Basics of hierarchical clustering

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

datacamp

Creating a distance matrix using linkage

```
scipy.cluster.hierarchy.linkage(observations,
                                method='single',
                                metric='euclidean',
                                optimal_ordering=False
)
```

- `method` : how to calculate the proximity of clusters
- `metric` : distance metric
- `optimal_ordering` : order data points

Which method should use?

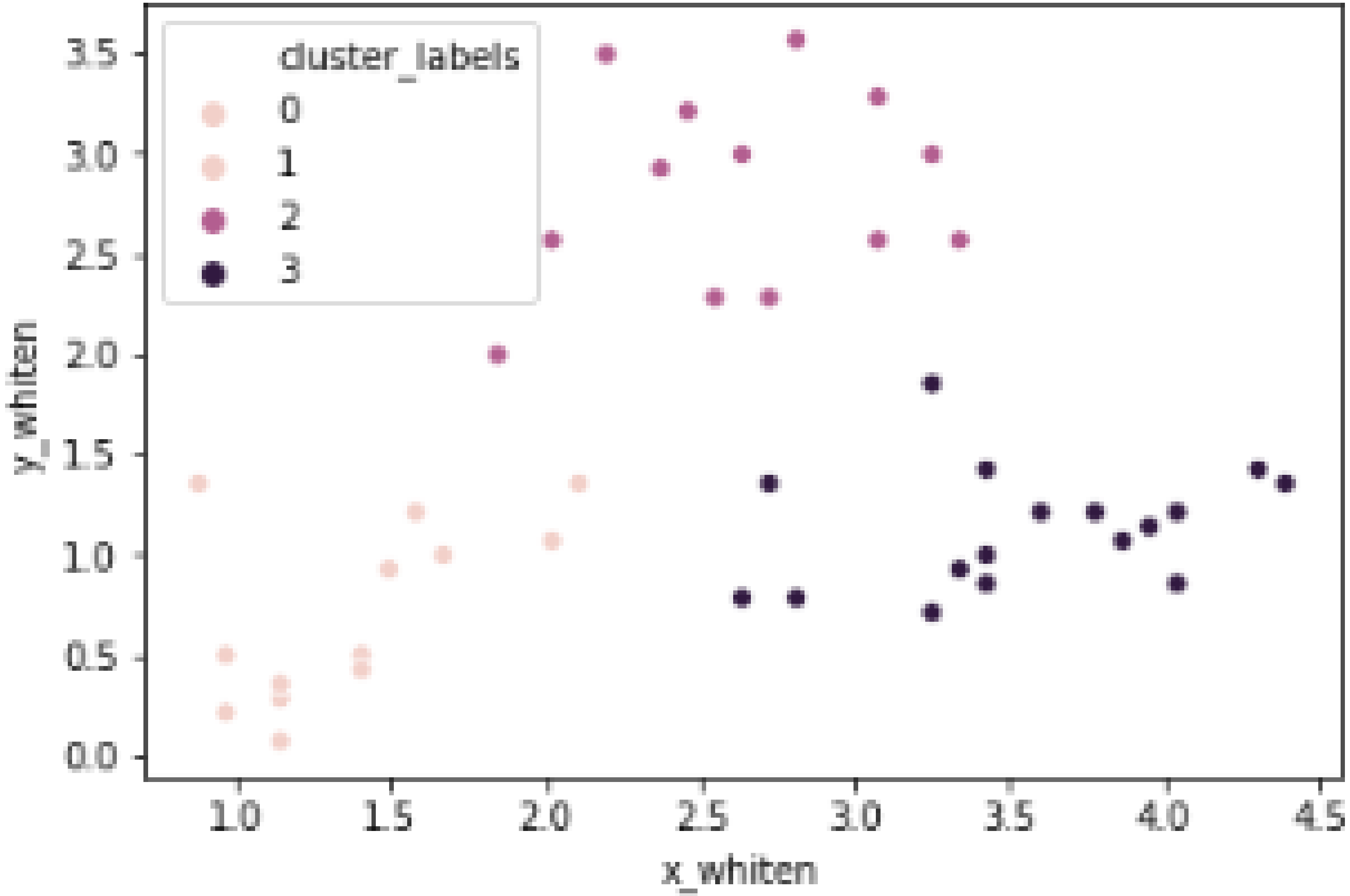
- single: based on two closest objects
- complete: based on two farthest objects
- average: based on the arithmetic mean of all objects
- centroid: based on the geometric mean of all objects
- median: based on the median of all objects
- ward: based on the sum of squares

Create cluster labels with fcluster

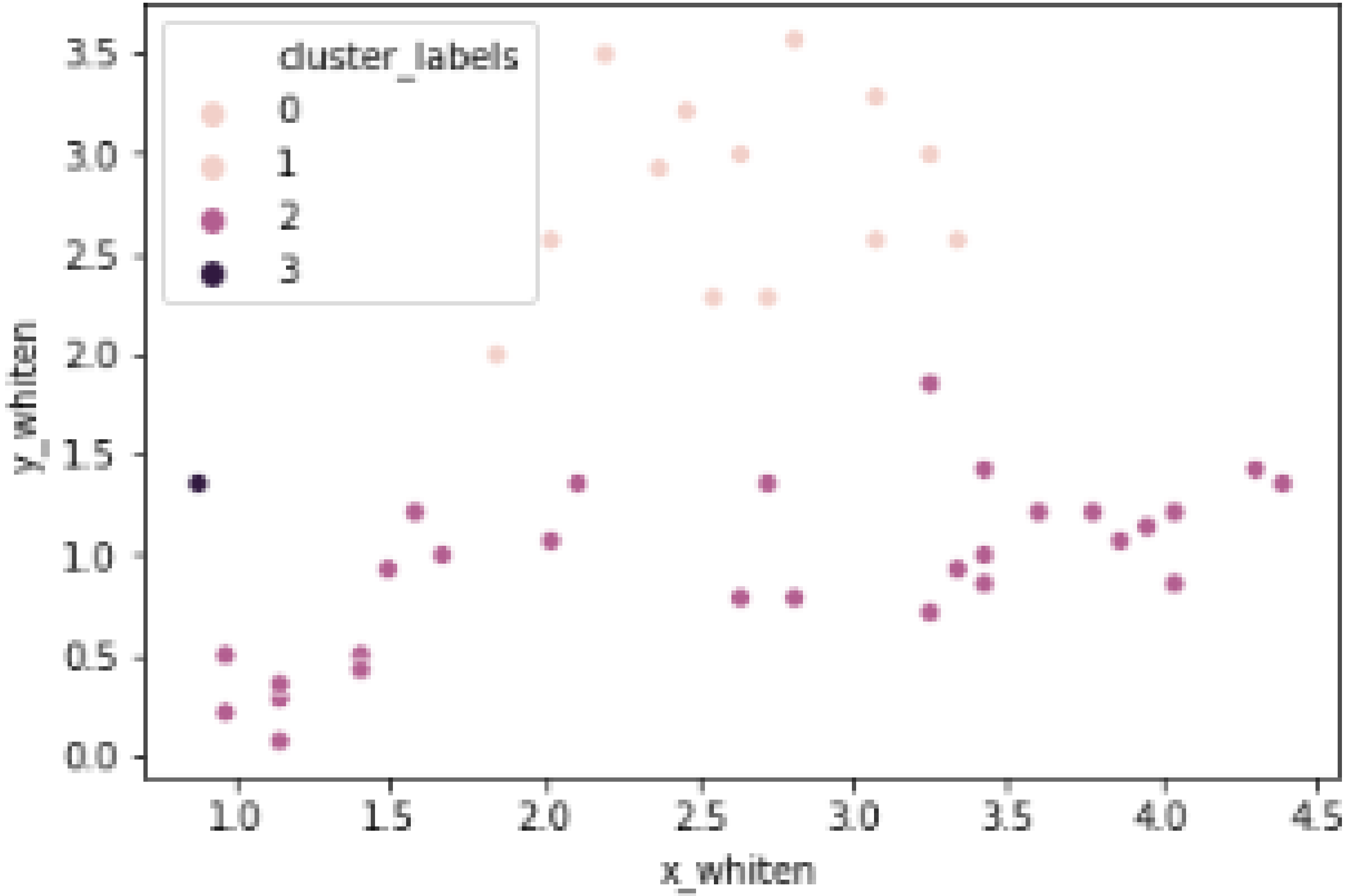
```
scipy.cluster.hierarchy.fcluster(distance_matrix,
                                 num_clusters,
                                 criterion
)
```

- `distance_matrix` : output of `linkage()` method
- `num_clusters` : number of clusters
- `criterion` : how to decide thresholds to form clusters

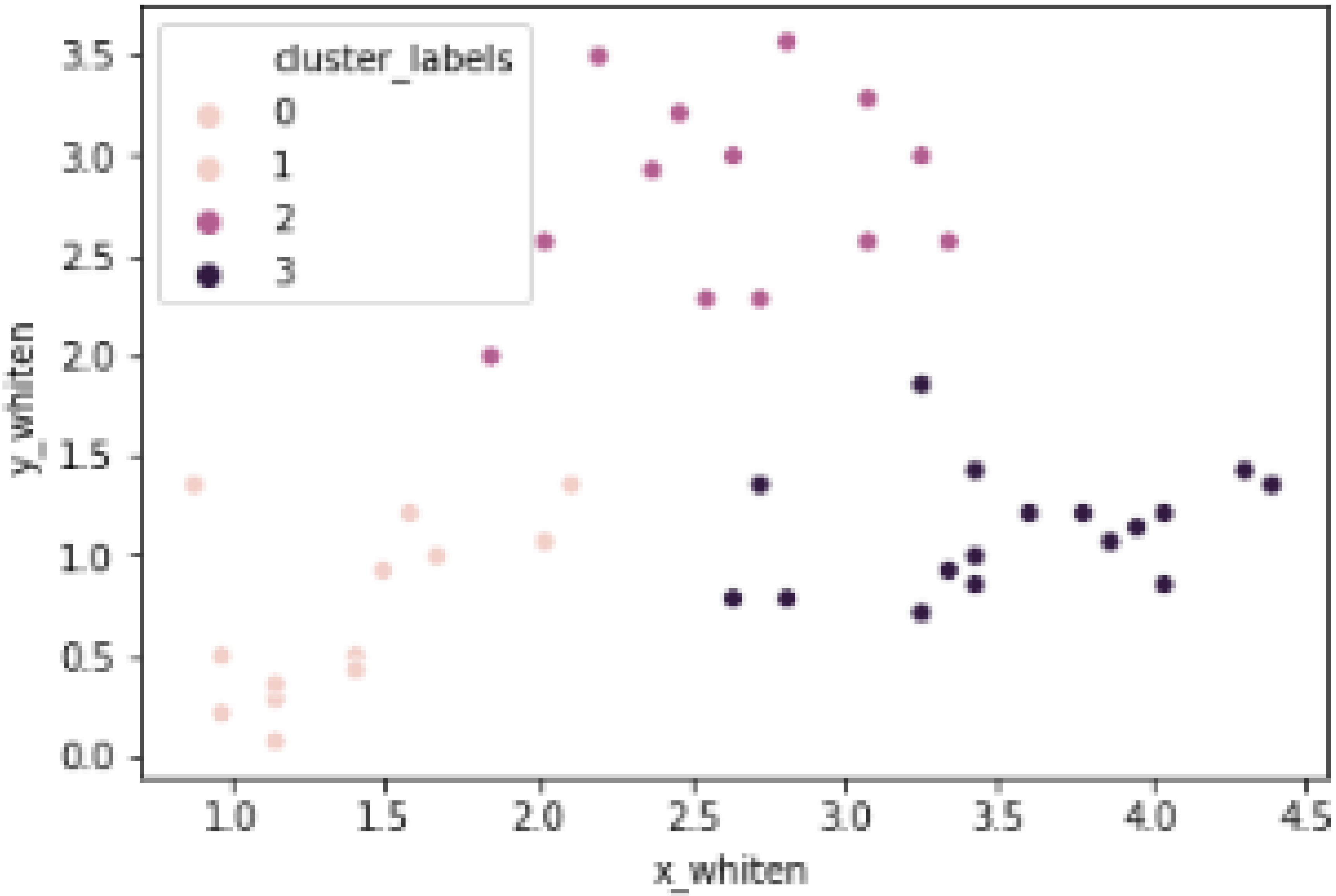
Hierarchical clustering with ward method



Hierarchical clustering with single method



Hierarchical clustering with complete method



Final thoughts on selecting a method

- No one right method for all
- Need to carefully understand the distribution of data

Let's try some exercises

CLUSTER ANALYSIS IN PYTHON

Visualize clusters

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

Why visualize clusters?

- Try to make sense of the clusters formed
- An additional step in validation of clusters
- Spot trends in data

An introduction to seaborn

- `seaborn` : a Python data visualization library based on `matplotlib`
- Has better, easily modifiable aesthetics than matplotlib!
- Contains functions that make data visualization tasks easy in the context of data analytics
- Use case for clustering: `hue` parameter for plots

Visualize clusters with matplotlib

```python
from matplotlib import pyplot as plt
```

```python
df = pd.DataFrame({'x': [2, 3, 5, 6, 2],
                   'y': [1, 1, 5, 5, 2],
                   'labels': ['A', 'A', 'B', 'B', 'A']})
colors = {'A':'red', 'B':'blue'}
df.plot.scatter(x='x',
                y='y',
                c=df['labels'].apply(lambda x: colors[x]))
plt.show()
```

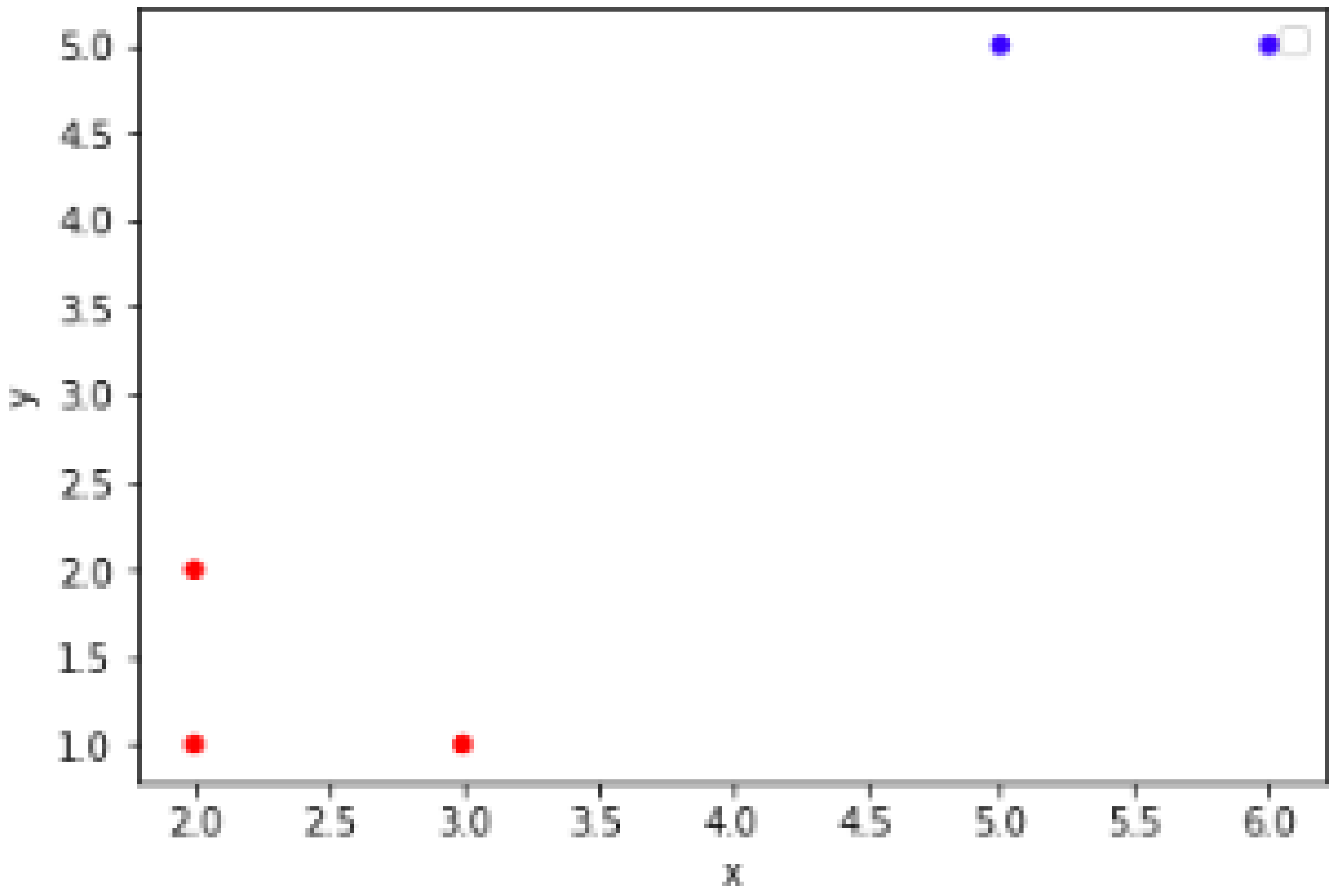
Visualize clusters with seaborn

```python
from matplotlib import pyplot as plt
import seaborn as sns
```

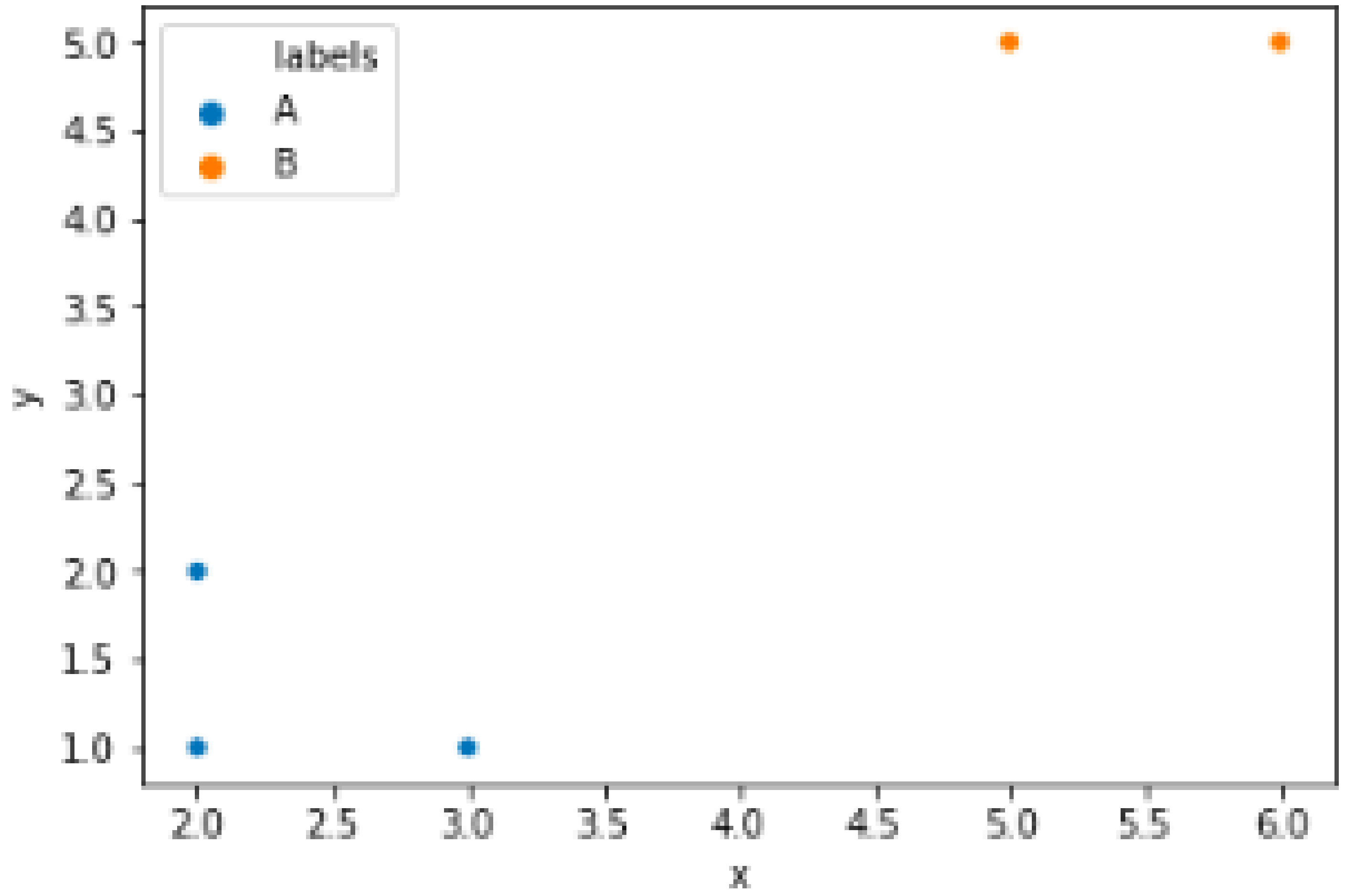
```python
df = pd.DataFrame({'x': [2, 3, 5, 6, 2],
                   'y': [1, 1, 5, 5, 2],
                   'labels': ['A', 'A', 'B', 'B', 'A']})
sns.scatterplot(x='x',
                y='y',
                hue='labels',
                data=df)
plt.show()
```

Comparison of both methods of visualization

MATPLOTLIB PLOT



SEABORN PLOT



Next up: Try some visualizations

CLUSTER ANALYSIS IN PYTHON

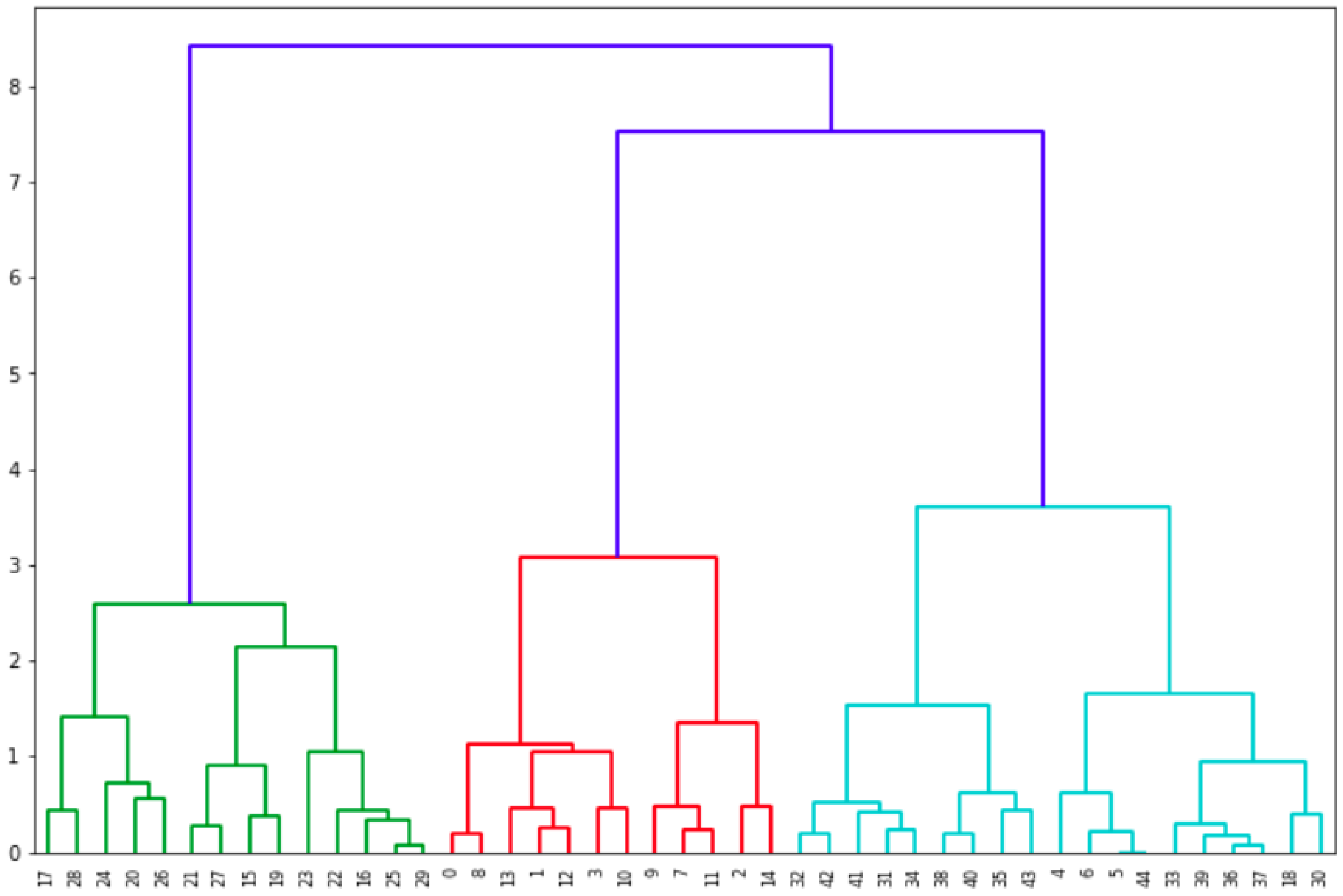
How many clusters?

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

Introduction to dendrograms

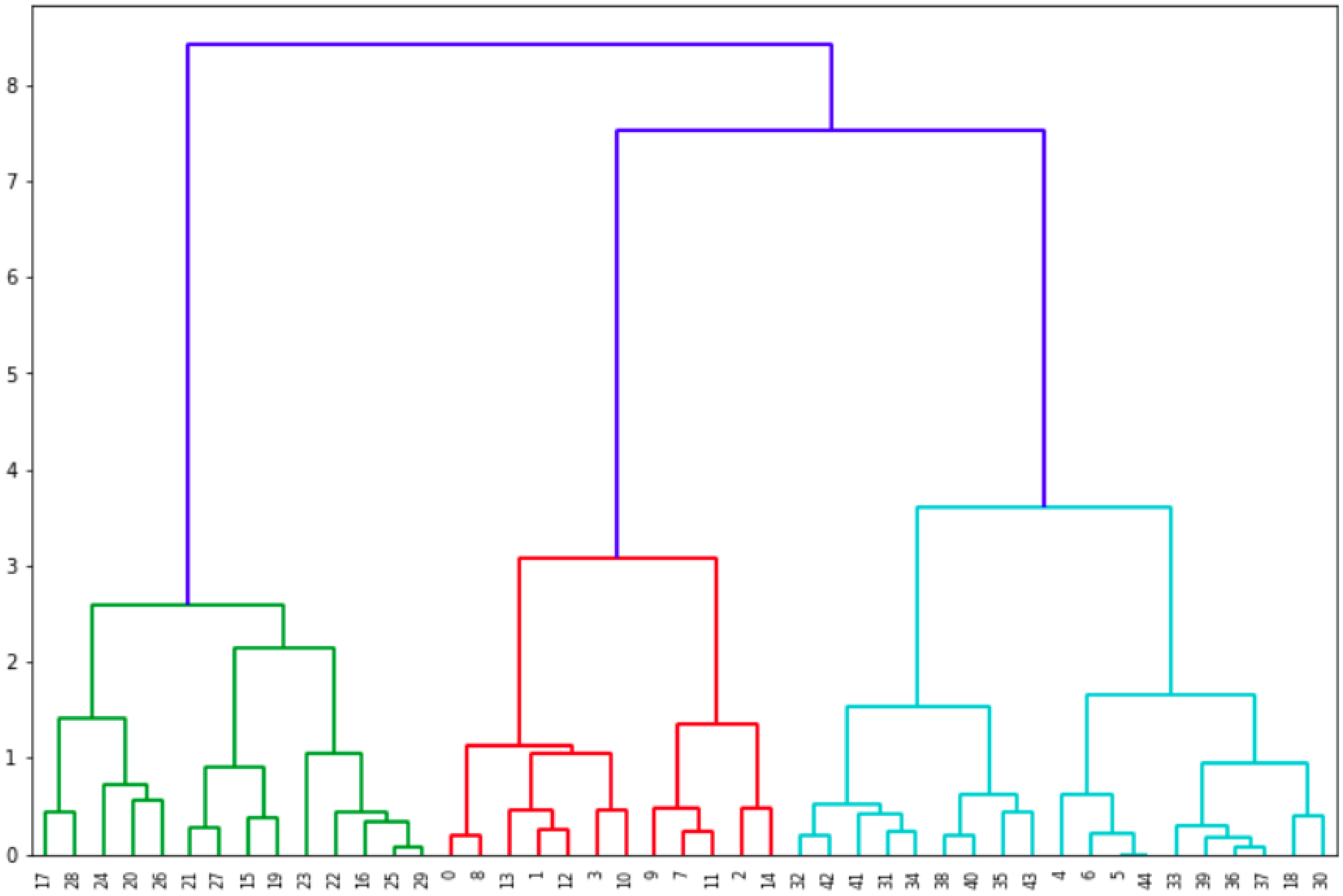
- Strategy till now - decide clusters on visual inspection
- Dendrograms help in showing progressions as clusters are merged
- A dendrogram is a branching diagram that demonstrates how each cluster is composed by branching out into its child nodes

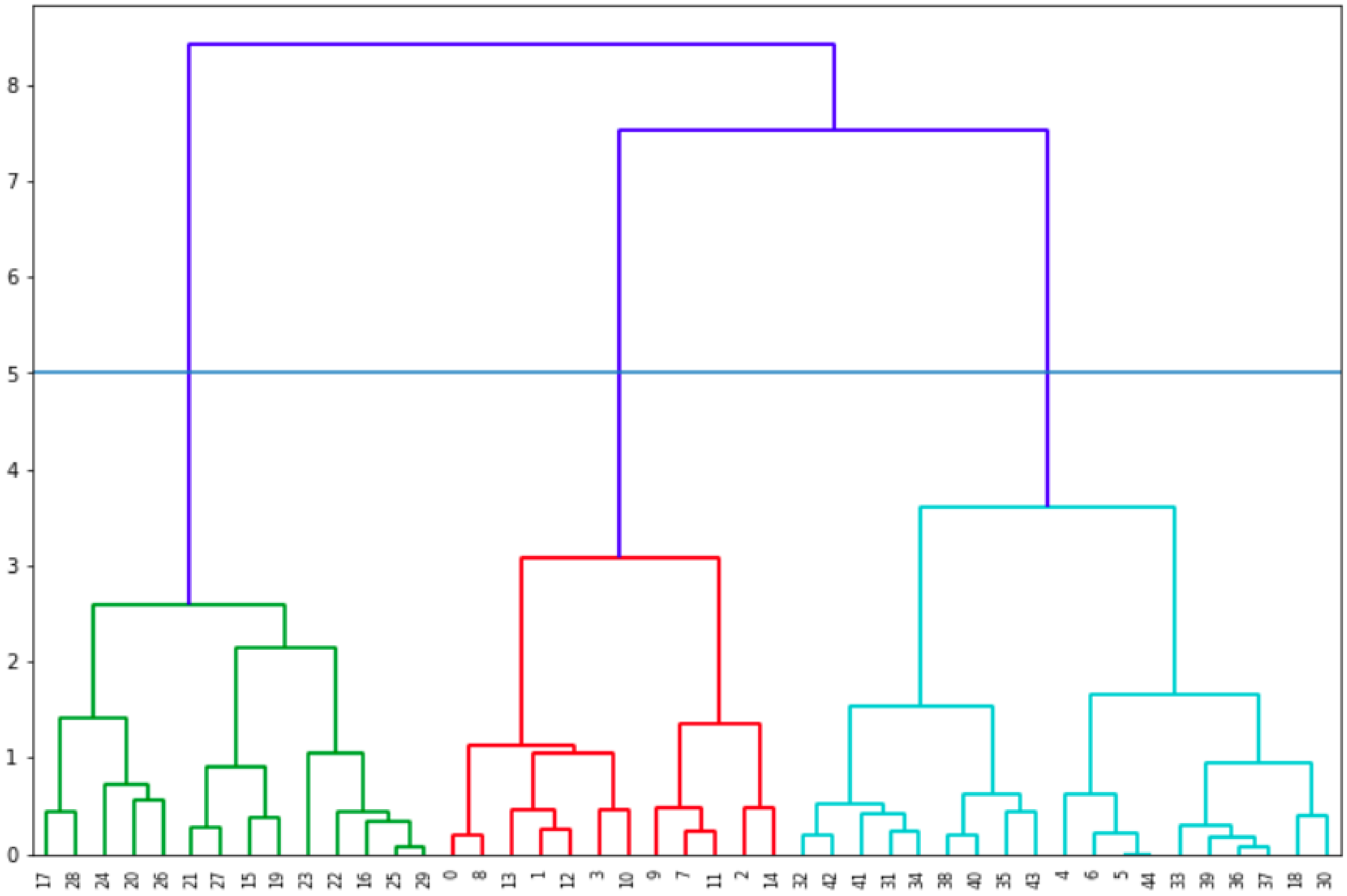


Create a dendrogram in SciPy

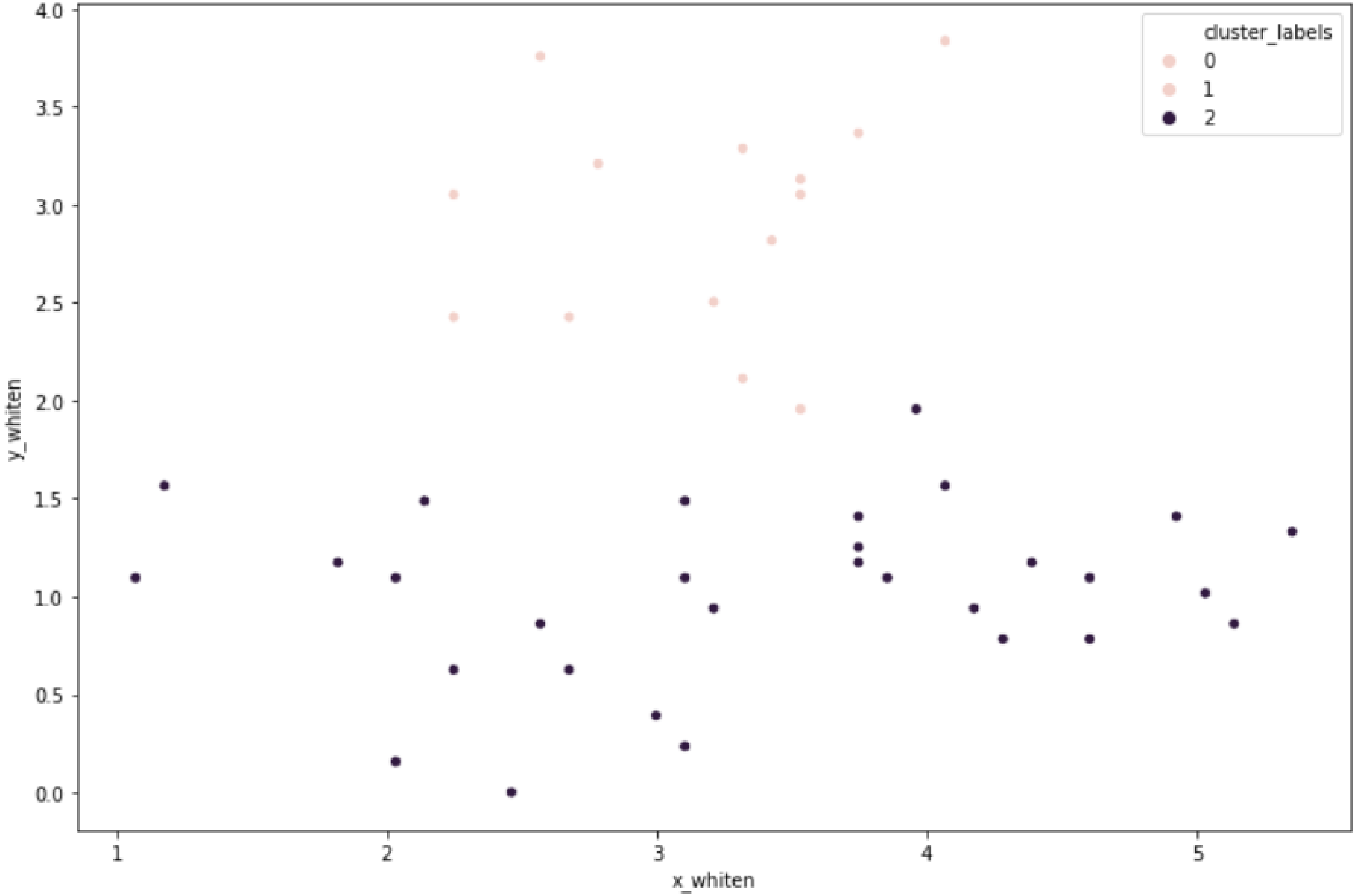
```python
from scipy.cluster.hierarchy import dendrogram
```

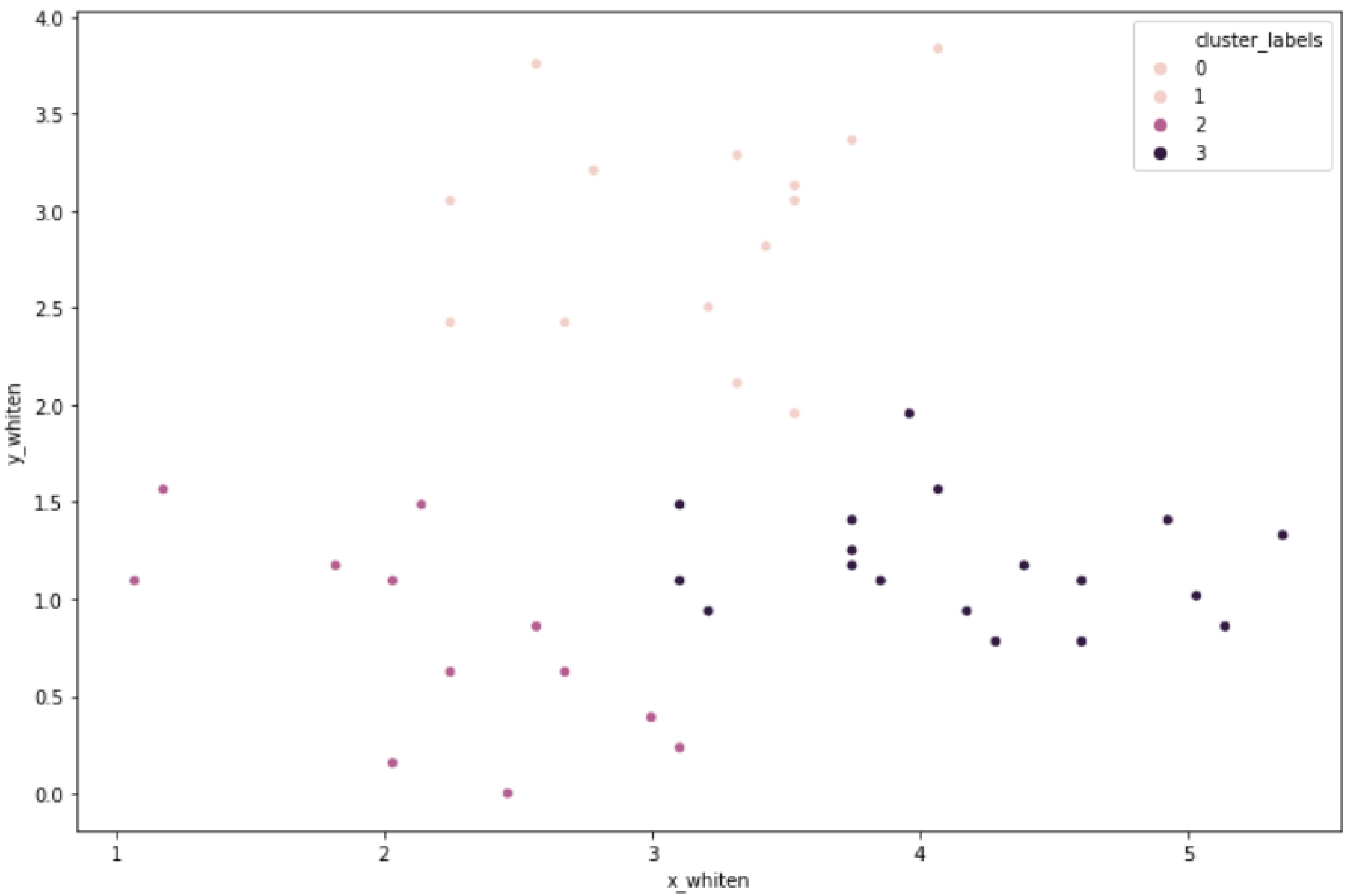
```python
Z = linkage(df[['x_whiten', 'y_whiten']],
            method='ward',
            metric='euclidean')
dn = dendrogram(Z)
plt.show()
```

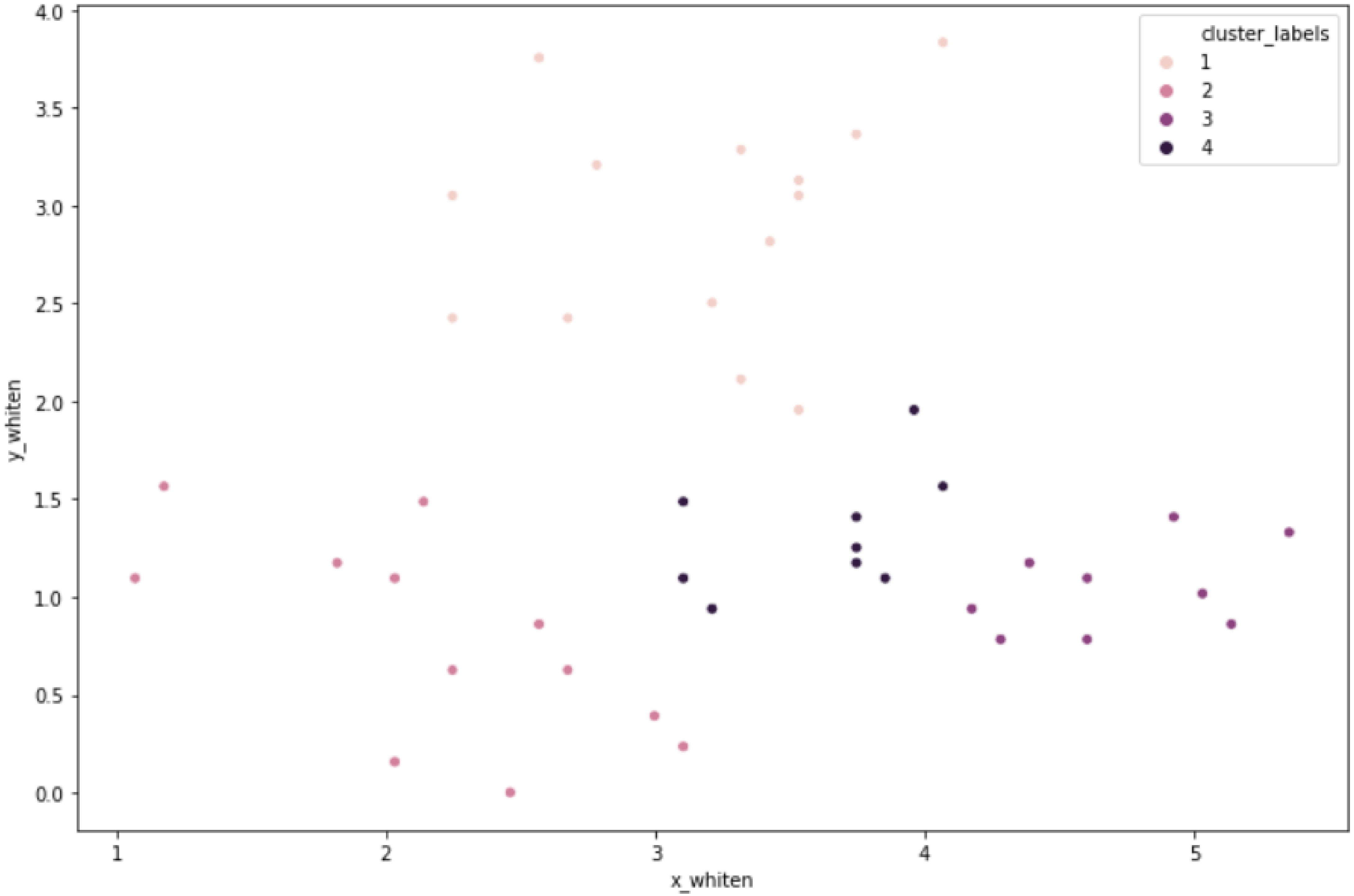

8
7
6
5
4
3
2
1
0
17
28
24
20
26
21
27
15
19
23
22
16
25
29
0
8
13
1
12
3
10
9
7
11
2
14
32
42
41
31
34
38
40
35
43
4
6
5
44
33
39
36
37
18
30



0
1
2
3
4
5
6
7
8
17
28
24
20
26
21
27
15
19
23
22
16
25
29
0
8
13
1
12
3
10
9
7
11
2
14
32
42
41
31
34
38
40
35
43
4
6
5
44
33
39
36
37
18
30


cluster_labels
0
1
2
y_whiten
x_whiten
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5


cluster_labels
0
1
2
3
y_whiten
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x_whiten
1
2
3
4
5


cluster_labels
1
2
3
4
y_whiten
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
x_whiten
1
2
3
4
5

Next up - try some exercises

CLUSTER ANALYSIS IN PYTHON

Limitations of hierarchical clustering

CLUSTER ANALYSIS IN PYTHON

Shaumik Daityari
Business Analyst

Measuring speed in hierarchical clustering

- `timeit` module
- Measure the speed of `.linkage()` method
- Use randomly generated points
- Run various iterations to extrapolate

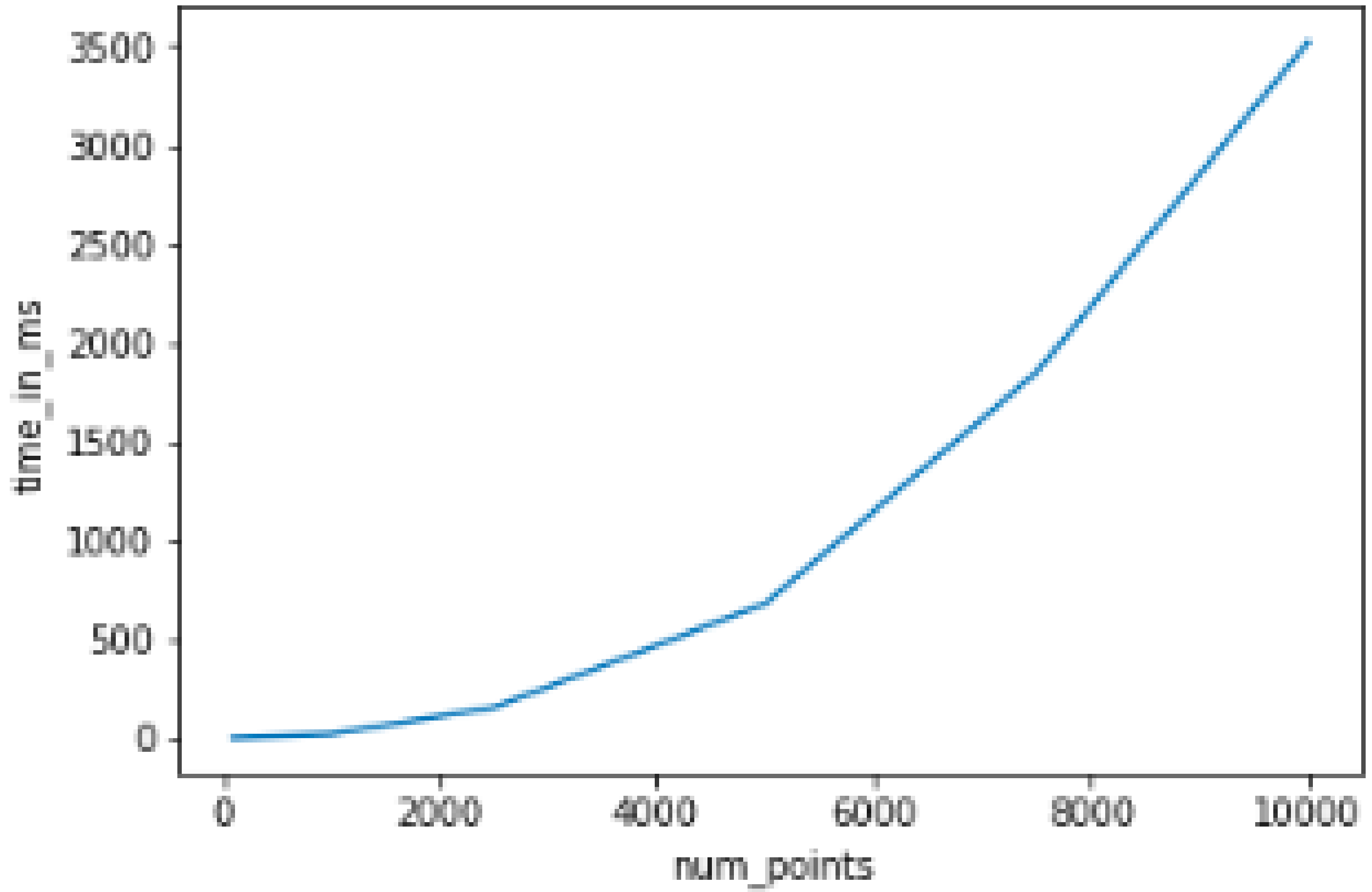
Use of timeit module

```
from scipy.cluster.hierarchy import linkage
import pandas as pd
import random, timeit
points = 100
df = pd.DataFrame({'x': random.sample(range(0, points), points),
                   'y': random.sample(range(0, points), points)})
%timeit linkage(df[['x', 'y']], method = 'ward', metric = 'euclidean')
```

```
1.02 ms ± 133 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Comparison of runtime of linkage method

- Increasing runtime with data points
- Quadratic increase of runtime
- Not feasible for large datasets



Next up - exercises

CLUSTER ANALYSIS IN PYTHON