Understanding Scatter Matrix Plot Formatting

What is a Scatter Matrix?

A scatter matrix is a grid of plots where:

- Each cell shows the relationship between two variables
- The diagonal shows histograms of individual variables
- It's like a correlation heatmap but with actual data points

Example Scatter Matrix Structure

Let's say you have 4 variables: **crim**, **zn**, **age**, **medv**

```
        crim    zn      age     medv
crim    [hist]  [plot]  [plot]  [plot]
zn      [plot]  [hist]  [plot]  [plot]
age     [plot]  [plot]  [hist]  [plot]
medv    [plot]  [plot]  [plot]  [hist]
```

This creates a 4×4 grid (16 plots total).

Breaking Down the Code

```python
n = len(sampled_data.columns)
```

What it does: Gets the number of columns (variables) **Example**: If you have 4 variables, n = 4

```python
for i in range(n):
```

What it does: Loops through each row/column position (0, 1, 2, 3)

Part 1: Formatting Left Column (Y-axis labels)

```python
v = axs[i, 0]  # Gets the leftmost plot in row i
```

Visual Example:

```
Position [0,0] → First plot in first row
Position [1,0] → First plot in second row
Position [2,0] → First plot in third row
Position [3,0] → First plot in fourth row
```

```
python

v.yaxis.label.set_rotation(0)
```

What it does: Makes Y-axis labels horizontal (0 degrees rotation) **Why**: By default, Y-axis labels might be rotated. This makes them readable.

Before:

```
c ← rotated text
r
i
m
```

After:

```
crim ← horizontal text
```

```
python

v.yaxis.label.set_ha('right')
```

What it does: Aligns Y-axis labels to the right **Why**: Makes labels line up nicely with the plot area

```
python

v.set_yticks(())
```

What it does: Removes Y-axis tick marks (the small lines and numbers) **Why**: Cleaner appearance, focus on relationships not exact values

Part 2: Formatting Bottom Row (X-axis labels)

```python
h = axs[n-1, i]  # Gets the bottom plot in column i
```

Visual Example (if n=4):

```
Position [3,0] → Bottom plot in first column
Position [3,1] → Bottom plot in second column
Position [3,2] → Bottom plot in third column
Position [3,3] → Bottom plot in fourth column
```

```python
h.xaxis.label.set_rotation(90)
```

What it does: Rotates X-axis labels vertically (90 degrees) **Why**: Prevents long variable names from overlapping

Before:

```
crim zn age medv  ← overlapping text
```

After:

```
c z a m
r n g e
i   e d
m     v  ← vertical text
```

```python
h.set_xticks(())
```

What it does: Removes X-axis tick marks **Why**: Cleaner appearance

Visual Before and After

Before Formatting:

```
[messy plot with overlapping labels, rotated text, lots of tick marks]
```

After Formatting:

[clean plot with readable labels, no distracting tick marks]

Why This Formatting Matters

1. **Readability**: You can actually read the variable names
2. **Clean Appearance**: No cluttered tick marks
3. **Professional Look**: Properly aligned labels
4. **Focus**: Emphasis on data patterns, not axis details

Real-World Analogy

Think of it like organizing a photo album:

- **Before**: Photos scattered, labels sideways, sticky notes everywhere
- **After**: Photos neatly arranged, clear labels, clean presentation

The data relationships are the same, but now you can actually see and understand them clearly!

Key Takeaway

This code is purely cosmetic - it's making the plot look professional and readable. The actual data and relationships don't change, but now humans can interpret the visualization much more easily.