

Exercise 14: This problem focuses on the collinearity problem.

Import notebook funcs

```
from notebookfuncs import *
```

Import user funcs

```
from userfuncs import *
```

Import libraries

```
from sympy import symbols, poly
import numpy as np
import seaborn as sns
import pandas as pd
import statsmodels.formula.api as smf
```

(a) Perform the following commands in Python:

```
rng = np.random.default_rng (10)
x1 = rng.uniform (0, 1, size =100)
x2 = 0.5 * x1 + rng.normal(size =100) / 10
y = 2 + 2 * x1 + 0.3 * x2 + rng.normal(size =100)
```

The last line corresponds to creating a linear model in which y is a function of x_1 and x_2 . Write out the form of the linear model. What are the regression coefficients?

```
x1, x2, y = symbols("x_1 x_2 y")
beta_0, beta_1, beta_2 = symbols(r"\beta_0 \beta_1 \beta_2")
equation = beta_0 + beta_1 * x1 + beta_2 * x2
display(equation)
equation = equation.subs([(beta_0,2), (beta_1,2),(beta_2, 0.3)])
equation = poly(equation)
```

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\text{Poly}(2.0x_1 + 0.3x_2 + 2.0, x_1, x_2, \text{domain} = \mathbb{R})$$

```
rng = np.random.default_rng(10)
x1 = rng.uniform(0, 1, size=100)
x2 = 0.5 * x1 + rng.normal(size=100) / 10
y = 2 + 2 * x1 + 0.3 * x2 + rng.normal(size=100);
```

(b) What is the correlation between x_1 and x_2 ? Create a scatterplot displaying the relationship between the variables.

Correlation between x1 and x2

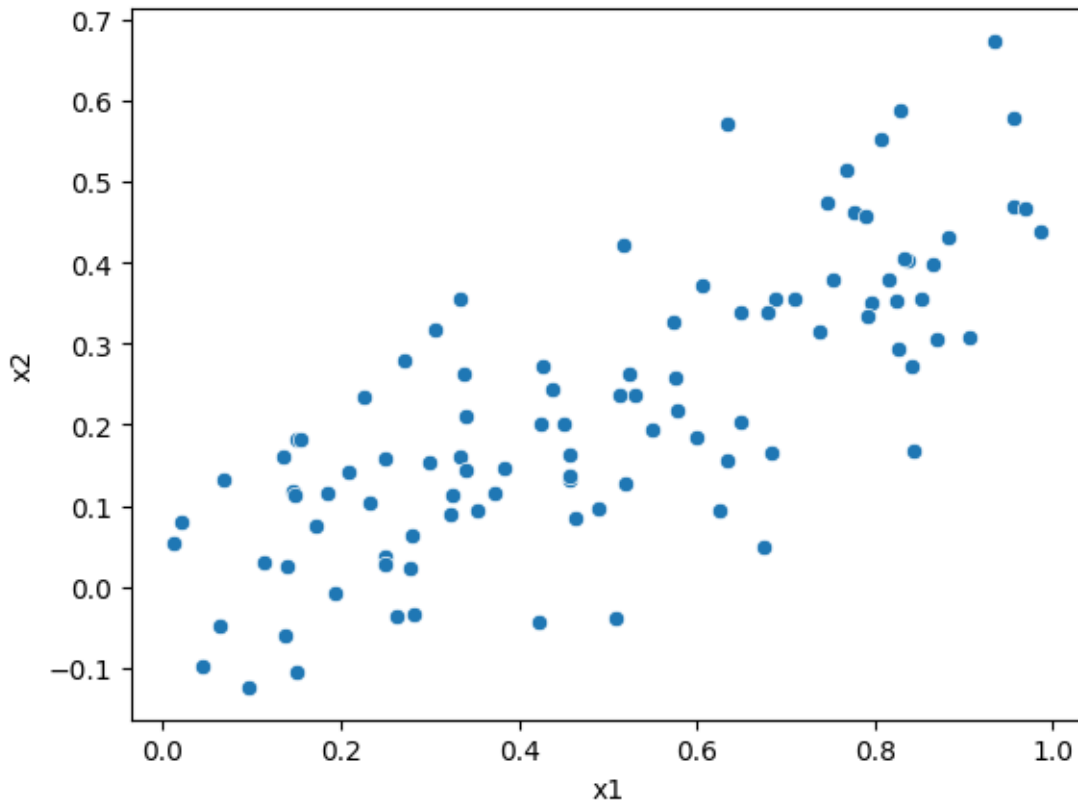
```
np.corrcoef(x1,x2)[0][1]
```

0.772324497691354

Display scatterplot of x1 against x2

```
def construct_df(x1, x2,y):
    df = pd.DataFrame({"x1": x1,"x2": x2, "y": y})
    return df

df = construct_df(x1,x2,y)
sns.scatterplot(df, x="x1", y="x2");
```



(c) Using this data, fit a least squares regression to predict y using x_1 and x_2 . Describe the results obtained. What are β_0 , β_1 , and β_2 ? How do these relate to the true β_0 , β_1 , and β_2 ? Can you reject the null hypothesis $H_0 : \beta_1 = 0$? How about the null hypothesis $H_0 : \beta_2 = 0$?

Fit a least squares regression

```
# Fit combined regression
def fit_combined(df):
    formula = "y ~ x1 + x2"
    model = smf.ols(f"{formula}", df)
    results = model.fit()
    print(results.summary())
    return results

results = fit_combined(df);
```

OLS Regression Results

=====						
Dep. Variable:	y	R-squared:	0.291			
Model:	OLS	Adj. R-squared:	0.276			
Method:	Least Squares	F-statistic:	19.89			
Date:	Wed, 16 Oct 2024	Prob (F-statistic):	5.76e-08			
Time:	18:03:33	Log-Likelihood:	-130.62			
No. Observations:	100	AIC:	267.2			
Df Residuals:	97	BIC:	275.1			
Df Model:	2					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

Intercept	1.9579	0.190	10.319	0.000	1.581	2.334
x1	1.6154	0.527	3.065	0.003	0.569	2.661
x2	0.9428	0.831	1.134	0.259	-0.707	2.592
=====						
Omnibus:	0.051	Durbin-Watson:	1.964			
Prob(Omnibus):	0.975	Jarque-Bera (JB):	0.041			
Skew:	-0.036	Prob(JB):	0.979			
Kurtosis:	2.931	Cond. No.	11.9			
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Describe the results

- The regression tests whether the coefficients β_0, β_1 and β_2 are 0. This is the null hypothesis.
- From the p-values, we deduce that the intercept and β_1 are significant and hence we do not accept the null hypothesis for them.
- β_2 , however, is not significant and thus its null hypothesis is accepted.
- The adjusted R^2 is 0.276 i.e., 27.6% of the variance of the response (y) is explained by the regressors x1 and x2.

What are β_0, β_1 and β_2 ?

```

params = results.params.to_frame().transpose()
params["Index"] = ["Estimate"]
params.set_index("Index")

```

	Intercept	x1	x2
Index			
Estimate	1.957909	1.615368	0.942777

How do these relate to the true β_0 , β_1 and β_2 ?

```

coeffs = equation.coeffs()
orig = pd.DataFrame({"Intercept": coeffs[2], "x1": coeffs[0], "x2":
    ↪ coeffs[1]}, index=[0])
orig["Index"] = ["Original"]
orig.set_index("Index")
res = pd.concat([params,orig], axis=0).set_index("Index")

```

	Intercept	x1	x2
Index			
Estimate	1.957909	1.615368	0.942777
Original	2.000000000000000	2.000000000000000	0.300000000000000

Influential points

```
get_influence_points(results)
```

```

n = 100.0, p = 3
Average Hat Leverage: 0.02999999999999995
Hat Leverage Cutoff = 2 * Average Hat Leverage = 0.05999999999999999
DFBetas Cutoff = 3 / sqrt(n) = 0.3
DFFITS Cutoff = 2 * sqrt(p/n) = 0.34641016151377546
Cooks Distance Cutoff = 1.0
Cooks Distance p-value Cutoff = 0.05
Studentized Residuals Cutoff = 3.0
Studentized Residuals p-value Cutoff = 0.01

```

```
(      dfb_Intercept      dfb_x1      dfb_x2      cooks_d      hat_diag      student_resid \
99          0.628518 -0.530616   0.253443   0.133708   0.04782         2.934959

      dffits      student_resid_pvalue      hat_influence      cooks_d_pvalue
99  0.657732          0.002087          0.140351          0.939756 ,
{'n': 100.0,
 'p': 3,
 'average_hat': 0.029999999999999995,
 'hat_leverage_cutoff': 0.05999999999999999,
 'dfbetas_cutoff': 0.3,
 'dffits_cutoff': 0.34641016151377546,
 'studentized_residuals_cutoff': 3.0,
 'studentized_residuals_pvalue_cutoff': 0.01,
 'cooks_d_cutoff': 1.0,
 'cooks_d_pvalue_cutoff': 0.05})
```

(d) Now fit a least squares regression to predict y using only x_1 . Comment on your results. Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

```
def fit_x1(df):
    formula = "y ~ x1"
    model = smf.ols(f"{formula}", df)
    results = model.fit()
    print(results.summary())
    return results

results = fit_x1(df);
```

```

                        OLS Regression Results
=====
Dep. Variable:          y      R-squared:                0.281
Model:                  OLS      Adj. R-squared:           0.274
Method:                 Least Squares      F-statistic:          38.39
Date:                  Wed, 16 Oct 2024      Prob (F-statistic):      1.37e-08
Time:                  18:03:33      Log-Likelihood:         -131.28
No. Observations:      100      AIC:                   266.6
Df Residuals:          98      BIC:                   271.8
Df Model:               1
Covariance Type:       nonrobust
=====
```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	1.9371	0.189	10.242	0.000	1.562	2.312
x1	2.0771	0.335	6.196	0.000	1.412	2.742
=====						
Omnibus:		0.204	Durbin-Watson:			1.931
Prob(Omnibus):		0.903	Jarque-Bera (JB):			0.042
Skew:		-0.046	Prob(JB):			0.979
Kurtosis:		3.038	Cond. No.			4.65
=====						

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

- Yes, we can reject the null hypothesis $H_0 : \beta_1 = 0$ since the p-value for the coefficient of x_1 is significant.

Influential points

```
get_influence_points(results)
```

```
n = 100.0, p = 2
Average Hat Leverage: 0.020000000000000004
Hat Leverage Cutoff = 2 * Average Hat Leverage = 0.040000000000000001
DFBetas Cutoff = 3 / sqrt(n) = 0.3
DFFITS Cutoff = 2 * sqrt(p/n) = 0.282842712474619
Cooks Distance Cutoff = 1.0
Cooks Distance p-value Cutoff = 0.05
Studentized Residuals Cutoff = 3.0
Studentized Residuals p-value Cutoff = 0.01

(   dfb_Intercept   dfb_x1   cooks_d   hat_diag   student_resid   dffits   \
99      0.623723 -0.541833  0.179606   0.04072      3.027795  0.623819

      student_resid_pvalue   hat_influence   cooks_d_pvalue
99      0.001578      0.123292      0.835874 ,
{'n': 100.0,
```

```
'p': 2,
'average_hat': 0.020000000000000004,
'hat_leverage_cutoff': 0.04000000000000001,
'dfbetas_cutoff': 0.3,
'dffits_cutoff': 0.282842712474619,
'studentized_residuals_cutoff': 3.0,
'studentized_residuals_pvalue_cutoff': 0.01,
'cooks_d_cutoff': 1.0,
'cooks_d_pvalue_cutoff': 0.05})
```

(e) Now fit a least squares regression to predict y using only x_2 . Comment on your results. Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

```
def fit_x2(df):
    formula = "y ~ x2"
    model = smf.ols(f"{formula}", df)
    results = model.fit()
    print(results.summary())
    return results

results = fit_x2(df);
```

```

                                OLS Regression Results
=====
Dep. Variable:                  y      R-squared:                0.222
Model:                            OLS      Adj. R-squared:           0.214
Method:                 Least Squares      F-statistic:                27.99
Date:                Wed, 16 Oct 2024      Prob (F-statistic):       7.43e-07
Time:                  18:03:33      Log-Likelihood:          -135.24
No. Observations:          100      AIC:                     274.5
Df Residuals:              98      BIC:                     279.7
Df Model:                   1
Covariance Type:            nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
Intercept	2.3239	0.154	15.124	0.000	2.019	2.629
x2	2.9103	0.550	5.291	0.000	1.819	4.002

```

=====
Omnibus:                  0.191      Durbin-Watson:           1.943

```


Prob(Omnibus):	0.909	Jarque-Bera (JB):	0.373
Skew:	-0.034	Prob(JB):	0.830
Kurtosis:	2.709	Cond. No.	6.11
=====			

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Can you reject the null hypothesis $H_0 : \beta_1 = 0$?

- Yes, we can reject the null hypothesis $H_0 : \beta_1 = 0$ since the p-value for the coefficient of x_2 is significant.

Influential points

```
get_influence_points(results)
```

```
n = 100.0, p = 2
Average Hat Leverage: 0.01999999999999993
Hat Leverage Cutoff = 2 * Average Hat Leverage = 0.03999999999999999
DFBetas Cutoff = 3 / sqrt(n) = 0.3
DFFITS Cutoff = 2 * sqrt(p/n) = 0.282842712474619
Cooks Distance Cutoff = 1.0
Cooks Distance p-value Cutoff = 0.05
Studentized Residuals Cutoff = 3.0
Studentized Residuals p-value Cutoff = 0.01
```

(Empty DataFrame

Columns: [dfb_Intercept, dfb_x2, cooks_d, hat_diag, student_resid, dffits, student_resid_pvalue]
Index: [],

```
{'n': 100.0,
 'p': 2,
 'average_hat': 0.01999999999999993,
 'hat_leverage_cutoff': 0.03999999999999999,
 'dfbetas_cutoff': 0.3,
 'dffits_cutoff': 0.282842712474619,
 'studentized_residuals_cutoff': 3.0,
 'studentized_residuals_pvalue_cutoff': 0.01,
 'cooks_d_cutoff': 1.0,
 'cooks_d_pvalue_cutoff': 0.05})
```

(f) Do the results obtained in (c)–(e) contradict each other? Explain your answer.

- No, the results do not contradict each other since the two variables are collinear and contain the same information.
- Thus, they can be interchanged for each other without much loss of information in the regression model.

(g) Suppose we obtain one additional observation, which was unfortunately mismeasured. We use the function `np.concatenate()` to add this additional observation to each of x_1, x_2 and y .

```
x1 = np.concatenate ([x1 , [0.1]])
x2 = np.concatenate ([x2 , [0.8]])
y = np.concatenate ([y, [6]])
```

Re-fit the linear models from (c) to (e) using this new data. What effect does this new observation have on the each of the models? In each model, is this observation an outlier? A high-leverage point? Both? Explain your answers.

Add an additional observation

```
x1 = np.concatenate ([x1 , [0.1]])
x2 = np.concatenate ([x2 , [0.8]])
y = np.concatenate ([y, [6]]);
```

```
x1[-1], x2[-1], y[-1]
```

```
(0.1, 0.8, 6.0)
```

```
df = construct_df(x1,x2,y)
df.tail(1)
```

	x1	x2	y
100	0.1	0.8	6.0

Combined regression

```
results = fit_combined(df);
```

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.292			
Model:	OLS	Adj. R-squared:	0.277			
Method:	Least Squares	F-statistic:	20.17			
Date:	Wed, 16 Oct 2024	Prob (F-statistic):	4.60e-08			
Time:	18:03:34	Log-Likelihood:	-135.30			
No. Observations:	101	AIC:	276.6			
Df Residuals:	98	BIC:	284.5			
Df Model:	2					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

Intercept	2.0618	0.192	10.720	0.000	1.680	2.443
x1	0.8575	0.466	1.838	0.069	-0.068	1.783
x2	2.2663	0.705	3.216	0.002	0.868	3.665
=====						
Omnibus:	0.139	Durbin-Watson:	1.894			
Prob(Omnibus):	0.933	Jarque-Bera (JB):	0.320			
Skew:	0.013	Prob(JB):	0.852			
Kurtosis:	2.725	Cond. No.	9.68			

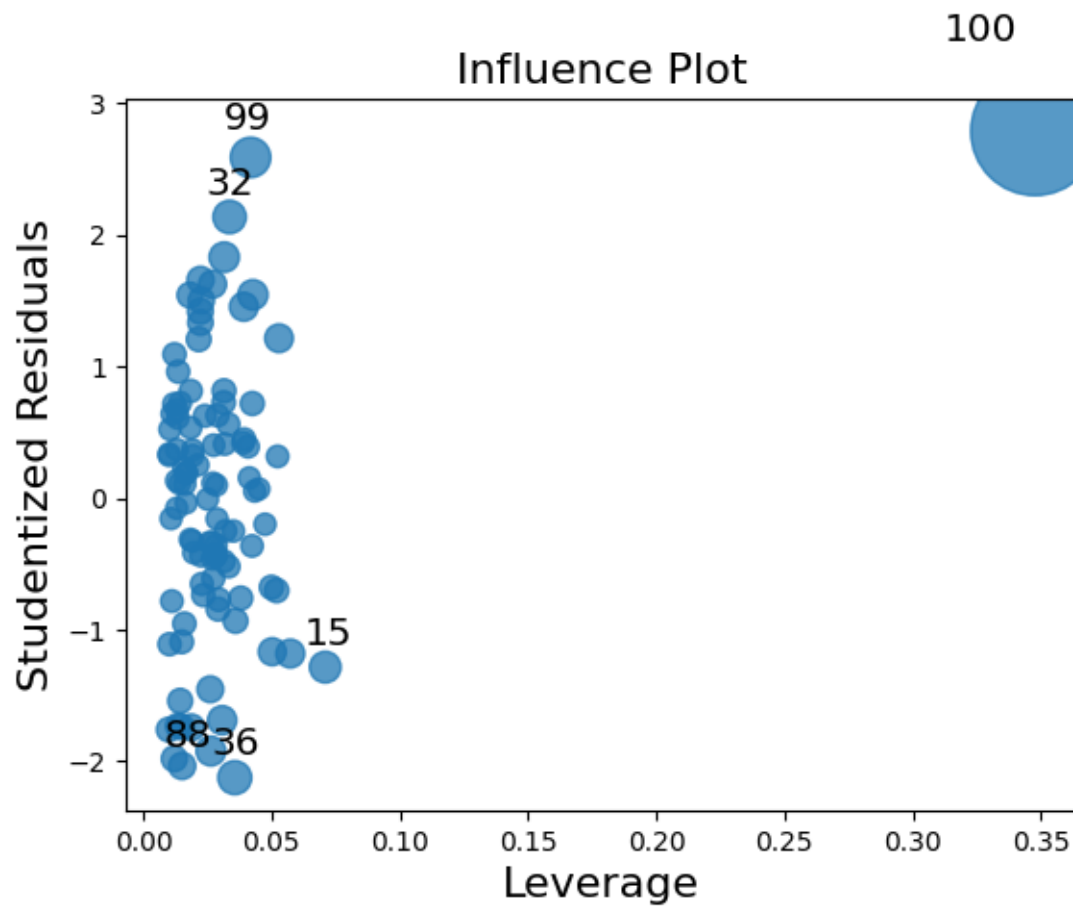
Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- Here, we see the effect of the additional mismeasured data point.
- The effect on the combined regression is to switch the significance of the regressors x1 and x2.
- Now, the coefficient of x1 is not statistically significant with a p-value of 0.07.

Residuals, outliers, leverage and influence

```
display_cooks_distance_plot(results);
```



```
display_hat_leverage_plot(results)
```

Unable to display output for mime type(s): text/html

Unable to display output for mime type(s): application/vnd.plotly.v1+json, text/html

```
get_influence_points(results)
```

```

n = 101.0, p = 3
Average Hat Leverage: 0.02970297029702972
Hat Leverage Cutoff = 2 * Average Hat Leverage = 0.05940594059405944
DFBetas Cutoff = 3 / sqrt(n) = 0.29851115706299675
DFFITS Cutoff = 2 * sqrt(p/n) = 0.3446909937728556
Cooks Distance Cutoff = 1.0
Cooks Distance p-value Cutoff = 0.05
Studentized Residuals Cutoff = 3.0
Studentized Residuals p-value Cutoff = 0.01

(      dfb_Intercept    dfb_x1    dfb_x2    cooks_d    hat_diag    student_resid \
99      0.522200 -0.421657    0.129666    0.092112    0.041905      2.585431
100     0.558412 -1.679554    1.941733    1.287988    0.347672      2.783731

      dffits    student_resid_pvalue    hat_influence    cooks_d_pvalue
99    0.540708          0.005607          0.108343          0.964231
100   2.032257          0.003230          0.967824          0.282850 ,
{'n': 101.0,
 'p': 3,
 'average_hat': 0.02970297029702972,
 'hat_leverage_cutoff': 0.05940594059405944,
 'dfbetas_cutoff': 0.29851115706299675,
 'dffits_cutoff': 0.3446909937728556,
 'studentized_residuals_cutoff': 3.0,
 'studentized_residuals_pvalue_cutoff': 0.01,
 'cooks_d_cutoff': 1.0,
 'cooks_d_pvalue_cutoff': 0.05})

```

- From the above, we can see that there are two influential datapoints, 99 and 100.
- This is initially surprising until we compute the influential points without the freshly added mis-measured data point and discover that point 99 was influential in the earlier regression.

Regress on x1

```
results = fit_x1(df);
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:          y    R-squared:          0.217

```

```

Model:                OLS      Adj. R-squared:      0.209
Method:               Least Squares  F-statistic:      27.42
Date:                 Wed, 16 Oct 2024  Prob (F-statistic): 9.23e-07
Time:                 18:03:35   Log-Likelihood:    -140.37
No. Observations:     101      AIC:                284.7
Df Residuals:         99      BIC:                290.0
Df Model:              1
Covariance Type:      nonrobust

```

```

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept      2.0739      0.201      10.310      0.000      1.675      2.473
x1              1.8760      0.358       5.236      0.000      1.165      2.587
=====

```

```

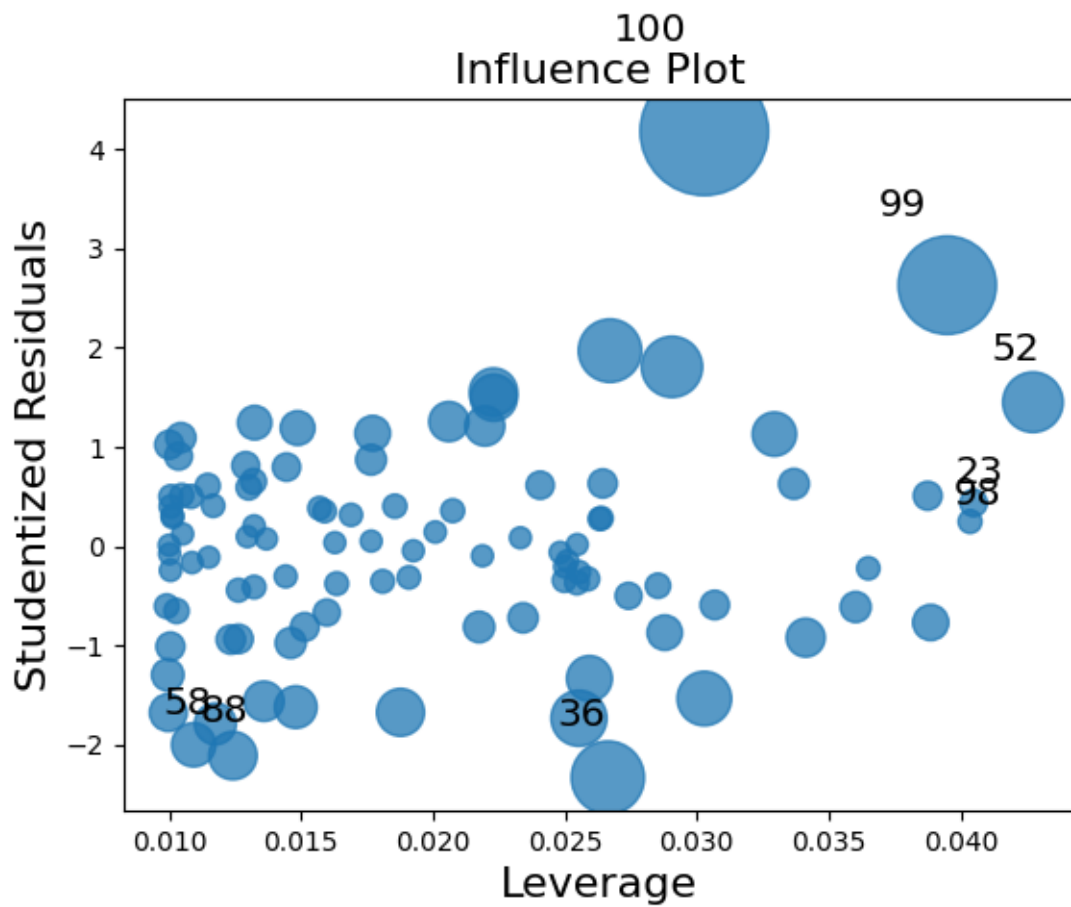
Omnibus:                8.232   Durbin-Watson:      1.636
Prob(Omnibus):          0.016   Jarque-Bera (JB):    10.781
Skew:                   0.396   Prob(JB):            0.00456
Kurtosis:               4.391   Cond. No.            4.61
=====

```

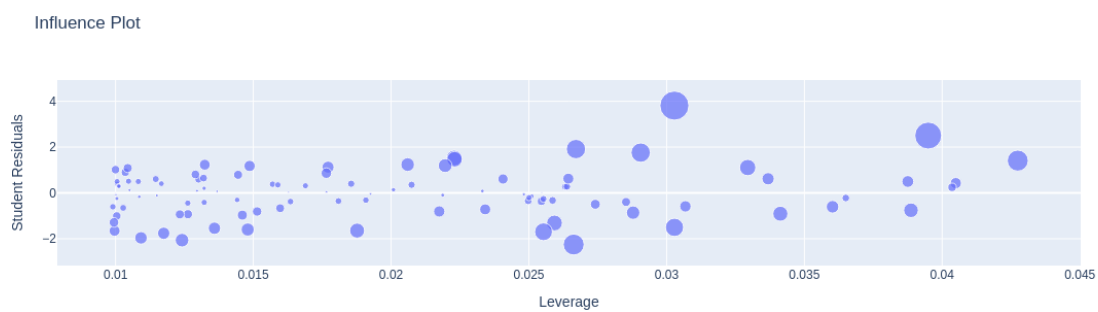
Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
display_cooks_distance_plot(results);
```



```
display_hat_leverage_plot(results)
```



```
get_influence_points(results)
```

```
n = 101.0, p = 2
Average Hat Leverage: 0.0198019801980198
Hat Leverage Cutoff = 2 * Average Hat Leverage = 0.0396039603960396
DFBetas Cutoff = 3 / sqrt(n) = 0.29851115706299675
DFFITS Cutoff = 2 * sqrt(p/n) = 0.2814390178921167
Cooks Distance Cutoff = 1.0
Cooks Distance p-value Cutoff = 0.05
Studentized Residuals Cutoff = 3.0
Studentized Residuals p-value Cutoff = 0.01
```

```
(      dfb_Intercept    dfb_x1    cooks_d    hat_diag    student_resid    dffits  \
99          0.532991 -0.461445    0.134079    0.039495          2.628842    0.533075
100         0.734700 -0.605898    0.233830    0.030283          4.179207    0.738540

      student_resid_pvalue    hat_influence    cooks_d_pvalue
99              0.004974          0.103827          0.874679
100             0.000032          0.126561          0.791933 ,
{'n': 101.0,
 'p': 2,
 'average_hat': 0.0198019801980198,
 'hat_leverage_cutoff': 0.0396039603960396,
 'dfbetas_cutoff': 0.29851115706299675,
 'dffits_cutoff': 0.2814390178921167,
 'studentized_residuals_cutoff': 3.0,
 'studentized_residuals_pvalue_cutoff': 0.01,
 'cooks_d_cutoff': 1.0,
 'cooks_d_pvalue_cutoff': 0.05})
```

- Similarly, in the regression of y on x_1 only, we find points 99 and 100 to be influential.

Regress on x_2

```
results = fit_x2(df);
```

OLS Regression Results

```
=====
Dep. Variable:          y    R-squared:          0.267
```



```

Model:                OLS      Adj. R-squared:      0.260
Method:               Least Squares    F-statistic:      36.10
Date:                 Wed, 16 Oct 2024    Prob (F-statistic): 3.13e-08
Time:                 18:03:54    Log-Likelihood:    -137.01
No. Observations:     101    AIC:                278.0
Df Residuals:          99    BIC:                283.3
Df Model:              1
Covariance Type:      nonrobust

```

```

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept      2.2840      0.151      15.088      0.000      1.984      2.584
x2              3.1458      0.524       6.008      0.000      2.107      4.185
=====

```

```

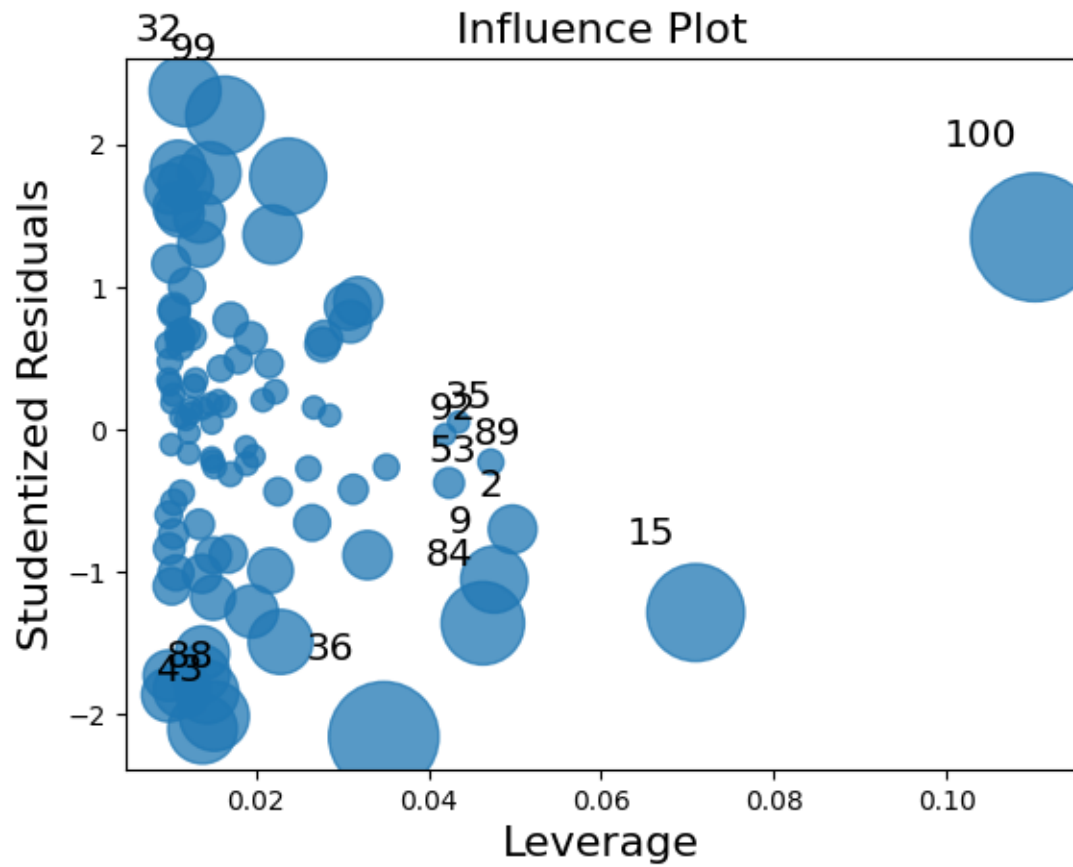
Omnibus:            0.495    Durbin-Watson:      1.939
Prob(Omnibus):      0.781    Jarque-Bera (JB):    0.631
Skew:               -0.041    Prob(JB):            0.729
Kurtosis:           2.621    Cond. No.            5.84
=====

```

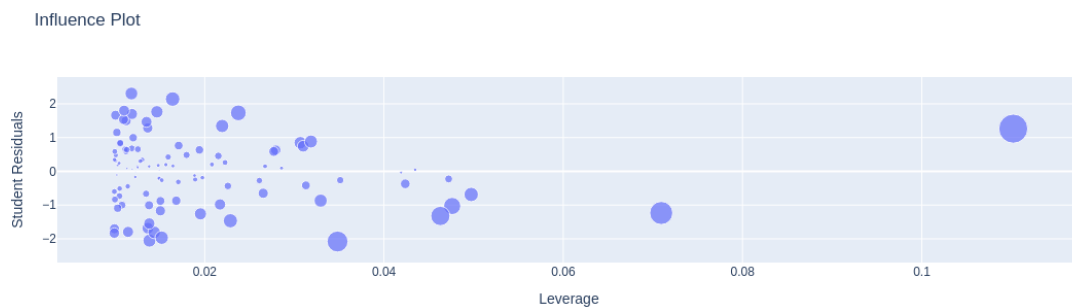
Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```
display_cooks_distance_plot(results);
```



```
display_hat_leverage_plot(results)
```



```
get_influence_points(results)
```

$n = 101.0$, $p = 2$

```

Average Hat Leverage: 0.019801980198019806
Hat Leverage Cutoff = 2 * Average Hat Leverage = 0.03960396039603961
DFBetas Cutoff = 3 / sqrt(n) = 0.29851115706299675
DFFITS Cutoff = 2 * sqrt(p/n) = 0.2814390178921167
Cooks Distance Cutoff = 1.0
Cooks Distance p-value Cutoff = 0.05
Studentized Residuals Cutoff = 3.0
Studentized Residuals p-value Cutoff = 0.01

```

```
(Empty DataFrame
```

```
Columns: [dfb_Intercept, dfb_x2, cooks_d, hat_diag, student_resid, dffits, student_resid_pvalue]
```

```
Index: [],
```

```
{'n': 101.0,
```

```
'p': 2,
```

```
'average_hat': 0.019801980198019806,
```

```
'hat_leverage_cutoff': 0.03960396039603961,
```

```
'dfbetas_cutoff': 0.29851115706299675,
```

```
'dffits_cutoff': 0.2814390178921167,
```

```
'studentized_residuals_cutoff': 3.0,
```

```
'studentized_residuals_pvalue_cutoff': 0.01,
```

```
'cooks_d_cutoff': 1.0,
```

```
'cooks_d_pvalue_cutoff': 0.05})
```

- In the regression of y on x2, no data point is influential since neither the studentized residuals or their associated p-values cross the thresholds for these parameters.

```
allDone();
```

```
<IPython.lib.display.Audio object>
```