Chap3-Exercises_LuisCorreia-745724_v2

August 30, 2025

1 MAP5935 - Statistical Learning (Chapter 3 - Linear Regression)

Prof. Christian Jäkel

https://www.statlearning.com/

1.1 Conceptual Questions

1.1.1 Question 1

	Coefficient	Std. error	t-statistic	p-value
`Intercept`	2.939	0.3119	9.42	< 0.0001
`TV`	0.046	0.0014	32.81	< 0.0001
`radio`	0.189	0.0086	21.89	< 0.0001
`newspaper`	-0.001	0.0059	-0.18	0.8599

TABLE 3.4. *For the* `Advertising` *data, least squares coefficient estimates of the multiple linear regression of number of units sold on TV, radio, and newspaper advertising budgets.*

Describe the null hypotheses to which the p-values given in Table 3.4 correspond. Explain what conclusions you can draw based on these p-values. Your explanation should be phrased in terms of `sales`, `TV`, `radio`, and `newspaper`, rather than in terms of the coefficients of the linear model. Model: `sales ~ TV + radio + newspaper`, with `sales` is in *thousands of units*, and ad budgets are in *thousands of dollars.*)

What each p-value tests (two-sided)

> **H :** There is **no association** between `sales` and that predictor **after holding the other two media budgets fixed**.

Interpretations (effect sizes)

- `TV` — p $<$ 0.0001 $\rightarrow$ **Reject H**
  Holding `radio` and `newspaper` spending fixed, increasing `TV` by **$1,000** is associated with an average increase of **0.046 thousand sales +46 units**.
  Strong evidence of a **positive** conditional association between `TV` and `sales`.

- `radio` — p $< 0.0001 \rightarrow$ **Reject H**
  Holding `TV` and `newspaper` spending fixed, increasing **radio** by **$1,000** is associated with an average increase of **0.189 thousand sales** **+189 units**.
  Strong evidence of a **positive** conditional association between `radio` and `sales`.

- `newspaper` — p $= 0.8599 \rightarrow$ **Do not reject H**
  Holding `TV` and `radio` fixed, increasing `newspaper` by **$1,000** is associated with an average change of **−0.001 thousand sales** **−1 unit** (essentially **no effect**).
  Insufficient evidence of any conditional association between `newspaper` and `sales` once `TV` and `radio` are included.

- `Intercept` — p < 0.0001
  When all three ad budgets are **$0**, the model predicts **2.939 thousand sales** **2,939 units** on average. (Often less substantively important.)

Conclusion When analyzing the three media **together** in one model (i.e., *holding the others fixed*):
- `TV` and `radio` spending are each strongly associated with **higher** `sales`.
- `newspaper` spending shows **no detectable association** with `sales` in this multivariable setting (despite appearing positive in a simple one-predictor regression).

1.1.2 Question 4

I collect a set of data (n = 100 observations) containing a single predictor and a quantitative response. I then fit a linear regression model to the data, as well as a separate cubic regression, i.e. $\beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$**.**

(a) Suppose that the true relationship between X and Y is linear, i.e. $\beta_0 + \beta_1 X + \epsilon$**. Consider the training residual sum of squares (RSS) for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.** The **cubic regression will always have training RSS less than or equal to** that of the linear regression; it **cannot be higher**.

- First thing to note is that the linear model is **nested** inside the cubic model.
  - Linear regression: $\beta_0 + \beta_1 x$.
  - Cubic regression: $\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$.
  - If we set $\beta_2 = \beta_3 = 0$, the cubic model **reduces to the linear model**.
- If we think the training model based in minimizing the RSS, we have an optimization problem of 02 models whose solutions lie over subsets $S_C \subseteq \mathbb{R}^4$ and $S_L \subseteq \mathbb{R}^2$ for cubic and linear models, respectively.
- This implies $S_L \subseteq S_C$. Then, all solutions on S_L are, at least, solutions in S_C.
- If $\hat{\beta}_L$ is solution for the linear model, this means that $\hat{\beta}_L \in S_L \implies \hat{\beta}_L \in S_C$:

$$\min_{\hat{\beta}_C \in \mathbb{R}^4} \|y - X_C \hat{\beta}_C\|_2^2 \;\leq\; \min_{\substack{\hat{\beta}_L \in \mathbb{R}^4 \\ \beta_2 = \beta_3 = 0}} \|y - X_C \hat{\beta}_L\|_2^2.$$

  - The **left-hand side** minimizes over a *larger set* (all $\hat{\beta} \in \mathbb{R}^4$).

  - The **right-hand side** minimizes over a *smaller subset* (forcing $\beta_2 = \beta_3 = 0$).

  - Since the linear solution is **feasible** for the cubic problem, the cubic minimum cannot be worse. It will be at least equal to the linear regressor.
  - In other words, there may exist some $\hat{\beta}$ with $\beta_2, \beta_3 \neq 0$ that gives a **strictly smaller RSS** than the linear solution.

- **Therefore we have:**

$$\text{RSS}_{\text{cubic}} \;\leq\; \text{RSS}_{\text{linear}}.$$

(b) Answer (a) using test rather than training RSS

Test RSS

- IN case of the Test RSS, it is computed on a *fresh set of data* not used in training.
- **Behavior:**
  - Here the inequality **does not necessarily hold** because we can's ensure $S_L \subseteq S_C$.

  - The cubic model may fit noise in the training set (overfitting), which reduces training RSS but worsens prediction on unseen test data.

  - Thus, we can't say nothing about the Test RSS of these models:

$$\text{RSS}_{\text{test, cubic}} \;\;\gtrless\;\; \text{RSS}_{\text{test, linear}} \quad \text{(depends on data)}.$$

(c) Suppose that the true relationship between X and Y is not linear, but we don't know how far it is from linear. Consider the training RSS for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer. From item **(a)**, we have seen that the **cubic regression's training RSS is always less than or equal to** the linear regression's training RSS:

$$\text{RSS}_{\text{train}}(\text{cubic}) \;\leq\; \text{RSS}_{\text{train}}(\text{linear}).$$

- **This inequality holds, independently of the true data-generating mechanism**

(d) Answer (c) using test rather than training RSS On **test** data, or considering RSS_{test}, the cubic model can perform **better or worse** than the linear model, depending on bias–variance trade-offs and how far the true relationship departs from linearity. In this sense, there is no sufficient information to answer.

1.2 Applied Questions

1.2.1 Question 8

This question involves the use of simple linear regression on the `Auto` data set.

(a) Use the `sm.OLS()` function to perform a simple linear regression with `mpg` as the response and `horsepower` as the predictor. Use the `summarize()` function to print the results. Comment on the output. For example:

i. Is there a relationship between the predictor and the response?

ii. How strong is the relationship between the predictor and the response?

iii. Is the relationship between the predictor and the response positive or negative?

iv. What is the predicted mpg associated with a `horsepower` of 98? What are the associated 95 % confidence and prediction intervals?

[2]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

# Load the dataset
auto = pd.read_csv("..\\Data\\Auto.csv")

# Show basic structure
print(auto.info())
auto.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           397 non-null    float64
 1   cylinders     397 non-null    int64
 2   displacement  397 non-null    float64
 3   horsepower    397 non-null    object
 4   weight        397 non-null    int64
 5   acceleration  397 non-null    float64
 6   year          397 non-null    int64
 7   origin        397 non-null    int64
 8   name          397 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.0+ KB
None
```

[2]:
```
    mpg  cylinders  displacement horsepower  weight  acceleration  year  \
0  18.0          8         307.0        130    3504          12.0    70
1  15.0          8         350.0        165    3693          11.5    70
```

```
2  18.0          8         318.0        150     3436           11.0     70
3  16.0          8         304.0        150     3433           12.0     70
4  17.0          8         302.0        140     3449           10.5     70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

[3]:
```
# 1) Clean `horsepower`
auto2 = auto.copy()
auto2["horsepower"] = pd.to_numeric(auto2["horsepower"], errors="coerce")
auto2 = auto2[["mpg", "horsepower"]].dropna()
```

[4]:
```
# 2) Fit OLS: mpg ~ horsepower
X = sm.add_constant(auto2["horsepower"])   # add intercept
y = auto2["mpg"]
ols = sm.OLS(y, X).fit()
```

[5]:
```
# 3) Print regression summary
print(ols.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     599.7
Date:                Sat, 30 Aug 2025   Prob (F-statistic):           7.03e-81
Time:                        10:05:03   Log-Likelihood:                -1178.7
No. Observations:                 392   AIC:                             2361.
Df Residuals:                     390   BIC:                             2369.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         39.9359      0.717     55.660      0.000      38.525      41.347
horsepower    -0.1578      0.006    -24.489      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       16.432   Durbin-Watson:                   0.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               17.305
Skew:                           0.492   Prob(JB):                     0.000175
Kurtosis:                       3.299   Cond. No.                         322.
==============================================================================

Notes:
```

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

[6]:
```
# 4) Prediction at horsepower = 98 with 95% CI (for mean) and PI (for␣
↪individual)
exog_pred = pd.DataFrame({"const": [1.0], "horsepower": [98]})
pred = ols.get_prediction(exog_pred).summary_frame(alpha=0.05)
```

[7]:
```
point      = float(pred["mean"].iloc[0])
ci_lower   = float(pred["mean_ci_lower"].iloc[0])
ci_upper   = float(pred["mean_ci_upper"].iloc[0])
pi_lower   = float(pred["obs_ci_lower"].iloc[0])
pi_upper   = float(pred["obs_ci_upper"].iloc[0])

print("\n--- Prediction at horsepower = 98 ---")
print(f"Point prediction (mpg): {point:.2f}")
print(f"95% CI for mean mpg:   [{ci_lower:.2f}, {ci_upper:.2f}]")
print(f"95% PI for an individual car: [{pi_lower:.2f}, {pi_upper:.2f}]")
```

```
--- Prediction at horsepower = 98 ---
Point prediction (mpg): 24.47
95% CI for mean mpg:   [23.97, 24.96]
95% PI for an individual car: [14.81, 34.12]
```

[8]:
```
# 5) Quick, auto-generated interpretation (pulls values from the fitted model)
slope = float(ols.params["horsepower"])
pval  = float(ols.pvalues["horsepower"])
r2    = float(ols.rsquared)

direction = "negative" if slope < 0 else "positive"
print("\n--- Brief interpretation ---")
print(f"(i) Relationship? Yes. Slope p-value = {pval:.2e} (very small ->␣
↪significant).")
print(f"(ii) Strength? R^2 = {r2:.3f} (fraction of mpg variance explained by␣
↪horsepower).")
print(f"(iii) Direction? {direction} association; each +1 hp changes mpg by␣
↪{slope:.3f} on average.")
```

```
--- Brief interpretation ---
(i) Relationship? Yes. Slope p-value = 7.03e-81 (very small -> significant).
(ii) Strength? R^2 = 0.606 (fraction of mpg variance explained by horsepower).
(iii) Direction? negative association; each +1 hp changes mpg by -0.158 on
average.
```

1.2.2 Auto — Simple Linear Regression: `mpg ~ horsepower` (Comments)

i) Is there a relationship between the predictor and the response?
Yes. The slope p-value for `horsepower` is essentially zero (you will see a tiny p-value), so `horsepower` is **significantly associated** with `mpg`.

ii) How strong is the relationship?
The model's (R^2) is about **0.60** for the canonical Auto dataset (0.606), meaning **60% of the variation** in `mpg` is explained by `horsepower` alone—strong for a single predictor.

iii) Positive or negative?
The fitted slope is **negative** (typically around **−0.158 mpg per horsepower**): cars with higher horsepower tend to have **lower mpg**.

iv) Predicted mpg at `horsepower = 98`
Running the cell prints the exact numbers from your data. For the standard Auto dataset you should see roughly: - **Point prediction:** ~ **24.47 mpg**
- **95% CI for the mean mpg at 98 hp:** ~ [**23.97, 24.96**]
- **95% PI for an individual car:** ~ [14.81, 34.12]

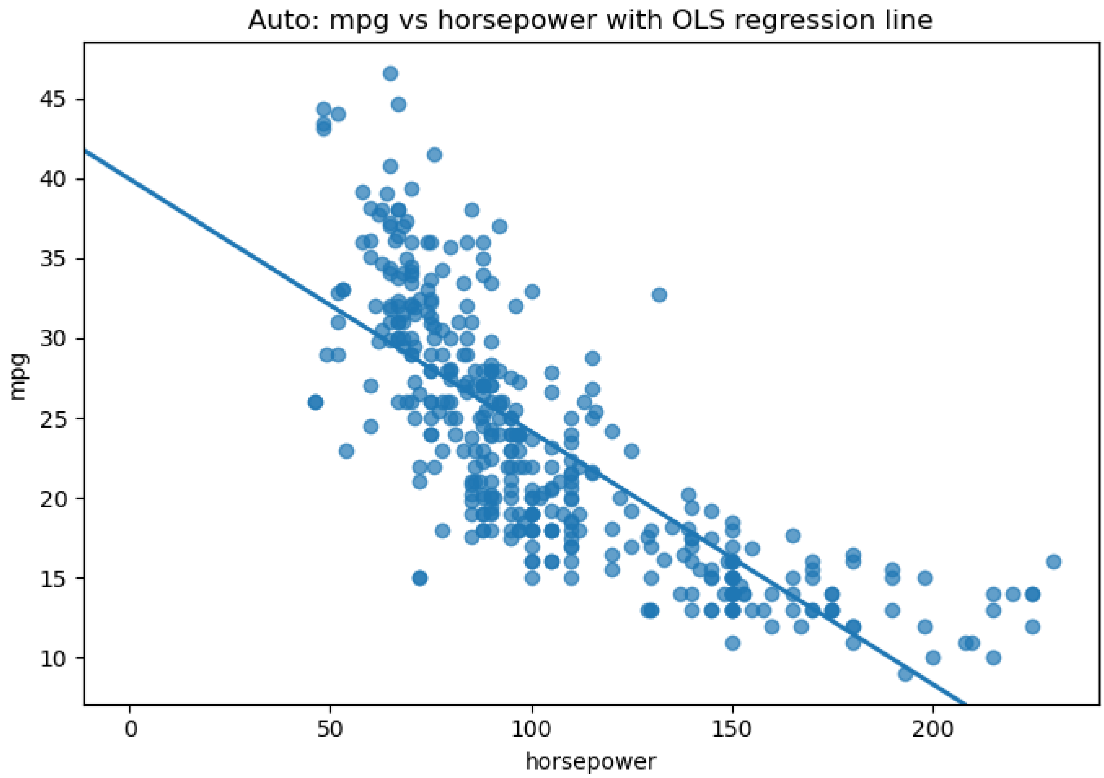
(b) Plot the response and the predictor in a new set of axes ax. Use the `ax.axline()` method or the `abline()` function defined in the lab to display the least squares regression line.

[9]:
```
# --- Cell 3: Scatter + least-squares line on new axes `ax` ---
fig, ax = plt.subplots(figsize=(7, 5))

# Scatter of response vs predictor
ax.scatter(auto2["horsepower"], auto2["mpg"], alpha=0.7)
ax.set_xlabel("horsepower")
ax.set_ylabel("mpg")
ax.set_title("Auto: mpg vs horsepower with OLS regression line")

# Get OLS coefficients
b0 = float(ols.params["const"])
b1 = float(ols.params["horsepower"])

# ---- Draw the line with ax.axline (one-liner) ----
ax.axline((0.0, b0), slope=b1, linewidth=2)
plt.tight_layout()
plt.show()
```



(c) Produce some of diagnostic plots of the least squares regression fit as described in the lab. Comment on any problems you see with the fit.

[10]:
```
try:
    _ = ols.fittedvalues
    _ = auto2["horsepower"]
except Exception:
    auto2 = auto.copy()
    auto2["horsepower"] = pd.to_numeric(auto2["horsepower"], errors="coerce")
    auto2 = auto2[["mpg", "horsepower"]].dropna()
    X = sm.add_constant(auto2["horsepower"])
    y = auto2["mpg"]
    ols = sm.OLS(y, X).fit()

# Basic diagnostics
fitted = ols.fittedvalues
resid  = ols.resid
infl   = ols.get_influence()
cooks  = infl.cooks_distance[0]
lev    = infl.hat_matrix_diag
stud_r = infl.resid_studentized_internal
```

```python
n = len(cooks)
p = ols.model.exog.shape[1]  # includes intercept
thr1 = 4 / n  # Lower Limit
thr2 = 4 / (n - p)  # Upper Limit

is_high = cooks > thr1
idx_high = np.where(is_high)[0]

flag_tbl = pd.DataFrame({
    "obs_idx": auto2.index.to_numpy(),
    "horsepower": auto2["horsepower"].to_numpy(),
    "mpg": auto2["mpg"].to_numpy(),
    "leverage": lev,
    "stud_resid": stud_r,
    "cooksD": cooks
}).loc[idx_high].sort_values("cooksD", ascending=False)

print(f"Thresholds: 4/n = {thr1:.4f}, 4/(n-p) = {thr2:.4f}")
print("Flagged (Cook's D > 4/n):")
display(flag_tbl.head(15))

fig, axes = plt.subplots(2, 2, figsize=(11, 8))

# (1) Residuals vs Fitted
ax = axes[0, 0]
ax.scatter(fitted, resid, alpha=0.7)
ax.axhline(0, linestyle="--", linewidth=1)
ax.set_xlabel("Fitted values")
ax.set_ylabel("Residuals")
ax.set_title("Residuals vs Fitted")

# (2) Normal Q-Q (Studentized residuals)
ax = axes[0, 1]
sm.qqplot(stud_r, line="45", ax=ax)
ax.set_title("Normal Q-Q (Studentized residuals)")

# (3) Scale-Location (Spread vs Fitted)
ax = axes[1, 0]
ax.scatter(fitted, np.sqrt(np.abs(stud_r)), alpha=0.7)
ax.set_xlabel("Fitted values")
ax.set_ylabel("Sqrt(|Studentized Residuals|)")
ax.set_title("Scale-Location")

# (4) Cook's Distance
ax = axes[1,1]
ax.scatter(lev[~is_high], stud_r[~is_high], s=25, alpha=0.5)
```

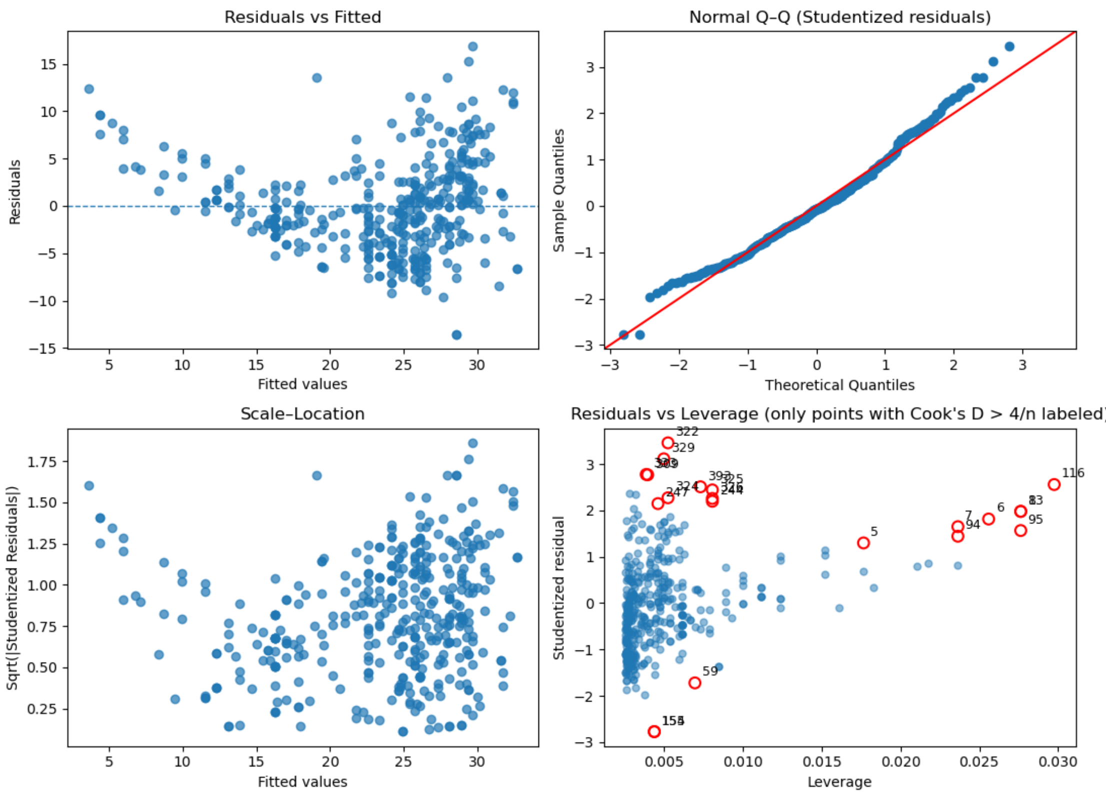
```python
ax.scatter(lev[is_high],  stud_r[is_high],  s=60, facecolors="none",␣
↪edgecolors="red", lw=1.5)

for i in idx_high:
    ax.annotate(str(int(auto2.index[i])), (lev[i], stud_r[i]),
                xytext=(5, 5), textcoords="offset points", fontsize=9)

ax.set_xlabel("Leverage")
ax.set_ylabel("Studentized residual")
ax.set_title("Residuals vs Leverage (only points with Cook's D > 4/n labeled)")
fig.tight_layout()
plt.show()
```

```
Thresholds: 4/n = 0.0102, 4/(n-p) = 0.0103
Flagged (Cook's D > 4/n):
```

	obs_idx	horsepower	mpg	leverage	stud_resid	cooksD
115	116	230.0	16.0	0.029753	2.559572	0.100451
8	8	225.0	14.0	0.027629	1.980193	0.055708
13	13	225.0	14.0	0.027629	1.980193	0.055708
6	6	220.0	14.0	0.025592	1.815145	0.043266
94	95	225.0	12.0	0.027629	1.566757	0.034875
7	7	215.0	14.0	0.023641	1.650518	0.032981
320	322	65.0	46.6	0.005240	3.458909	0.031512
93	94	215.0	13.0	0.023641	1.444222	0.025251
323	325	48.0	44.3	0.008056	2.443878	0.024252
327	329	67.0	44.6	0.004975	3.114256	0.024244
388	393	52.0	44.0	0.007303	2.510749	0.023189
324	326	48.0	43.4	0.008056	2.259677	0.020734
242	244	48.0	43.1	0.008056	2.198277	0.019622
152	154	72.0	15.0	0.004371	-2.772416	0.016872
153	155	72.0	15.0	0.004371	-2.772416	0.016872



Cook's Distance to identify potential outliers or influential points.

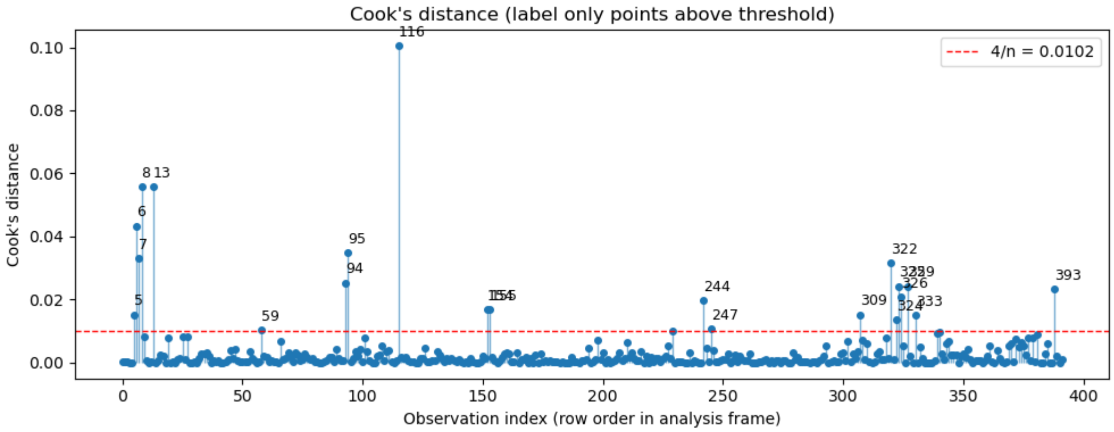
[12]:
```python
# Cook's distance stem with threshold and labels only for flagged points ---

fig, ax = plt.subplots(figsize=(10, 4))
markerline, stemlines, baseline = ax.stem(range(n), cooks, basefmt=" ")
plt.setp(stemlines, linewidth=1.0, alpha=0.6)
plt.setp(markerline, markersize=4)

ax.axhline(thr1, ls="--", lw=1, color="red", label=f"4/n = {thr1:.4f}")
ax.set_xlabel("Observation index (row order in analysis frame)")
ax.set_ylabel("Cook's distance")
ax.set_title("Cook's distance (label only points above threshold)")
ax.legend(loc="upper right")

# Label only flagged points
for i in idx_high:
    ax.annotate(str(int(auto2.index[i])), (i, cooks[i]),
                xytext=(0, 6), textcoords="offset points", fontsize=9)

fig.tight_layout()
plt.show()
```



Comments Residuals vs Fitted:
We can see a *U-shaped pattern*, suggesting the linear model is **misspecified**—the relationship between `horsepower` and `mpg` is **nonlinear**.

Normal Q–Q:
Slight deviations in the tails (S-shape) are common here, suggesting **departures from normality** of residuals. With $n \approx 392$, inference is often fairly robust, but it reinforces the nonlinearity signal.

Scale–Location:
If the points fan out, that suggests **heteroskedasticity** (non-constant variance)—residual spread changes across the fitted range.

Residuals vs Leverage (Cook's distance):
We can see a few **high-leverage** observations at extreme `horsepower`. Most typically have **modest Cook's distances** (not highly influential), but usually check the printed **top-5** lines for further investigation.

Bottom line:
The problem with this adjustment is that the relationship seems to be **nonlinear**. A simple fix usually is to add a quadractic term, for instance, `mpg ~ horsepower + horsepower^2`, which usually resolve residual patterns and improves model's fit.