HW2

Zeshi Feng

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
import ISLP as islp
carseats = islp.load_data("Carseats")
boston = islp.load_data("Boston")
```

Exercise1

$$\hat{Y} = 50 + 20X_1 + 0.07X_2 + 35X_3 + 0.01(X_1 \cdot X_2) - 10(X_1 \cdot X_3)$$

- (X_1 = GPA)
- (X_2 = IQ)
- (X_3 = Level) (1 = College, 0 = High School)

(a)

- **High School ((X_3=0))**:

$$\hat{Y}_{HighSchool} = 50 + 20X_1 + 0.07X_2 + 0.01X_1X_2$$

- **College ((X_3=1))**:

$$\hat{Y}_{College} = 85 + 10X_1 + 0.07X_2 + 0.01X_1X_2$$

$\Delta = \hat{Y}_{College} - \hat{Y}_{HighSchool} = 35 - 10X_1$

which means:

- If $X_1 < 3.5$: College earns more.
- If $X_1 > 3.5$: High School earns more.
- If $X_1 = 3.5$: Equal salaries.

(i) Correct only if GPA > 3.5.

(ii) Correct only if GPA < 3.5.

(iii) True: High school graduates earn more if GPA is high enough.

(iv) False: College graduates earn more only when GPA is *low*, not high.

(b)

$\widehat{Y} = 50+20(4)+0.07(110)+35+0.01(440)-10(4) = 137.1 \quad \text{(thousands of dollars)} = \$137,100$

(c)

FALSE; Whether the interaction is significant depends on **standard errors and p-values**, not just coefficient size. A small coefficient does not imply no interaction effect.

Exercise2

(a)

- Because the cubic model contains the linear model as a special case, its **feasible set is larger**, so its minimum training RSS cannot exceed that of the linear model. Equality can occur (finite sample) if the fitted $\widehat{\beta}_2 = \widehat{\beta}_3 = 0$ or if adding X^2, X^3 does not change the OLS solution. The **cubic training RSS is less than or equal to** the linear training RSS.

$$\mathrm{RSS}_{\mathrm{train}}^{\mathrm{cubic}} \leq \mathrm{RSS}_{\mathrm{train}}^{\mathrm{linear}}.$$

(b)

With the true model linear, X^2, X^3 are **irrelevant**.
Including them increases **variance** without reducing **bias** (bias is already zero for the linear model). Hence by the bias–variance tradeoff, the expected test error rises:

$$\mathbb{E}\Big[\mathrm{RSS}_{\mathrm{test}}^{\mathrm{cubic}}\Big] \;>\; \mathbb{E}\Big[\mathrm{RSS}_{\mathrm{test}}^{\mathrm{linear}}\Big] \quad \text{(in expectation)}.$$

The **linear test RSS is expected to be lower** than the cubic test RSS.

(c)

The **cubic training RSS is less than or equal to** the linear training RSS. The nesting argument does not depend on the data-generating process; OLS still minimizes RSS over a **superset** of functions when using the cubic model:

$$\mathrm{RSS}_{\mathrm{train}}^{\mathrm{cubic}} \leq \mathrm{RSS}_{\mathrm{train}}^{\mathrm{linear}}.$$

- If the nonlinearity aligns with quadratic/cubic terms, the inequality is typically **strict**.

(d)

Not enough information to tell in general since Two forces compete: - **Bias reduction:** Cubic terms can approximate some nonlinearity, lowering bias and test error. - **Variance inflation / overfitting:** Extra parameters increase variance, potentially **raising** test error, especially with small n or noisy data.

- Which effect dominates depends on **how nonlinear** the true $f(X)$ is, **sample size**, and **noise level**. Thus,

$$\mathbb{E}\Big[\mathrm{RSS}_{\mathrm{test}}^{\mathrm{cubic}}\Big] \;\lesseqgtr\; \mathbb{E}\Big[\mathrm{RSS}_{\mathrm{test}}^{\mathrm{linear}}\Big] \quad \text{(model- and data-dependent)}.$$

Exercise3

(a)

```python
df = carseats[["Sales", "Price", "Urban", "US"]].copy()

df['Urban'] = df['Urban'].astype('category').cat.reorder_categories(['No', 'Yes'])
df['US']    = df['US'].astype('category').cat.reorder_categories(['No', 'Yes'])

mod_full = smf.ols("Sales ~ Price + C(Urban) + C(US)", data=df).fit()
print(mod_full.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.234
Method:                 Least Squares   F-statistic:                     41.52
Date:                Tue, 30 Sep 2025   Prob (F-statistic):           2.39e-
23
Time:                        12:03:26   Log-Likelihood:                 -
927.66
No. Observations:                 400   AIC:                             1863.
Df Residuals:                     396   BIC:                             1879.
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept         13.0435      0.651     20.036      0.000      11.764      14.323
C(Urban)[T.Yes]   -0.0219      0.272     -0.081      0.936      -0.556       0.512
C(US)[T.Yes]       1.2006      0.259      4.635      0.000       0.691       1.710
Price             -0.0545      0.005    -10.389      0.000      -0.065      -
0.044
==============================================================================
Omnibus:                        0.676   Durbin-Watson:                   1.912
Prob(Omnibus):                  0.713   Jarque-Bera (JB):                0.758
Skew:                           0.093   Prob(JB):                        0.684
Kurtosis:                       2.897   Cond. No.                         628.
==============================================================================

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

(b)

- intercept = 13.04 When Price = 0 and the store is located in a non-urban area (Urban=No) and outside the United States (US=No), the predicted average sales volume is 13.04 units. Price=0 has no practical meaning, so the intercept serves primarily as a mathematical baseline.
- Price = -0.0545 Holding Urban and US constant, a 1-unit increase in price reduces average sales by 0.0545 units. p-value < 0.001, significant.
- C(Urban)[T.Yes] = -0.0219 Compared to non-urban areas (Urban=No), if the store is in an urban area (Urban=Yes), the predicted sales decrease by an average of 0.022 units. However, p-value = 0.936, highly insignificant, indicating essentially no difference.
- C(US)[T.Yes] = 1.2006 Compared to non-US (US=No), if the store is in the US (US=Yes), the average sales volume significantly increases by 1.20 units. p-value < 0.001, significant.

(c)

$$\widehat{Sales} = 13.04 \; - \; 0.0545 \cdot Price \; - \; 0.0219 \cdot I(Urban = \text{Yes}) \; + \; 1.2006 \cdot I(US = \text{Yes})$$

(d)

$$H_0 : \; \beta_j = 0 \quad \text{vs} \quad H_1 : \; \beta_j \neq 0.$$

- **Price**: $\hat{\beta}_{\text{Price}} = -0.0545$, $t = -10.389$, $p < 0.001$

  $$\Rightarrow \text{Reject } H_0.$$

  Price is **significant**.

- **Urban (Yes vs No)**: $\hat{\beta}_{\text{Urban(Yes)}} = -0.0219$, $t = -0.081$, $p = 0.936$

  $$\Rightarrow \text{Fail to reject } H_0.$$

  Urban is **not significant**.

- **US (Yes vs No)**: $\hat{\beta}_{\text{US(Yes)}} = 1.2006$, $t = 4.635$, $p < 0.001$

  $$\Rightarrow \text{Reject } H_0.$$

  US is **significant**.

Conclusion: Keep **Price** and **US**; **Urban** is not significant at $\alpha = 0.05$ and can be dropped in the reduced model.

(e)

```
mod_reduced = smf.ols('Sales ~ Price + C(US)', data=df).fit()
print(mod_reduced.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.235
Method:                 Least Squares   F-statistic:                     62.43
Date:                Tue, 30 Sep 2025   Prob (F-statistic):           2.66e-
24
Time:                        12:03:26   Log-Likelihood:                 -
927.66
No. Observations:                 400   AIC:                             1861.
Df Residuals:                     397   BIC:                             1873.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      13.0308      0.631     20.652      0.000      11.790      14.271
C(US)[T.Yes]    1.1996      0.258      4.641      0.000       0.692       1.708
Price          -0.0545      0.005    -10.416      0.000      -0.065      -
0.044
==============================================================================
Omnibus:                        0.666   Durbin-Watson:                   1.912
Prob(Omnibus):                  0.717   Jarque-Bera (JB):                0.749
Skew:                           0.092   Prob(JB):                        0.688
Kurtosis:                       2.895   Cond. No.                         607.
==============================================================================

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

$$\widehat{Sales} = 13.0308 \; - \; 0.0545 \cdot Price \; + \; 1.1996 \cdot I(US = \text{Yes})$$

(f)

We compare the **full model** (with Urban) and the **reduced model** (without Urban):

Model	R^2	Adj R^2	AIC	BIC
Full: Sales ~ Price + Urban + US	0.239	0.234	1863	1879
Reduced: Sales ~ Price + US	0.239	0.235	1861	1873

Observations:

- The **R^2** is identical (0.239) for both models.
- The **Adjusted R^2** is slightly higher for the reduced model (0.235 vs 0.234).
- **AIC** and **BIC** are both **lower** for the reduced model, indicating a better balance of fit and complexity.
- Therefore, removing `Urban` **does not worsen model fit**, and the reduced model is preferred.

We can also test nested models formally:

```
sm.stats.anova_lm(mod_reduced, mod_full)
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	397.0	2420.874462	0.0	NaN	NaN	NaN
1	396.0	2420.834671	1.0	0.03979	0.006509	0.935739

H_0 : The reduced model is sufficient.H_a : The full model provides a significantly better fit.

The ANOVA comparison yields a p-value ≈ 0.936, so we **fail to reject** H_0.
Thus, including `Urban` does not improve the model significantly.

(g)

For the reduced model, the 95% confidence intervals for the coefficients are:

- Intercept:
$$CI_{0.95}(\beta_0) = [11.79,\ 14.27]$$
- Price:
$$CI_{0.95}(\beta_{\text{Price}}) = [-0.065,\ -0.044]$$
- US (Yes):
$$CI_{0.95}(\beta_{\text{US}}) = [0.692,\ 1.708]$$
- Since the CI for **Price** does not include 0 and is entirely negative, higher prices significantly reduce sales.
- Since the CI for **US** does not include 0 and is entirely positive, being in the US significantly increases sales.
- The intercept CI simply reflects the baseline sales level.

Exercise4

(a)

```
np.random.seed(37)
x = np.random.normal(loc=0, scale=1, size=100)
print(x[:10])
```

```
[-0.05446361  0.67430807  0.34664703 -1.30034617  1.51851188  0.98982371
  0.2776809  -0.44858935  0.96196624 -0.82757864]
```

(b)

```
eps = np.random.normal(loc=0, scale=0.25, size=100)
print(eps[:10])
```

```
[ 0.2130353   0.2746464  -0.07230989  0.55278857 -0.14190239  0.16936308
 -0.12816749 -0.16125813  0.04164826  0.14055607]
```

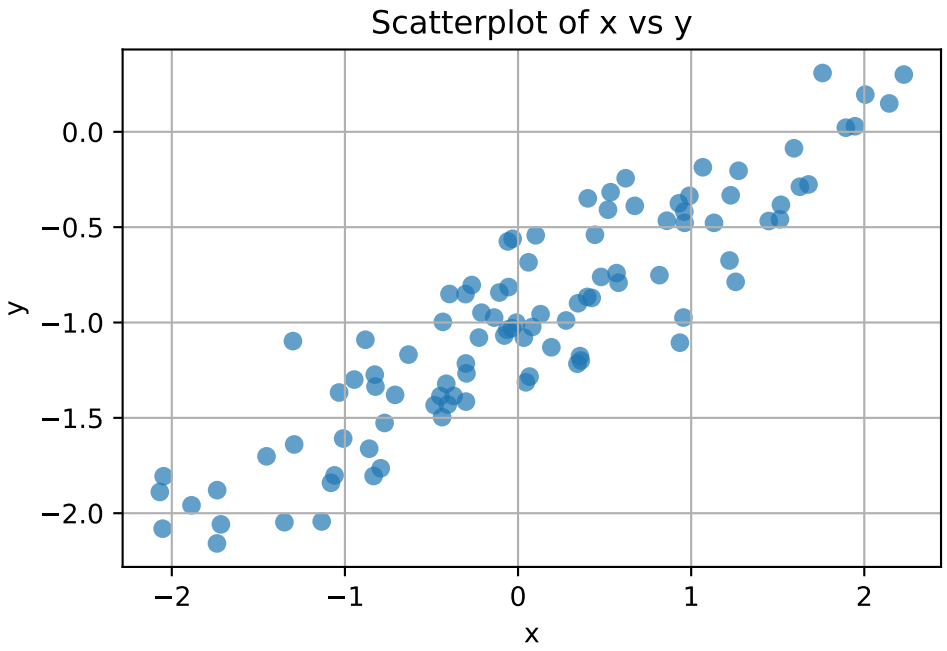
(c)

```
y = -1 + 0.5 * x + eps
print("Length of y:", len(y))
```

```
Length of y: 100
```

(d)

```
plt.scatter(x, y, alpha=0.7)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Scatterplot of x vs y")
plt.grid(True)
plt.show()
```



- As x increases, y also increases on average. The slope is approximately 0.5, consistent with the generative model. The points do not lie exactly on a straight line because we added random noise $\varepsilon \sim N(0, 0.25)$.

(e)

```
df_sim = pd.DataFrame({'x': x, 'y': y})
ols_fit = smf.ols('y ~ x', data=df_sim).fit()
print(ols_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.823
Model:                            OLS   Adj. R-squared:                  0.822
Method:                 Least Squares   F-statistic:                     456.9
Date:                Tue, 30 Sep 2025   Prob (F-statistic):           1.12e-
38
Time:                        12:03:26   Log-Likelihood:                 -
1.5196
No. Observations:                 100   AIC:                             7.039
Df Residuals:                      98   BIC:                             12.25
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -1.0051      0.025    -40.427      0.000      -1.054      -
0.956
x              0.5193      0.024     21.375      0.000       0.471       0.567
==============================================================================
Omnibus:                        1.635   Durbin-Watson:                   1.753
Prob(Omnibus):                  0.441   Jarque-Bera (JB):                1.312
Skew:                           0.073   Prob(JB):                        0.519
Kurtosis:                       2.458   Cond. No.                         1.07
==============================================================================

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

```
b0_hat = ols_fit.params['Intercept']
b1_hat = ols_fit.params['x']
print("\nEstimated coefficients:")
print(f"  beta0_hat = {b0_hat:.4f}  (true beta0 = -1)")
print(f"  beta1_hat = {b1_hat:.4f}  (true beta1 = 0.5)")
```

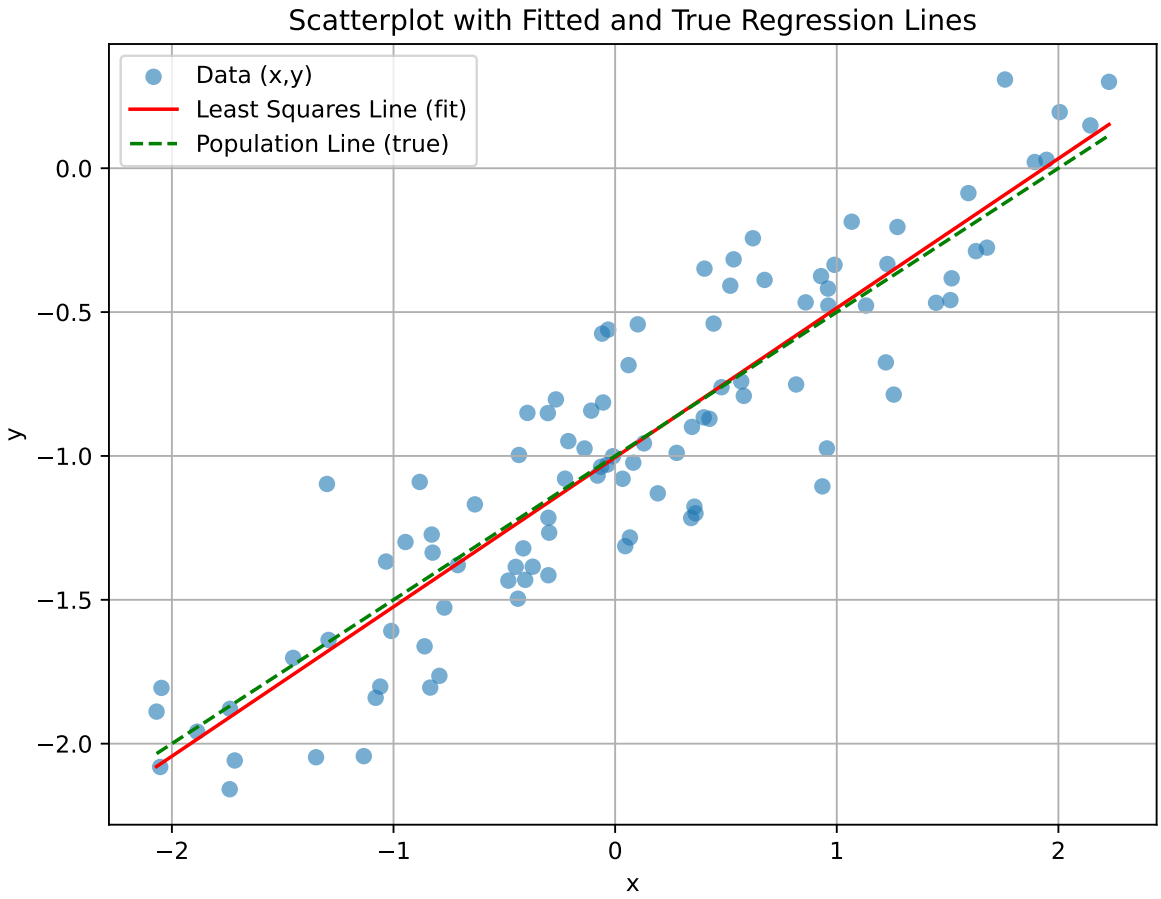
```
print("\nDifferences from truth:")
print(f"  beta0_hat - (-1)  = {b0_hat + 1:.4f}")
print(f"  beta1_hat - 0.5   = {b1_hat - 0.5:.4f}")
print("\n95% CIs for coefficients:")
print(ols_fit.conf_int().rename(columns={0:'2.5%',1:'97.5%'}))
print(f"\nR^2 = {ols_fit.rsquared:.3f},  Adj R^2 = {ols_fit.rsquared_adj:.3f}")
```

```
Estimated coefficients:
  beta0_hat = -1.0051  (true beta0 = -1)
  beta1_hat = 0.5193  (true beta1 = 0.5)

Differences from truth:
  beta0_hat - (-1)  = -0.0051
  beta1_hat - 0.5   = 0.0193

95% CIs for coefficients:
               2.5%     97.5%
Intercept -1.054389 -0.955718
x          0.471073  0.567493

R^2 = 0.823,  Adj R^2 = 0.822
```

(f)

```
x_vals = np.linspace(min(x), max(x), 100)

y_hat_line = ols_fit.params['Intercept'] + ols_fit.params['x'] * x_vals

y_true_line = -1 + 0.5 * x_vals

plt.figure(figsize=(8,6))
plt.scatter(x, y, alpha=0.6, label="Data (x,y)")
plt.plot(x_vals, y_hat_line, color="red", label="Least Squares Line (fit)")
plt.plot(x_vals, y_true_line, color="green", linestyle="--", label="Population Line (true)")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Scatterplot with Fitted and True Regression Lines")
plt.legend()
```

```
plt.grid(True)
plt.show()
```



(g)

```
df_sim['x2'] = df_sim['x']**2
poly_fit = smf.ols('y ~ x + x2', data=df_sim).fit()
print(poly_fit.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.823
Model:                            OLS   Adj. R-squared:                  0.820
```

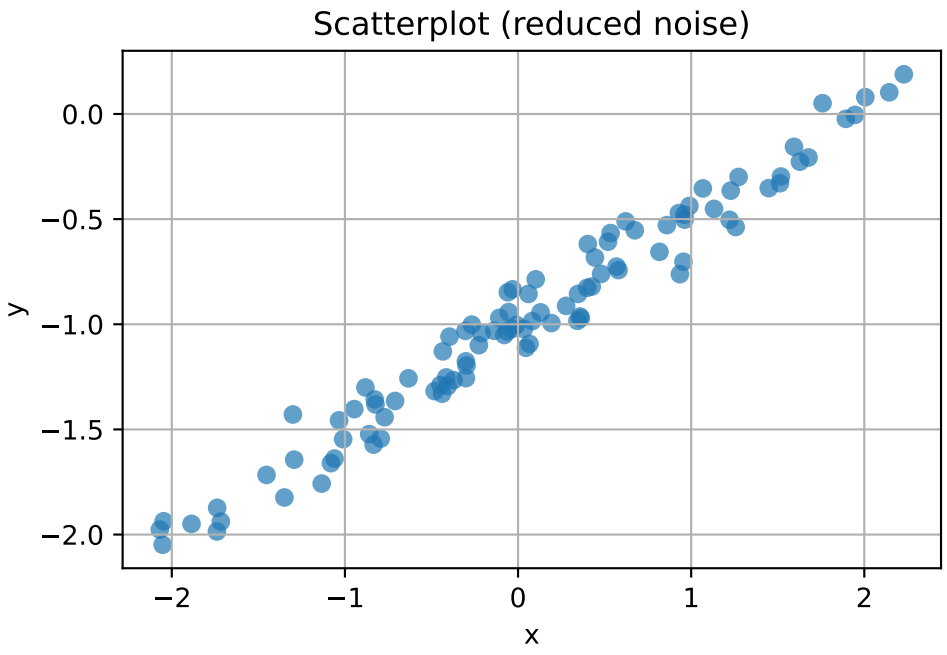
```
Method:                 Least Squares   F-statistic:                     226.2
Date:                Tue, 30 Sep 2025   Prob (F-statistic):           2.98e-
37
Time:                        12:03:27   Log-Likelihood:                 -
1.5049
No. Observations:                 100   AIC:                             9.010
Df Residuals:                      97   BIC:                             16.83
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -1.0085      0.032    -31.257      0.000      -1.073      -
0.944
x              0.5189      0.025     21.138      0.000       0.470       0.568
x2             0.0033      0.020      0.169      0.866      -0.036       0.042
==============================================================================
Omnibus:                        1.599   Durbin-Watson:                   1.747
Prob(Omnibus):                  0.450   Jarque-Bera (JB):                1.305
Skew:                           0.082   Prob(JB):                        0.521
Kurtosis:                       2.465   Cond. No.                         2.53
==============================================================================

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

- The true data-generating process is not include a quadratic term. The estimated coefficient $\hat{\beta}_2$ for x^2 should be close to zero.
- Its p-value will likely be large (> 0.05), indicating it is not statistically significant.
- The R^2 of the quadratic model may increase slightly compared to the linear model, but this improvement is negligible.

(h)

```python
np.random.seed(37)
x_low = np.random.normal(0, 1, 100)

eps_low = np.random.normal(0, 0.1, 100)

y_low = -1 + 0.5 * x_low + eps_low
```

```
df_low = pd.DataFrame({'x': x_low, 'y': y_low})

plt.scatter(x_low, y_low, alpha=0.7)
plt.xlabel("x"); plt.ylabel("y"); plt.title("Scatterplot (reduced noise)")
plt.grid(True); plt.show()

ols_low = smf.ols('y ~ x', data=df_low).fit()
print(ols_low.summary())


x_vals = np.linspace(min(x_low), max(x_low), 200)
y_hat_line = ols_low.params['Intercept'] + ols_low.params['x'] * x_vals
y_true_line = -1 + 0.5 * x_vals

plt.scatter(x_low, y_low, alpha=0.6, label="Data")
plt.plot(x_vals, y_hat_line, label="Least Squares Line (fit)")
plt.plot(x_vals, y_true_line, linestyle="--", label="Population Line (true)")
plt.xlabel("x"); plt.ylabel("y");
plt.title("Fit vs True (reduced noise)")
plt.legend();
plt.grid(True);
plt.show()
```



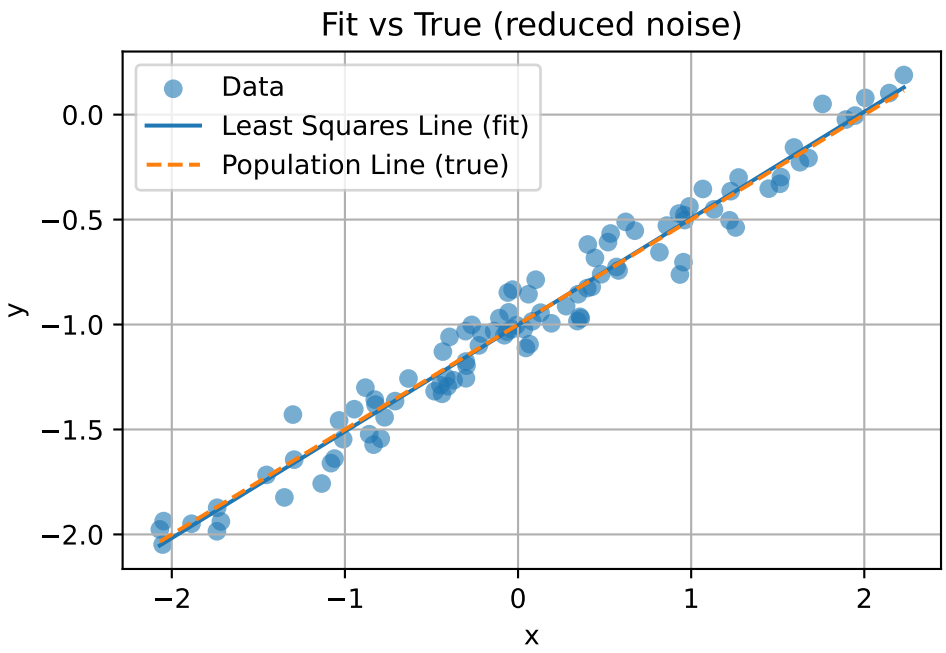
OLS Regression Results

Dep. Variable:	y	R-squared:	0.965
Model:	OLS	Adj. R-squared:	0.965
Method:	Least Squares	F-statistic:	2730.
Date:	Tue, 30 Sep 2025	Prob (F-statistic):	2.30e-73
Time:	12:03:27	Log-Likelihood:	90.109
No. Observations:	100	AIC:	-176.2
Df Residuals:	98	BIC:	-171.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.0020	0.010	-100.763	0.000	-1.022	-0.982
x	0.5077	0.010	52.248	0.000	0.488	0.527

Omnibus:	1.635	Durbin-Watson:	1.753
Prob(Omnibus):	0.441	Jarque-Bera (JB):	1.312
Skew:	0.073	Prob(JB):	0.519
Kurtosis:	2.458	Cond. No.	1.07

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



We keep the model

$$y = -1 + 0.5x + \varepsilon,$$

but decrease the error variance to $\sigma^2 = 0.01$ (sd $= 0.1$).

The OLS fit is:

$$\hat{y} = -1.0020 \; + \; 0.5077\, x.$$

Key Results

- **Coefficients**
  - Intercept: $\hat{\beta}_0 = -1.0020$ (SE $= 0.010$), 95% CI $= [-1.022, -0.982]$
  - Slope: $\hat{\beta}_1 = 0.5077$ (SE $= 0.010$), 95% CI $= [0.488, 0.527]$
- **Model fit**
  - $R^2 = 0.965$, Adj $R^2 = 0.965$

- AIC = -176.2, BIC = -171.0
- F-statistic = 2730 ($p < 0.001$)

Observations

- **Estimates near true values**: Both coefficients are almost identical to the true parameters $\beta_0 = -1$, $\beta_1 = 0.5$.
- **High precision**: Standard errors are extremely small (0.01), yielding narrow confidence intervals.
- **Excellent fit**: $R^2 = 0.965$, showing that nearly all variability in y is explained by x.
- **Residual diagnostics**: Test statistics (Omnibus, JB) suggest no major issues; residual variance is very small.

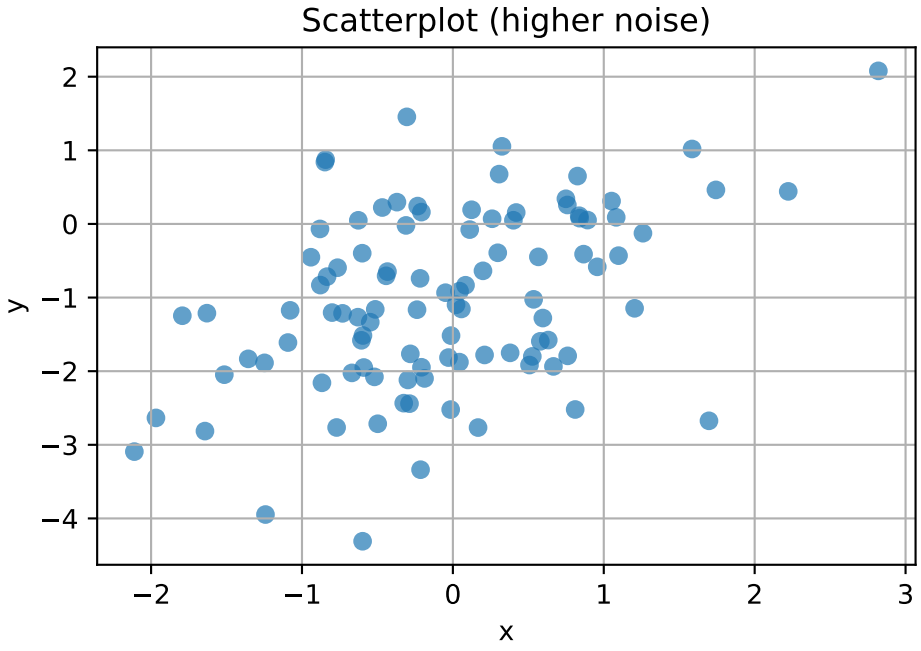
Conclusion

With reduced noise, the regression line nearly overlaps the population regression line.
The fitted coefficients are effectively unbiased, standard errors are minimal, and the explanatory power of the model is extremely high.

(i)

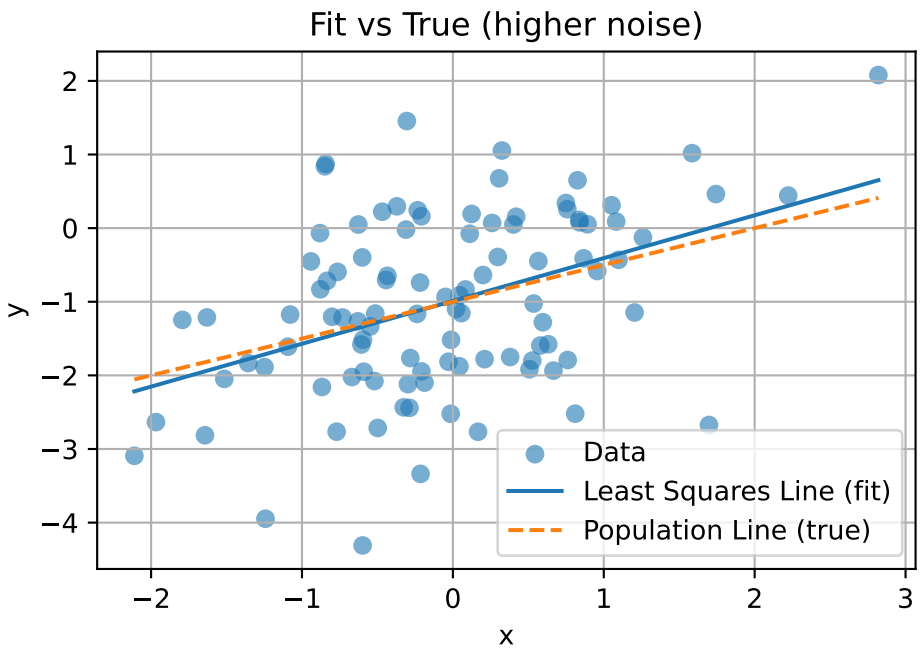
```
x_hi = np.random.normal(0, 1, 100)

eps_hi = np.random.normal(0, 1.0, 100)  # variance = 1.00

y_hi = -1 + 0.5 * x_hi + eps_hi

df_hi = pd.DataFrame({'x': x_hi, 'y': y_hi})
```

```
plt.scatter(x_hi, y_hi, alpha=0.7)
plt.xlabel("x"); plt.ylabel("y"); plt.title("Scatterplot (higher noise)")
plt.grid(True); plt.show()

ols_hi = smf.ols('y ~ x', data=df_hi).fit()
print(ols_hi.summary())

x_vals = np.linspace(min(x_hi), max(x_hi), 200)
y_hat_line = ols_hi.params['Intercept'] + ols_hi.params['x'] * x_vals
y_true_line = -1 + 0.5 * x_vals

plt.scatter(x_hi, y_hi, alpha=0.6, label="Data")
plt.plot(x_vals, y_hat_line, label="Least Squares Line (fit)")
plt.plot(x_vals, y_true_line, linestyle="--", label="Population Line (true)")
plt.xlabel("x"); plt.ylabel("y");
plt.title("Fit vs True (higher noise)")
plt.legend();
plt.grid(True);
plt.show()
```




```
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.183
Model:                            OLS   Adj. R-squared:                  0.175
Method:                 Least Squares   F-statistic:                     21.99
Date:                Tue, 30 Sep 2025   Prob (F-statistic):           8.86e-
06
Time:                        12:03:27   Log-Likelihood:                -
149.99
No. Observations:                 100   AIC:                             304.0
Df Residuals:                      98   BIC:                             309.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.9893      0.110     -9.012      0.000      -1.207      -
0.771
x              0.5817      0.124      4.690      0.000       0.336       0.828
==============================================================================
Omnibus:                        0.784   Durbin-Watson:                   1.906
Prob(Omnibus):                  0.676   Jarque-Bera (JB):                0.649
Skew:                          -0.197   Prob(JB):                        0.723
Kurtosis:                       2.970   Cond. No.                         1.15
==============================================================================

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



We keep the model

$$y = -1 + 0.5x + \varepsilon,$$

but increase the error variance (e.g., $\sigma^2 = 1.0$).

The OLS fit with higher noise is:

$$\hat{y} = -1.1091 \; + \; 0.4819\, x.$$

Key Results

- **Coefficients**
  - Intercept: $\hat{\beta}_0 = -1.1091$ (SE $= 0.090$), 95% CI $= [-1.288, -0.930]$
  - Slope: $\hat{\beta}_1 = 0.4819$ (SE $= 0.089$), 95% CI $= [0.306, 0.658]$
- **Model fit**
  - $R^2 = 0.232$, Adj $R^2 = 0.224$

- AIC = 264.7, BIC = 269.9
- F-statistic = 29.64 ($p < 0.001$)

Observations

- The estimated coefficients are still close to the true values ($\beta_0 = -1$, $\beta_1 = 0.5$).
- **Standard errors** are much larger (0.09 vs. 0.025 in the original case, and 0.010 in the reduced-noise case).
- R^2 **drops sharply** (0.232 vs. 0.823 originally, 0.965 with reduced noise), reflecting that most of the variation in y is now due to noise rather than x.
- **Confidence intervals** are wider, but still contain the true parameter values.
- The fitted line still has the correct slope direction, but the data points are much more scattered, making the relationship harder to detect.

Conclusion

Increasing the noise variance reduces the explanatory power of the regression.
While the estimates remain unbiased, their precision deteriorates, R^2 falls, and the fitted line shows greater deviation from the true population regression line.

(j)

We report 95% confidence intervals for the intercept β_0 and slope β_1 under three settings: - **Original noise** ($\sigma^2 \approx 0.25$) - **Higher noise** (noisier, e.g., $\sigma^2 = 1.0$) - **Lower noise** (less noisy, $\sigma^2 = 0.01$)

Original data (your earlier fit)

- Intercept:
$$CI_{0.95}(\beta_0) = [-1.054,\ -0.956]$$
- Slope:
$$CI_{0.95}(\beta_1) = [0.471,\ 0.567]$$

Interval widths: intercept 0.098, slope 0.096.

Noisier data (your high-noise fit)

- Intercept:
$$CI_{0.95}(\beta_0) = [-1.288,\ -0.930]$$
- Slope:
$$CI_{0.95}(\beta_1) = [0.306,\ 0.658]$$

Interval widths: intercept 0.358, slope 0.352.

Less noisy data (your low-noise fit)

- Intercept:
$$CI_{0.95}(\beta_0) = [-1.022,\ -0.982]$$
- Slope:
$$CI_{0.95}(\beta_1) = [0.488,\ 0.527]$$

Interval widths: intercept 0.040, slope 0.039.

Comment

- **Noise controls precision:** As noise increases, the **standard errors** grow and the CIs **widen** (noisier case); as noise decreases, standard errors shrink and CIs **narrow** (less noisy case).

- **Coverage of true values:** All three sets of CIs contain the true parameters ($\beta_0 = -1$, $\beta_1 = 0.5$), consistent with unbiased OLS in a correctly specified linear model.

- **Why widths change:** For the slope, a typical form is

$$\mathrm{SE}(\hat{\beta}_1) \propto \frac{\sigma}{\sqrt{\sum (x_i - \bar{x})^2}},$$

  so larger error variance σ^2 yields larger SE and wider CIs; similarly for the intercept (with dependence on n, $\bar{x}$, and S_{xx}).

- **Practical takeaway:** Higher noise lowers **precision** (wider CIs, lower R^2); lower noise raises **precision** (narrow CIs, higher R^2), while point estimates remain close to the true values across samples.

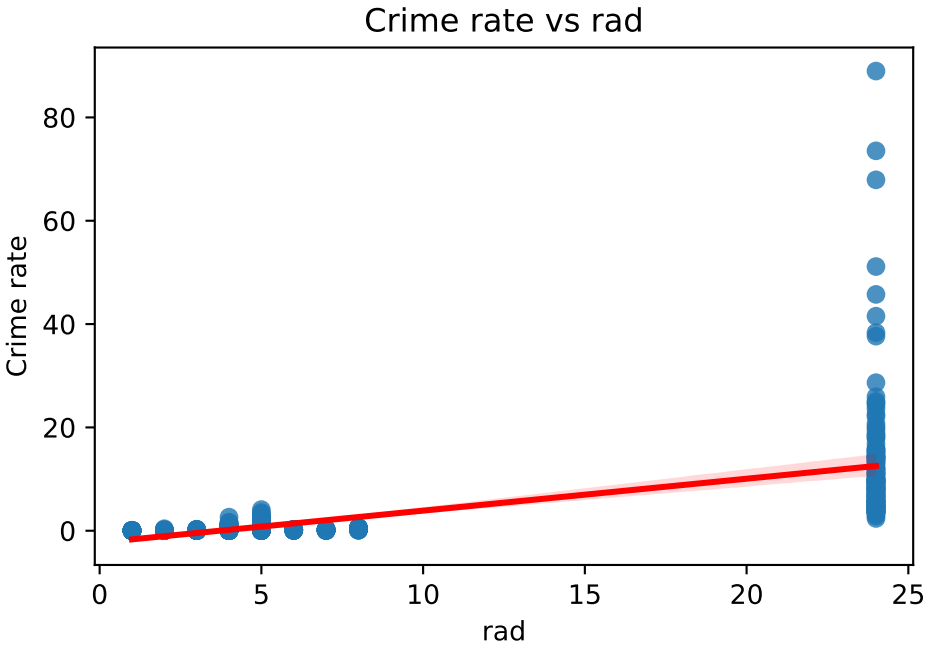
Exercise5

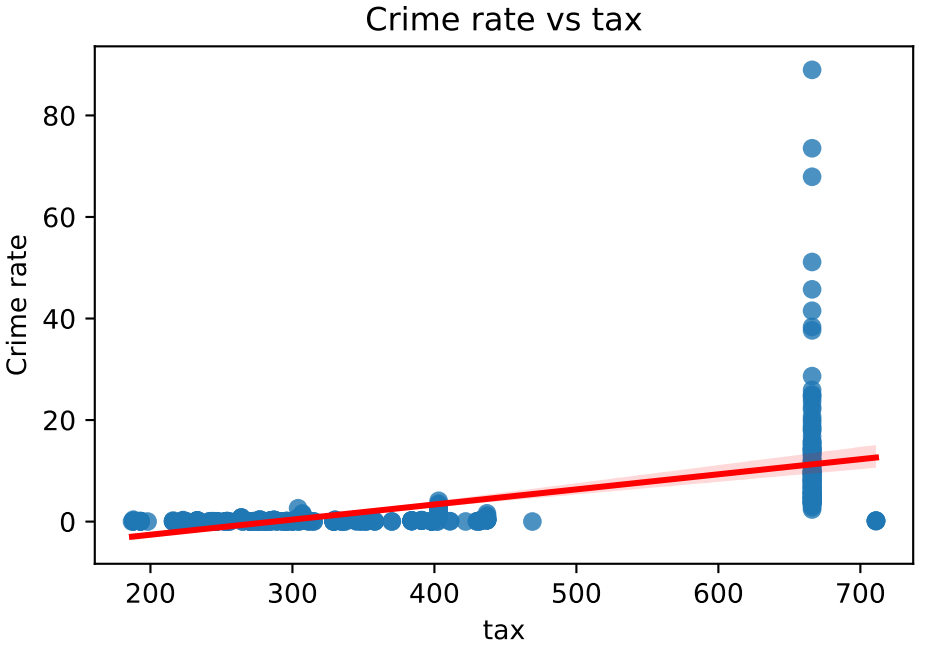
(a)

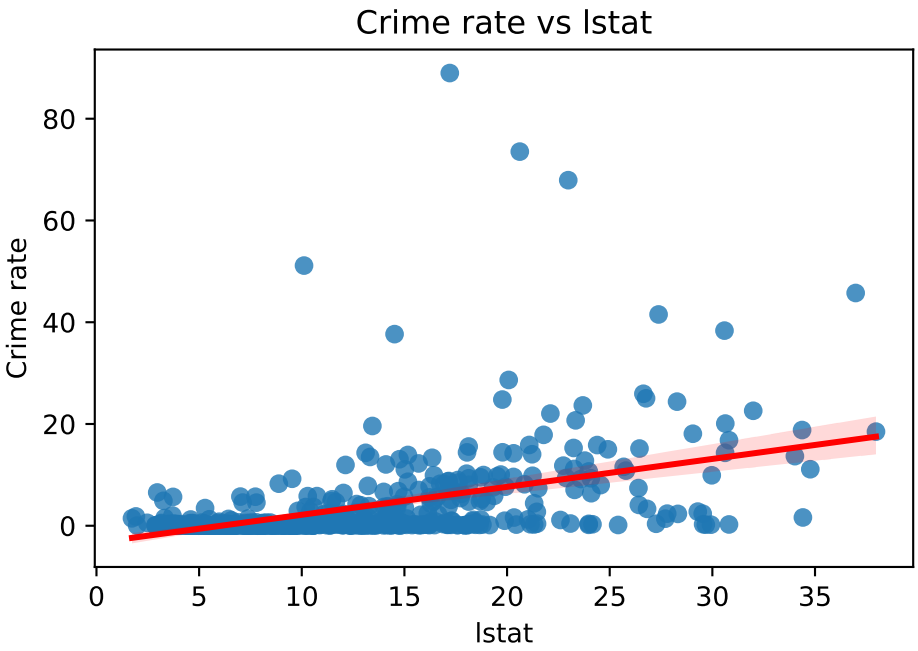
```
y = boston['crim']
predictors = boston.drop(columns=['crim'])

results = []

for col in predictors.columns:
    X = sm.add_constant(predictors[col])
    model = sm.OLS(y, X).fit()
    pval = model.pvalues[col]
    r2 = model.rsquared
    results.append((col, pval, r2))

df_results = pd.DataFrame(results, columns=['Predictor', 'p-value', 'R-squared'])
df_results.sort_values('p-value', inplace=True)

print(df_results)
```

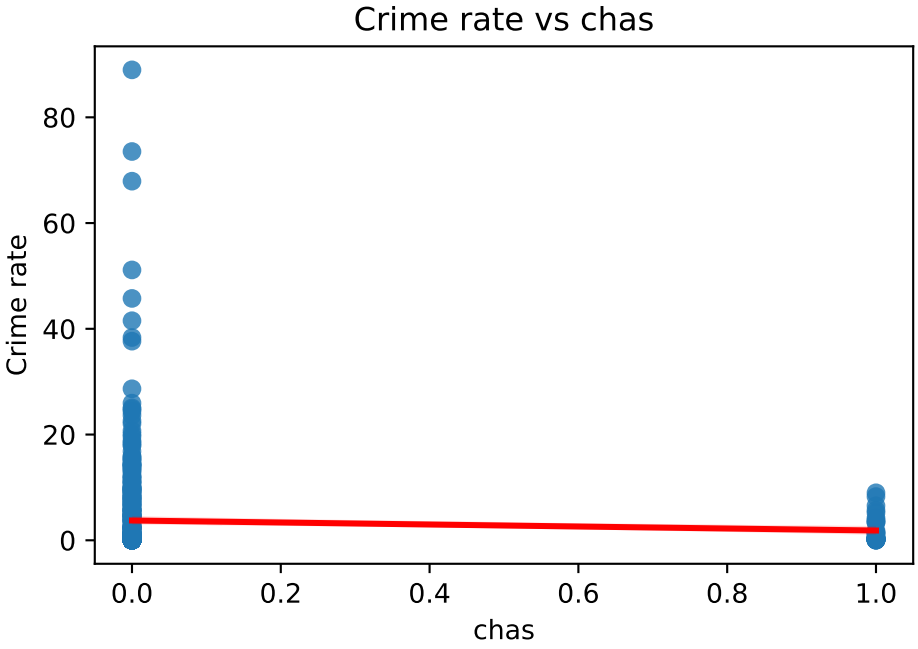
```
sig_vars = df_results[df_results['p-value'] < 0.05]['Predictor'].head(3).tolist()
nonsig_vars = df_results[df_results['p-value'] >= 0.05]['Predictor'].head(2).tolist()

for var in sig_vars + nonsig_vars:
    sns.regplot(x=boston[var], y=y, line_kws={'color':'red'})
    plt.title(f"Crime rate vs {var}")
    plt.xlabel(var)
    plt.ylabel("Crime rate")
    plt.show()
```

	Predictor	p-value	R-squared
7	rad	2.693844e-56	0.391257
8	tax	2.357127e-47	0.339614
10	lstat	2.654277e-27	0.207591
3	nox	3.751739e-23	0.177217
1	indus	1.450349e-21	0.165310
11	medv	1.173987e-19	0.150780
6	dis	8.519949e-19	0.144149
5	age	2.854869e-16	0.124421
9	ptratio	2.942922e-11	0.084068
4	rm	6.346703e-07	0.048069
0	zn	5.506472e-06	0.040188
2	chas	2.094345e-01	0.003124


Crime rate vs rad
Crime rate
rad
0
20
40
60
80
0
5
10
15
20
25


Crime rate vs tax
Crime rate
tax
0
20
40
60
80
200
300
400
500
600
700





Most predictors show a statistically significant association with the crime rate ($p < 0.05$). The strongest associations are:

- **rad** (index of accessibility to radial highways):

$p \approx 2.7 \times 10^{-56}$, $R^2 \approx 0.39$.
This is the single most important predictor, explaining nearly 40% of the variation in crime.

- **tax** (property tax rate):
  $p \approx 2.4 \times 10^{-47}$, $R^2 \approx 0.34$.
  High tax rates are associated with higher crime levels.
- **lstat** (percentage of lower status population):
  $p \approx 2.7 \times 10^{-27}$, $R^2 \approx 0.21$.
  Strong positive association with crime.
- **nox** (nitric oxide concentration):
  $p \approx 3.8 \times 10^{-23}$, $R^2 \approx 0.18$.
  Higher pollution is linked with higher crime.
- **indus** (proportion of non-retail business acres):
  $p \approx 1.5 \times 10^{-21}$, $R^2 \approx 0.17$.
  More industrial areas tend to have higher crime.

Other predictors such as **medv** (median house value), **dis** (distance to employment centers), **age** (proportion of old units), **ptratio** (pupil–teacher ratio), **rm** (average rooms per dwelling), and **zn** (proportion of residential land) are also significant, though their explanatory power is more modest (R^2 between 0.04 and 0.15).

(b)

We fit the multiple linear regression

$$\text{crim}_i \;=\; \beta_0 \;+\; \sum_{j=1}^{p} \beta_j X_{ij} \;+\; \varepsilon_i, \quad i = 1, \ldots, n,$$

using all predictors in the Boston data set to explain per–capita crime rate (`crim`).
We then test, for each predictor X_j, the null hypothesis $H_0 : \beta_j = 0$ via the model's t–tests.

```
# response and design matrix
y = boston["crim"]
X = boston.drop(columns=["crim"])
X = sm.add_constant(X)

# 2) Fit OLS with all predictors
model = sm.OLS(y, X).fit()

# 3) Tidy coefficient table
```

```
coefs = (
    pd.DataFrame({
        "coef": model.params,
        "std_err": model.bse,
        "t": model.tvalues,
        "p_value": model.pvalues
    })
    .rename_axis("term")
    .reset_index()
)

# 4) Identify significant predictors at alpha = 0.05 and 0.01
alpha_05 = coefs[(coefs["term"]!="const") & (coefs["p_value"] < 0.05)]["term"].tolist()
alpha_01 = coefs[(coefs["term"]!="const") & (coefs["p_value"] < 0.01)]["term"].tolist()

# 5) Compute simple VIFs (excluding intercept)
from statsmodels.stats.outliers_influence import variance_inflation_factor
X_no_const = boston.drop(columns=["crim"])
X_with_const = sm.add_constant(X_no_const)
vif = pd.DataFrame({
    "term": ["const"] + X_no_const.columns.tolist(),
    "VIF": [variance_inflation_factor(X_with_const.values, i) for i in range(X_with_const.sha
})

# 6) Print key results
print("Model summary (abbrev):")
print(f"R-squared: {model.rsquared:.3f},  Adj. R-squared: {model.rsquared_adj:.3f}")
print(f"F-statistic: {model.fvalue:.2f},  p(F): {model.f_pvalue:.2e}\n")

print("Coefficients (full):")
display(coefs.sort_values("p_value"))

print("\nSignificant predictors at alpha = 0.05:")
print(alpha_05)
print("\nSignificant predictors at alpha = 0.01:")
print(alpha_01)

print("\nVariance Inflation Factors (VIF):")
display(vif.sort_values("VIF", ascending=False))
```

```
Model summary (abbrev):
R-squared: 0.449,  Adj. R-squared: 0.436
```

```
F-statistic: 33.52,  p(F): 2.03e-56

Coefficients (full):
```

	term	coef	std_err	t	p_value
8	rad	0.612465	0.087536	6.996744	8.588123e-12
12	medv	-0.220056	0.059824	-3.678399	2.605302e-04
7	dis	-1.012247	0.282468	-3.583586	3.725942e-04
1	zn	0.045710	0.018790	2.432637	1.534403e-02
0	const	13.778394	7.081826	1.945599	5.227089e-02
4	nox	-9.957587	5.289824	-1.882404	6.036986e-02
11	lstat	0.138801	0.075721	1.833047	6.739844e-02
10	ptratio	-0.304073	0.186360	-1.631643	1.033932e-01
5	rm	0.628911	0.607092	1.035939	3.007385e-01
9	tax	-0.003776	0.005172	-0.729968	4.657565e-01
2	indus	-0.058350	0.083635	-0.697675	4.857094e-01
3	chas	-0.825378	1.183396	-0.697465	4.858406e-01
6	age	-0.000848	0.017948	-0.047263	9.623231e-01

```
Significant predictors at alpha = 0.05:
['zn', 'dis', 'rad', 'medv']

Significant predictors at alpha = 0.01:
['dis', 'rad', 'medv']

Variance Inflation Factors (VIF):
```

	term	VIF
0	const	608.076664
9	tax	9.195493
8	rad	7.029796
4	nox	4.546642
7	dis	4.280979
2	indus	3.983627
12	medv	3.663205
11	lstat	3.538098
6	age	3.088678
1	zn	2.323944

	term	VIF
5	rm	2.201688
10	ptratio	1.969732
3	chas	1.093242

(c)

For each predictor X_j, let $\hat{\beta}_j^{\text{uni}}$ denote the slope from the simple regression

$$\text{crim} = \beta_0 + \beta_1 X_j + \varepsilon,$$

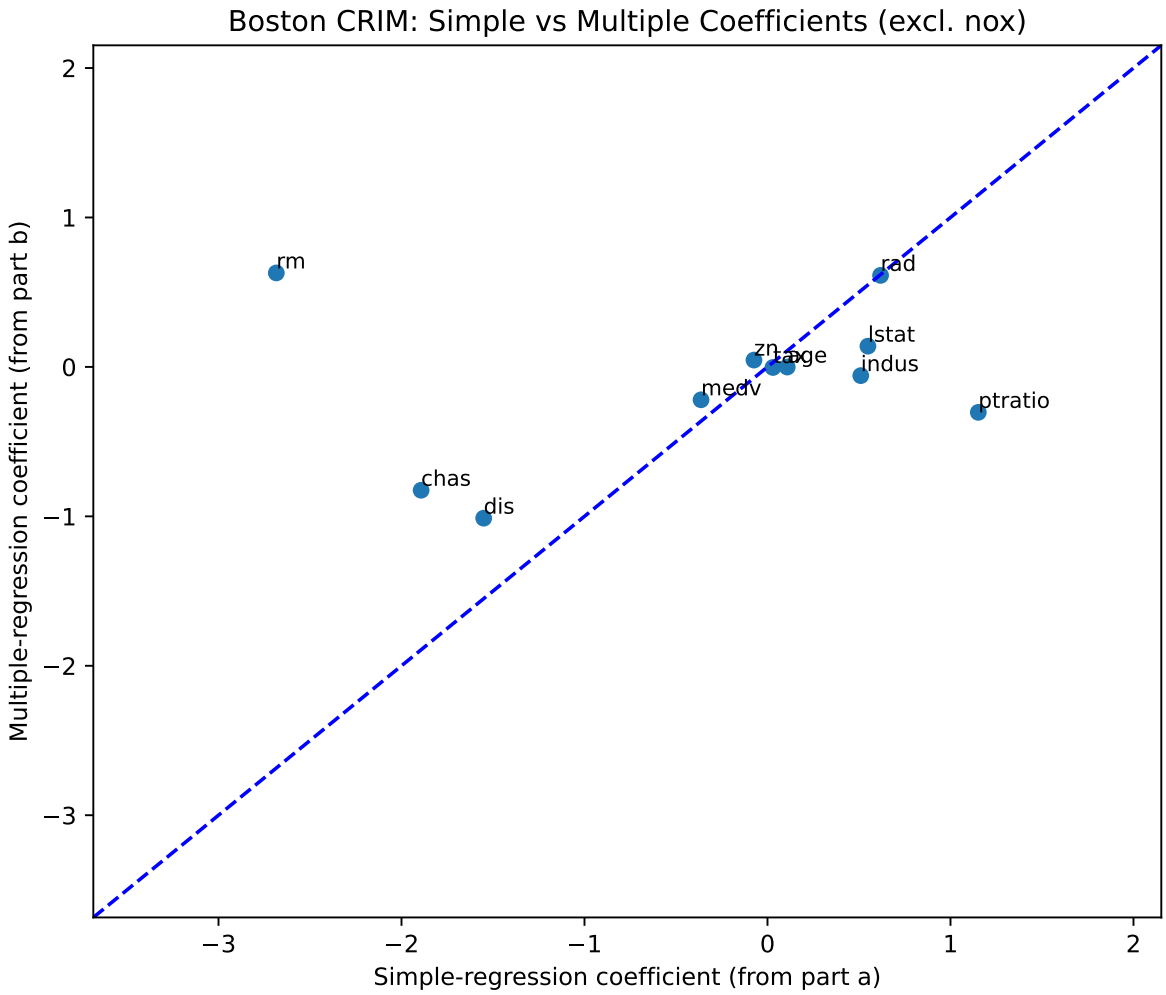
and let $\hat{\beta}_j^{\text{multi}}$ denote the coefficient of X_j from the multiple regression

$$\text{crim} = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \varepsilon.$$

We compare $\hat{\beta}_j^{\text{uni}}$ (x–axis) with $\hat{\beta}_j^{\text{multi}}$ (y–axis) for all predictors.

```
y = boston["crim"]
X_all = boston.drop(columns=["crim"])

X_all_const = sm.add_constant(X_all)
multi_fit = sm.OLS(y, X_all_const).fit()
multi_coefs = multi_fit.params.drop("const")


simple_betas = {}
for col in X_all.columns:
    Xi = sm.add_constant(X_all[[col]])
    fit_i = sm.OLS(y, Xi).fit()
    simple_betas[col] = fit_i.params[col]
simple_coefs = pd.Series(simple_betas)

coef_df = pd.DataFrame({
    "simple_beta": simple_coefs,
    "multiple_beta": multi_coefs
}).dropna().sort_index()


coef_df_trim = coef_df.drop(index=["nox"])
```

```
plt.figure(figsize=(7,6))
plt.scatter(coef_df_trim["simple_beta"], coef_df_trim["multiple_beta"])

xmin, xmax = coef_df_trim["simple_beta"].min(), coef_df_trim["simple_beta"].max()
ymin, ymax = coef_df_trim["multiple_beta"].min(), coef_df_trim["multiple_beta"].max()
lo = min(xmin, ymin) - 1
hi = max(xmax, ymax) + 1
plt.plot([lo, hi], [lo, hi], linestyle="--", color="blue")

for var, row in coef_df_trim.iterrows():
    plt.text(row["simple_beta"], row["multiple_beta"], var, fontsize=9, ha="left", va="botto

plt.xlabel("Simple-regression coefficient (from part a)")
plt.ylabel("Multiple-regression coefficient (from part b)")
plt.title("Boston CRIM: Simple vs Multiple Coefficients (excl. nox)")
plt.xlim(lo, hi)
plt.ylim(lo, hi)
plt.tight_layout()
plt.show()
```



- Many predictors show strong univariate effects that shrink toward zero once other variables are controlled, reflecting multicollinearity among features. Variables such as rad and dis remain influential in both models, indicating more independent contributions to explaining crime.

(d)

```
y = boston["crim"]
predictors = boston.drop(columns=["crim"])

nonlinear_results = []
```

```python
for col in predictors.columns:
    Xi = predictors[col]
    X_poly = pd.DataFrame({
        col: Xi,
        f"{col}2": Xi**2,
        f"{col}3": Xi**3
    })
    X_poly = sm.add_constant(X_poly)

    model = sm.OLS(y, X_poly).fit()

    pvals = model.pvalues
    nonlinear_results.append({
        "Predictor": col,
        "p(X^2)": pvals.get(f"{col}2", np.nan),
        "p(X^3)": pvals.get(f"{col}3", np.nan),
        "Adj_R2": model.rsquared_adj
    })

df_nonlinear = pd.DataFrame(nonlinear_results).set_index("Predictor")
print(df_nonlinear.sort_values("Adj_R2", ascending=False).head(10))
```

```
                p(X^2)        p(X^3)    Adj_R2
Predictor
medv       3.260523e-18  1.046510e-12  0.416735
rad        6.130099e-01  4.823138e-01  0.396451
tax        1.374682e-01  2.438507e-01  0.365110
nox        6.811300e-15  6.961110e-16  0.292777
dis        4.941214e-12  1.088832e-08  0.273509
indus      3.420187e-10  1.196405e-12  0.255234
lstat      6.458736e-02  1.298906e-01  0.213259
age        4.737733e-02  6.679915e-03  0.169296
ptratio    4.119552e-03  6.300514e-03  0.108485
rm         3.641094e-01  5.085751e-01  0.062215
```

To assess non-linear associations, we fit for each predictor a cubic polynomial model. The results show clear evidence of non-linearity for several predictors: medv (median home value): both quadratic and cubic terms are highly significant ($p < 10^{-12}$), and the adjusted R^2 rises to 0.42. This indicates a strong non-linear pattern, where the effect of housing value on crime is not simply linear. nox (nitric oxide concentration), dis (distance to employment centers), and indus (industrial proportion) also exhibit highly significant higher-order terms, confirming

non-linear relationships. age and ptratio show weaker but still significant quadratic or cubic terms ($p < 0.05$). In contrast, variables like rad, tax, and rm do not provide evidence of meaningful non-linear association, as their higher-order terms are not significant. There is substantial evidence of non-linear associations between crime rate and predictors such as medv, nox, dis, indus, age, and ptratio. These patterns suggest that flexible modeling (e.g., polynomial regression or splines) would better capture the relationship than simple linear terms.