid_student
15	4833	321773	5.82	1	0.076895	-0.553990	0.255425	-0.945855
42	5045	369989	9.65	1	0.079219	0.561652	0.286859	0.885566

Based on DFBETAS values:

Based on intercept: 10, 32, 43 is influential values.

Based on X1: 51 is influential values.

Based on X2: 10, 14, 16, 32, 43 is influential values.

Based on X3: 16, 43 is influential values.

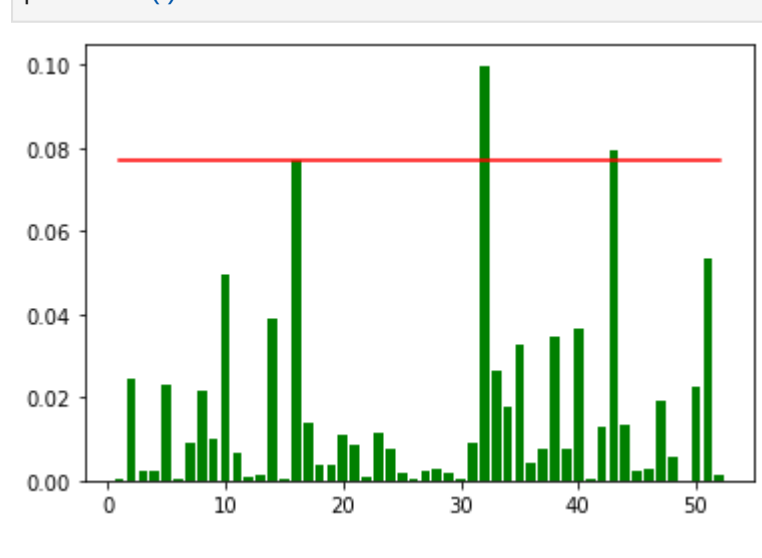
e. Calculate Cook's distance Di for each case and prepare an index plot. Are any cases

influential according to this measure?

In [45]:
```
Di.head()
```

Out[45]:
```
0    0.000296
1    0.024476
2    0.002081
3    0.002106
4    0.022996
dtype: float64
```

In [46]:
```
plt.bar(b, Di, color='g')
sns.lineplot(x=b, y= mean_hatval*1, color='red')
plt.show()
```



Conclusion e: 32 and 43 are influential cases according to this measure.

In [ ]: