Hierarchical Clustering

▼ Clustering

Hierarchical Clustering

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 9.8s Python

```python
df = pd.read_csv('cluster_mpg.csv')
```

✓ 0.6s Python

```python
df
```

✓ 0.6s Python

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino
...	...	...	...	...	...	...	...	...	...

Data & EDA

```python
df.describe()
```

0.2s Python

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327	75.979592
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864	3.683737
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000	73.000000
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

```python
df["origin"].value_counts()
```

0.2s Python

```
usa       245
japan      79
europe     68
Name: origin, dtype: int64
```

```python
df_w_dummies = pd.get_dummies(df.drop("name", axis=1))
```

0.1s Python

```python
1  df_w_dummies
```

✓ 0.2s Python

s	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan
8	307.0	130.0	3504	12.0	70	0	0
8	350.0	165.0	3693	11.5	70	0	0
8	318.0	150.0	3436	11.0	70	0	0
8	304.0	150.0	3433	12.0	70	0	0
8	302.0	140.0	3449	10.5	70	0	0
..	...	...	...	...	...	...	...
4	140.0	86.0	2790	15.6	82	0	0
4	97.0	52.0	2130	24.6	82	1	0
4	135.0	84.0	2295	11.6	82	0	0
4	120.0	79.0	2625	18.6	82	0	0
4	119.0	82.0	2720	19.4	82	0	0

s

```
[8]
from sklearn.preprocessing import MinMaxScaler
✓ 1.4s
```

```
[9]
scaler = MinMaxScaler()
✓ 0.2s
```

```
[10]
scaled_data = scaler.fit_transform(df_w_dummies)
✓ 0.1s
```

```
[11]
scaled_data
✓ 0.1s
```

```
array([[0.2393617 , 1.        , 0.61757106, ..., 0.        , 0.        ,
        1.        ],
       [0.15957447, 1.        , 0.72868217, ..., 0.        , 0.        ,
        1.        ],
       [0.2393617 , 1.        , 0.64599483, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.61170213, 0.2       , 0.17312661, ..., 0.        , 0.        ,
        1.        ],
       [0.50531915, 0.2       , 0.13436693, ..., 0.        , 0.        ,
        1.        ],
       [0.58510638, 0.2       , 0.13178295, ..., 0.        , 0.        ,
        1.        ]])
```

```python
scaled_df = pd.DataFrame(scaled_data, columns=df_w_dummies.columns)
```

✓ 0.8s Pytho

```python
scaled_df
```

✓ 0.1s Pytho

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	c
0	0.239362	1.0	0.617571	0.456522	0.536150	0.238095	0.0	
1	0.159574	1.0	0.728682	0.646739	0.589736	0.208333	0.0	
2	0.239362	1.0	0.645995	0.565217	0.516870	0.178571	0.0	
3	0.186170	1.0	0.609819	0.565217	0.516019	0.238095	0.0	
4	0.212766	1.0	0.604651	0.510870	0.520556	0.148810	0.0	
...	...	...	...	...	...	...	...	
387	0.478723	0.2	0.186047	0.217391	0.333711	0.452381	1.0	
388	0.930851	0.2	0.074935	0.032609	0.146583	0.988095	1.0	
389	0.611702	0.2	0.173127	0.206522	0.193365	0.214286	1.0	
390	0.505319	0.2	0.134367	0.179348	0.286929	0.630952	1.0	
391	0.585106	0.2	0.131783	0.195652	0.313864	0.678571	1.0	

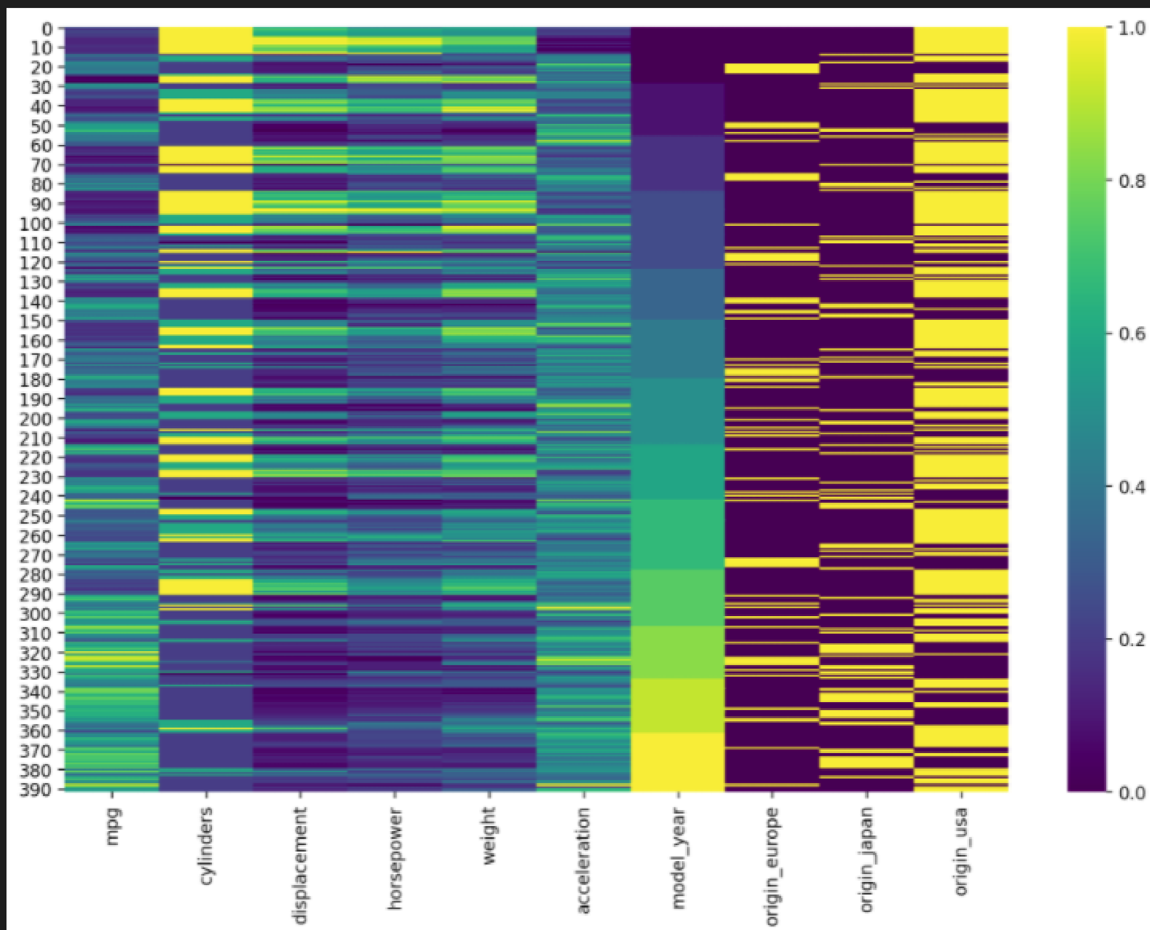
392 rows × 10 columns

```python
plt.figure(figsize=(12,8), dpi=200)
sns.heatmap(scaled_df, cmap="viridis")
```

✓ 2.4s

Python

```
<AxesSubplot:>
```

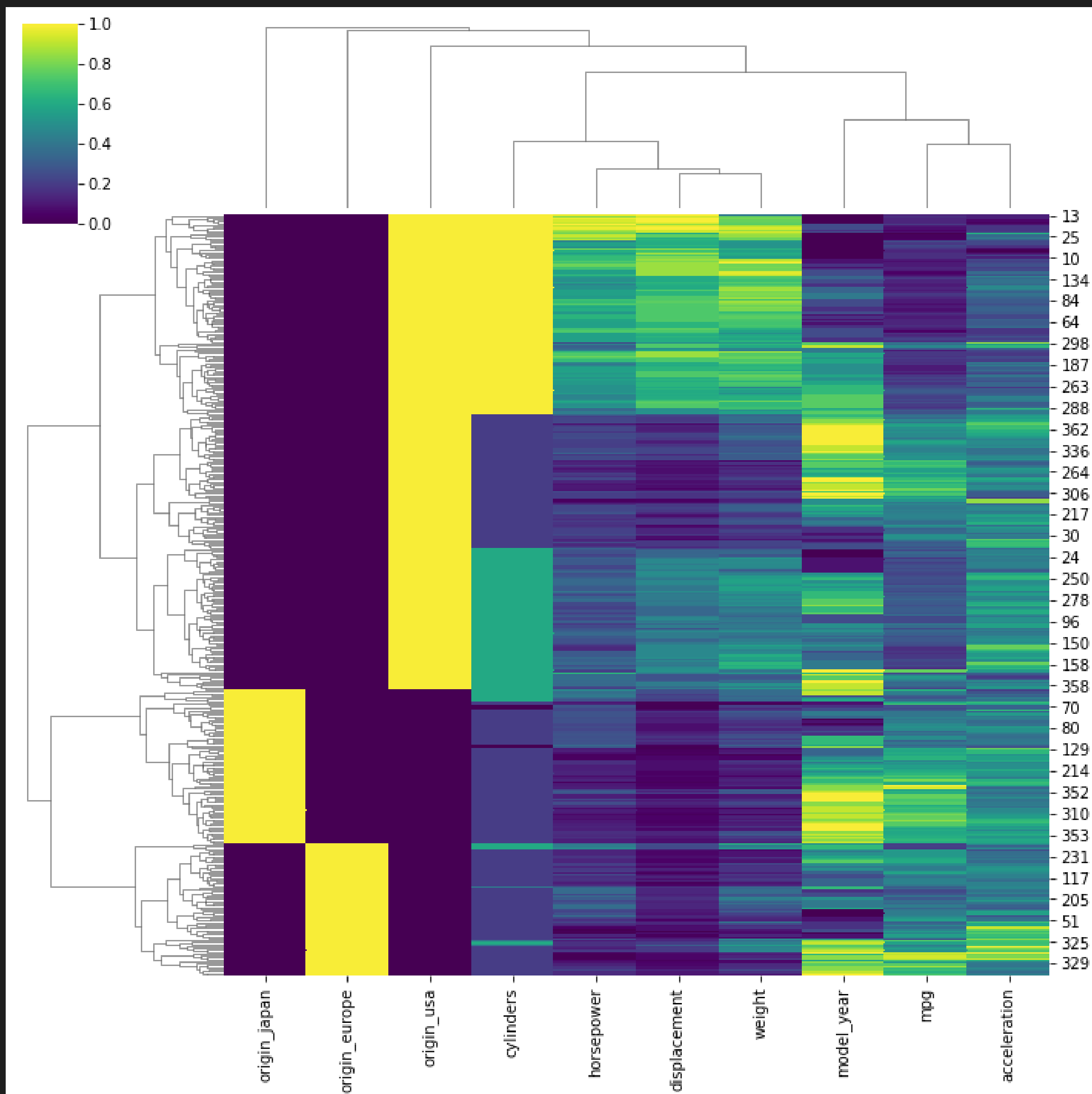


```
plt.figure(figsize=(12,12), dpi=200)
sns.clustermap(scaled_df, cmap="viridis")
```

✓ 1.7s Pyth

```
<seaborn.matrix.ClusterGrid at 0x207bc4f6970>

<Figure size 2400x2400 with 0 Axes>
```



```
1 scaled_df.corr()
```

✓ 0.1s

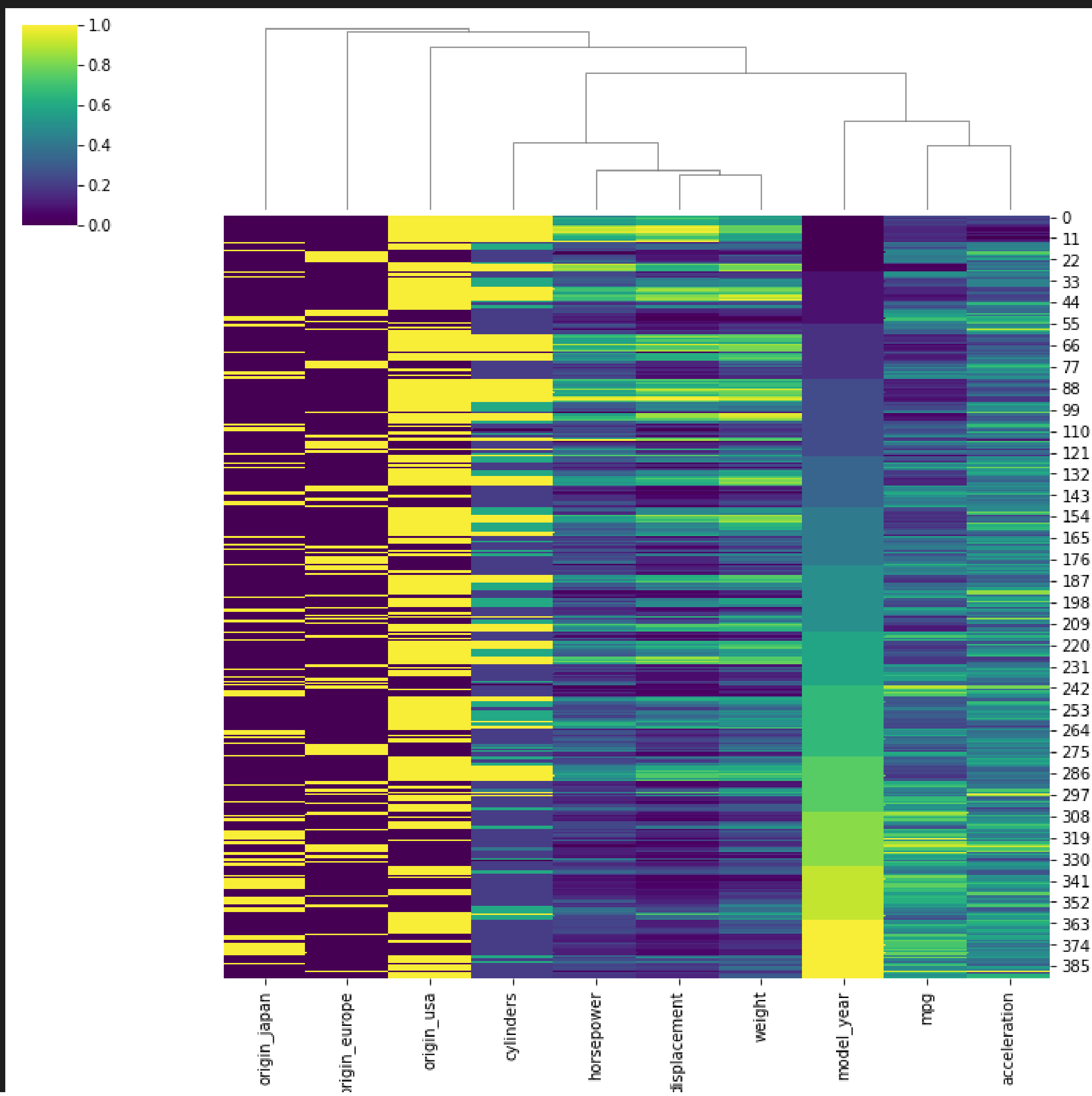
	mpg	cylinders	displacement	horsepower	weight	acceleration
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000
model_year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316
origin_europe	0.244313	-0.352324	-0.371633	-0.284948	-0.293841	0.208298
origin_japan	0.451454	-0.404209	-0.440825	-0.321936	-0.447929	0.115020
origin_usa	-0.565161	0.610494	0.655936	0.489625	0.600978	-0.258224

```python
plt.figure(figsize=(12,12), dpi=200)
sns.clustermap(scaled_df, cmap="viridis", row_cluster=False)
```

✓ 1.5s

```
<seaborn.matrix.ClusterGrid at 0x207ba5058e0>

<Figure size 2400x2400 with 0 Axes>
```

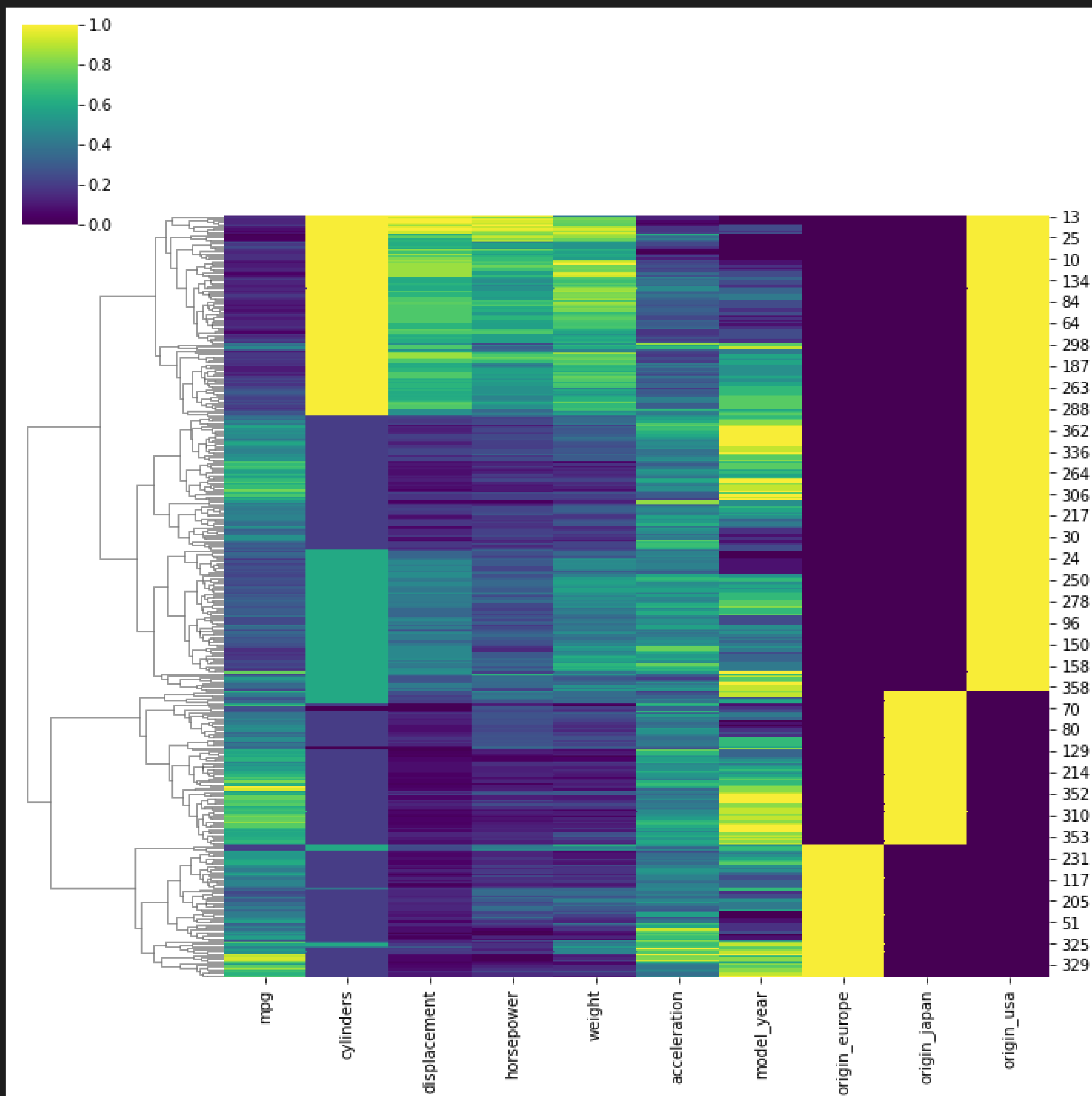


```python
plt.figure(figsize=(12,12), dpi=200)
sns.clustermap(scaled_df, cmap="viridis", col_cluster=False)
```

✓ 2.5s

```
<seaborn.matrix.ClusterGrid at 0x207b99d29d0>

<Figure size 2400x2400 with 0 Axes>
```



ML Model

```python
from sklearn.cluster import AgglomerativeClustering
```

✓ 0.4s

```python
model = AgglomerativeClustering(n_clusters=4)
```

✓ 0.7s

```python
cluster_labels = model.fit_predict(scaled_df)
```

✓ 0.1s

```python
cluster_labels
```

✓ 0.9s

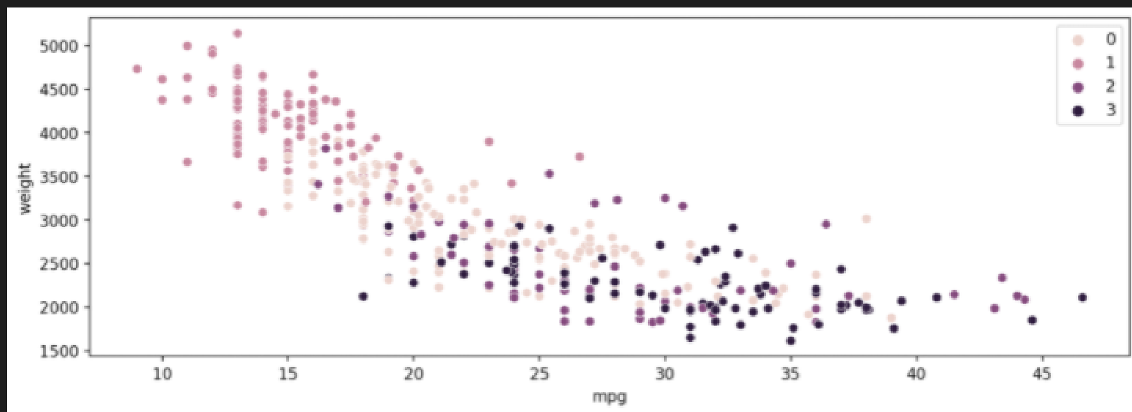
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 0, 0, 0, 3, 2, 2, 2,
       2, 2, 0, 1, 1, 1, 1, 3, 0, 3, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 2, 2, 2, 3, 3, 2, 0, 3, 0, 2, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 1, 1, 1, 1, 2, 2, 2, 2, 0, 3, 3, 0, 3, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 0, 3, 0, 3,
       3, 0, 0, 2, 1, 1, 2, 2, 2, 2, 1, 2, 3, 1, 0, 0, 0, 3, 0, 3, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 0, 2, 2, 3, 3, 2, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 3, 0, 0, 0, 3, 2, 3, 0, 2, 0, 2,
       2, 2, 2, 3, 2, 2, 0, 0, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 3, 0,
       0, 0, 0, 2, 3, 3, 0, 2, 1, 2, 3, 2, 1, 1, 1, 1, 3, 0, 2, 0, 3, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 2, 0, 3, 0, 0, 0, 3, 2, 3, 2, 3,
       2, 0, 3, 3, 3, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
```

```python
plt.figure(figsize=(12,4),dpi=200)
sns.scatterplot(data=df, x="mpg", y="weight", hue=cluster_labels)
```

✓ 0.9s

Python

```
<AxesSubplot:xlabel='mpg', ylabel='weight'>
```



Dendrograms

documentation
https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendro

```
model = AgglomerativeClustering(n_clusters=None, distance_threshold=0)
```

✓ 0.1s

```
cluster_labels = model.fit_predict(scaled_df)
```

✓ 0.1s

```
cluster_labels
```

✓ 0.1s

Output exceeds the size limit. Open the full output data in a text editor

```
array([247, 252, 360, 302, 326, 381, 384, 338, 300, 279, 217, 311, 377,
       281, 232, 334, 272, 375, 354, 333, 317, 345, 329, 289, 305, 383,
       290, 205, 355, 269, 202, 144, 245, 297, 386, 358, 199, 337, 330,
       339, 293, 352, 283, 196, 253, 168, 378, 331, 201, 268, 256, 361,
       250, 197, 246, 371, 324, 230, 203, 261, 380, 376, 308, 389, 332,
       306, 236, 391, 350, 274, 288, 313, 231, 298, 100, 295, 210, 248,
       187, 390, 373, 266, 307, 379, 212, 357, 191, 314, 208, 249, 343,
       294, 374, 322, 323, 362, 188, 296, 369, 286, 251, 229, 244, 285,
       349, 365, 259, 213, 276, 215, 222, 204, 359, 287, 166, 387, 291,
       220, 216, 260, 129, 367, 340, 346, 301, 342, 228, 388, 370, 218,
       255, 327, 347, 278, 271, 258, 282, 318, 273, 123, 172, 382, 363,
       356, 195, 280, 239, 364, 267, 351, 186, 257, 277, 299, 127, 366,
       234, 385, 192, 372, 292, 233, 270, 263, 133, 165, 161, 198,  97,
       315, 134, 207, 147, 175, 262, 348,  98, 214,  48, 353, 177, 325,
       128, 284, 275, 182, 184, 145, 344, 321, 200, 149, 240, 241, 235,
...
```

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

✓ 0.1s

```python
linkage_matrix = hierarchy.linkage(model.children_)
```

✓ 0.1s

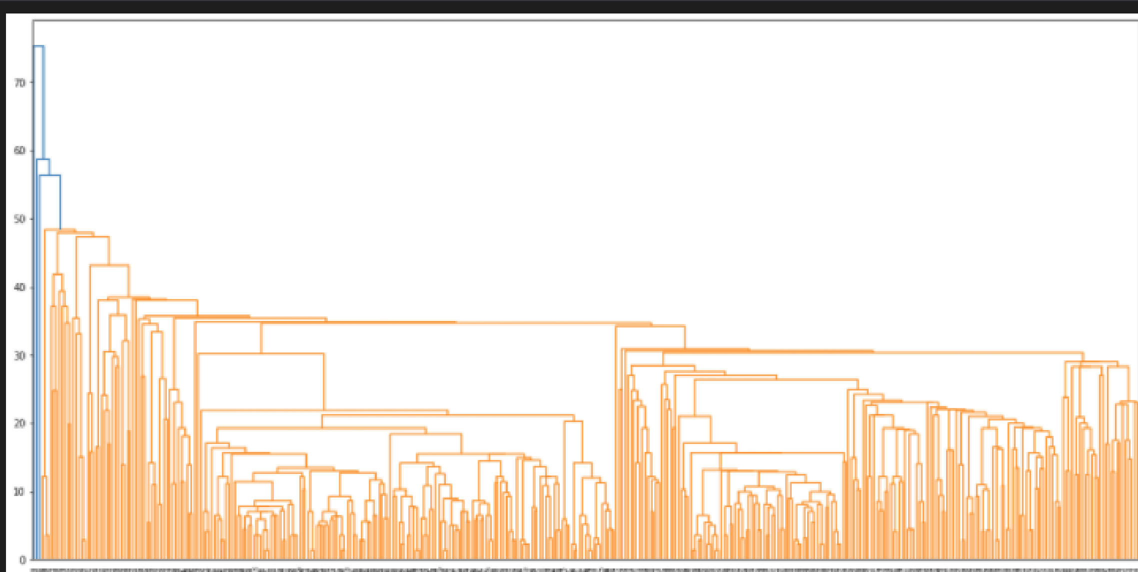
```python
linkage_matrix
```

✓ 0.1s

```
array([[ 67.        , 161.        ,   1.41421356,   2.        ],
       [ 10.        ,  45.        ,   1.41421356,   2.        ],
       [ 47.        ,  99.        ,   1.41421356,   2.        ],
       ...,
       [340.        , 777.        ,  56.40035461, 389.        ],
       [332.        , 778.        ,  58.69412236, 390.        ],
       [349.        , 779.        ,  75.32595834, 391.        ]])
```

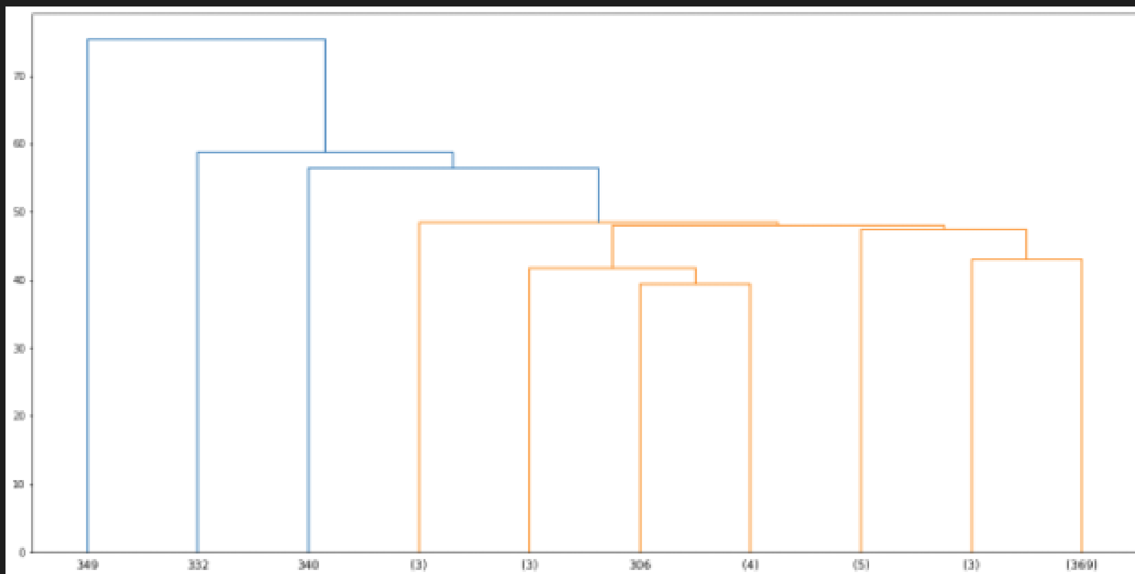
```python
plt.figure(figsize=(20,10))
dendro = dendrogram(linkage_matrix)
```

✓ 32.2s



```python
plt.figure(figsize=(20,10))
dendro = dendrogram(linkage_matrix, truncate_mode="lastp", p=10)
```

✓ 0.3s Python



Treshold Choosing

```
1  scaled_df.describe()
```
✓ 0.1s

	mpg	cylinders	displacement	horsepower	weight
count	392.000000	392.000000	392.000000	392.000000	392.000000
mean	0.384200	0.494388	0.326646	0.317768	0.386897
std	0.207580	0.341157	0.270398	0.209191	0.240829
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.212766	0.200000	0.095607	0.157609	0.173589
50%	0.365691	0.200000	0.214470	0.258152	0.337539
75%	0.531915	1.000000	0.536822	0.434783	0.567550
max	1.000000	1.000000	1.000000	1.000000	1.000000

```
1  scaled_df["mpg"].idxmax()
```
✓ 0.7s

```
320
```

```
1  scaled_df["mpg"].idxmin()
```
✓ 0.1s

```
28
```

```python
# https://stackoverflow.com/questions/1401712/how-can-th

a = scaled_df.iloc[320]
b = scaled_df.iloc[28]

```

✓ 0.1s

```python
np.sqrt(len(scaled_df.columns))
```

✓ 0.1s

```
3.1622776601683795
```

```python
distance = np.linalg.norm(a-b)
```

✓ 0.1s

```python
distance
```

✓ 0.8s

```
2.3852929970374714
```

```python
model = AgglomerativeClustering(n_clusters=None,
    distance_threshold=2)
```
✓ 0.1s Python

```python
cluster_labels = model.fit_predict(scaled_data)
```
✓ 0.1s Python

```python
cluster_labels
```
✓ 0.1s Python

```
Output exceeds the size limit. Open the full output data in a text editor
array([ 3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  1,  4,
4,
        4,  1,  0,  0,  0,  0,  0,  4,  3,  3,  3,  3,  1,  7,  1,  4,
4,
        4,  4,  4,  3,  3,  3,  3,  3,  3,  3,  4,  7,  4,  4,  7,  0,
0,
        0,  1,  1,  0,  7,  1,  7,  0,  7,  7,  3,  3,  3,  3,  3,  3,
3,
        3,  3,  1,  3,  3,  3,  3,  0,  0,  0,  0,  7,  1,  1,  7,  1,
3,
        3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  3,  4,  4,  4,  4,  4,
0,
        3,  3,  3,  3,  4,  1,  7,  1,  1,  7,  4,  0,  3,  3,  0,  0,
0,
```

```python
np.unique(cluster_labels)
```
✓ 0.8s

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10], dtype=int64)
```

Linkage Matrix

Source:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.

```
A (n-1) by 4 matrix Z is returned. At the i-th iteration,
clusters with indices Z[i, 0] and Z[i, 1] are combined to form
cluster n + i. A cluster with an index less than n corresponds
to one of the original observations. The distance between
clusters Z[i, 0] and Z[i, 1] is given by Z[i, 2]. The fourth
value Z[i, 3] represents the number of original observations
in the newly formed cluster.
```

```python
linkage_matrix = hierarchy.linkage(model.children_)
```

✓ 0.9s Python

```python
linkage_matrix
```

✓ 0.9s Python

```
array([[ 67.        , 161.        ,   1.41421356,   2.        ],
       [ 10.        ,  45.        ,   1.41421356,   2.        ],
       [ 47.        ,  99.        ,   1.41421356,   2.        ],
       ...,
       [340.        , 777.        ,  56.40035461, 389.        ],
       [332.        , 778.        ,  58.69412236, 390.        ],
       [349.        , 779.        ,  75.32595834, 391.        ]])
```

```python
plt.figure(figsize=(20,10))
dn = hierarchy.dendrogram(linkage_matrix,truncate_mode='lastp',p=11)
```