Hierarchal Clustering

imports

In [1]:

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

The Data

In [2]:

```
df = pd.read_csv("D:\\Study\\Programming\\python\\Python course from udemy\\Udemy - 2022 Python for Machine Learning & Data Science Masterclass\\23 - Hierarchical Clustering\\330 28506-cluster-mpg.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	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

In [4]:

```
df = df.dropna()
```

In [5]:

```
df.head()
```

Out[5]:

	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

In [6]:

```
df.describe()
```

Out[6]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
	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

In [7]:

```
df['origin'].value_counts()
```

Out[7]:

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

In [8]:

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

In [9]:

```
df_w_dummies
```

Out[9]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan	origin_usa
0	18.0	8	307.0	130.0	3504	12.0	70	0	0	1
1	15.0	8	350.0	165.0	3693	11.5	70	0	0	1
2	18.0	8	318.0	150.0	3436	11.0	70	0	0	1
3	16.0	8	304.0	150.0	3433	12.0	70	0	0	1
4	17.0	8	302.0	140.0	3449	10.5	70	0	0	1
...	...	...	...	...	...	...	...	...	...	...
387	27.0	4	140.0	86.0	2790	15.6	82	0	0	1
388	44.0	4	97.0	52.0	2130	24.6	82	1	0	0
389	32.0	4	135.0	84.0	2295	11.6	82	0	0	1
390	28.0	4	120.0	79.0	2625	18.6	82	0	0	1
391	31.0	4	119.0	82.0	2720	19.4	82	0	0	1

392 rows × 10 columns

In [10]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [11]:

```
scaler = MinMaxScaler()
```

In [12]:

```
scaler_data = scaler.fit_transform(df_w_dummies)
```

In [13]:

```
scaler_data
```

Out[13]:

```
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.        ]])
```

In [14]:

```
scaler_df = pd.DataFrame(scaler_data,columns=df_w_dummies.columns)
```

In [15]:

```
scaler_df
```

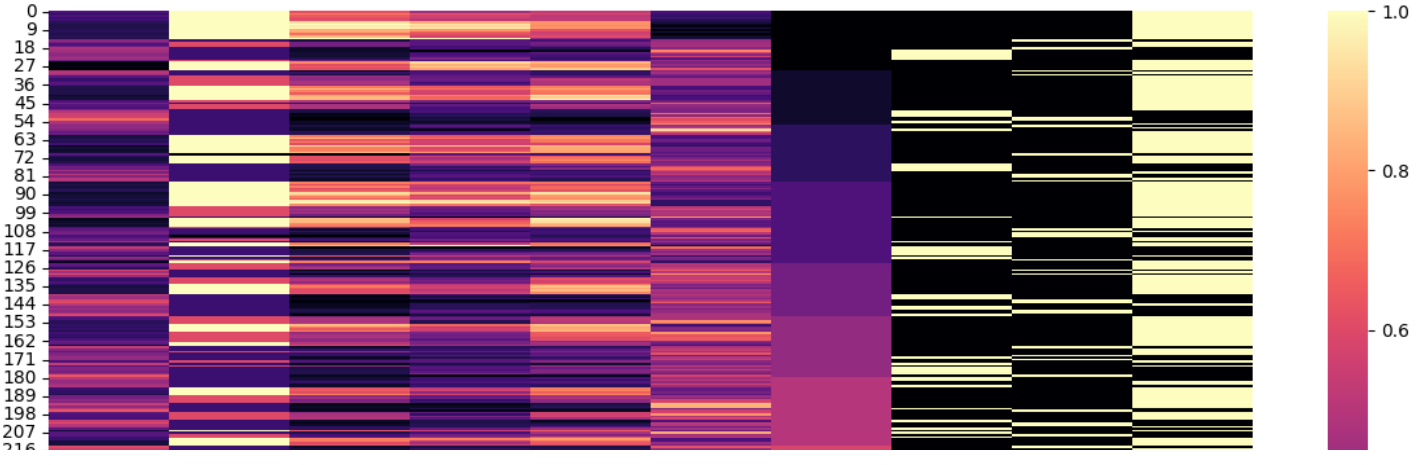
Out[15]:

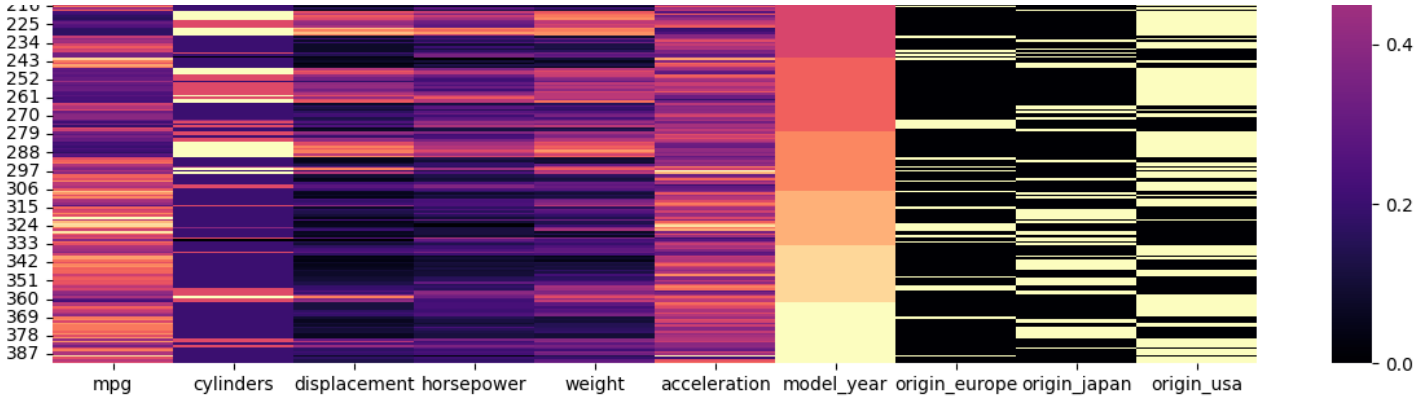
	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan	origin_us
0	0.239362	1.0	0.617571	0.456522	0.536150	0.238095	0.0	0.0	0.0	1
1	0.159574	1.0	0.728682	0.646739	0.589736	0.208333	0.0	0.0	0.0	1
2	0.239362	1.0	0.645995	0.565217	0.516870	0.178571	0.0	0.0	0.0	1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

392 rows × 10 columns

In [16]:

```
plt.figure(figsize=(15,8))
sns.heatmap(scaler_df,cmap='magma');
```



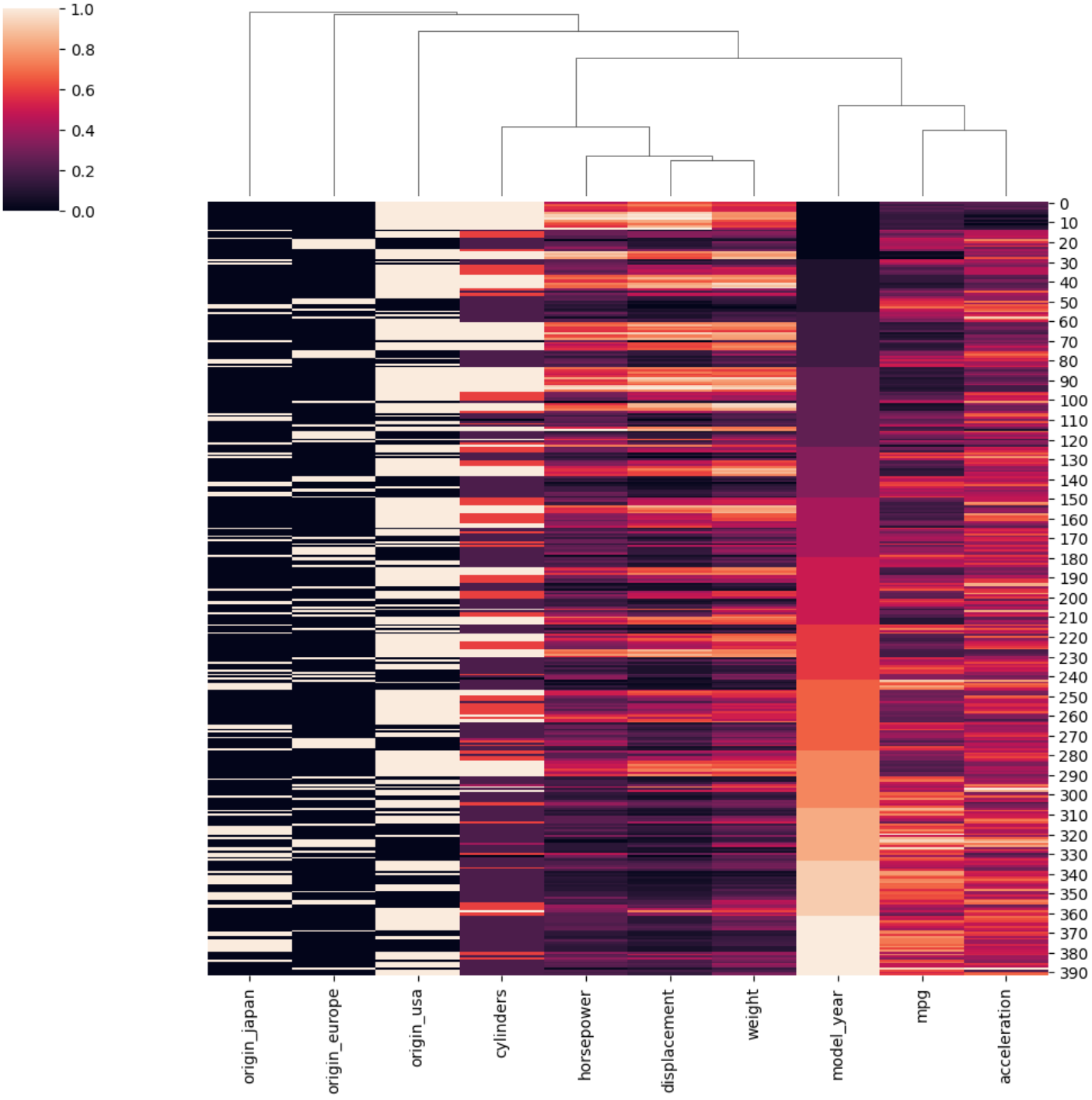


In [17]:

```
sns.clustermap(scaler_df,row_cluster=False)
```

Out[17]:

```
<seaborn.matrix.ClusterGrid at 0x1f46de5c520>
```

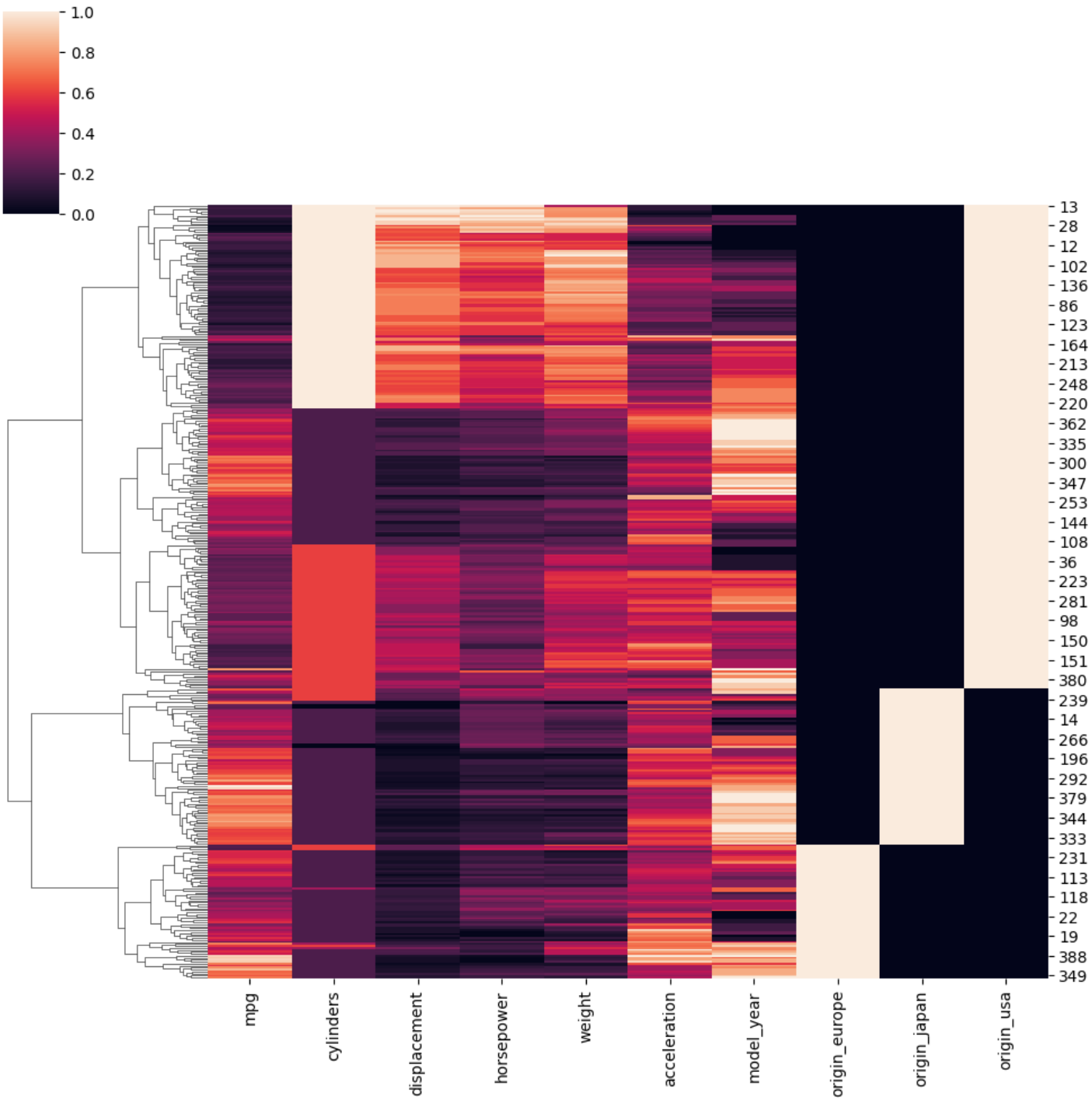


In [18]:

```
sns.clustermap(scaler_df,col_cluster=False)
```

Out[18]:

```
<seaborn.matrix.ClusterGrid at 0x1f46d520d30>
```



Using Scikit-Learn

In [19]:

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

In [20]:

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

In [21]:

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

In [22]:

```
cluster_labels
```

Out[22]:

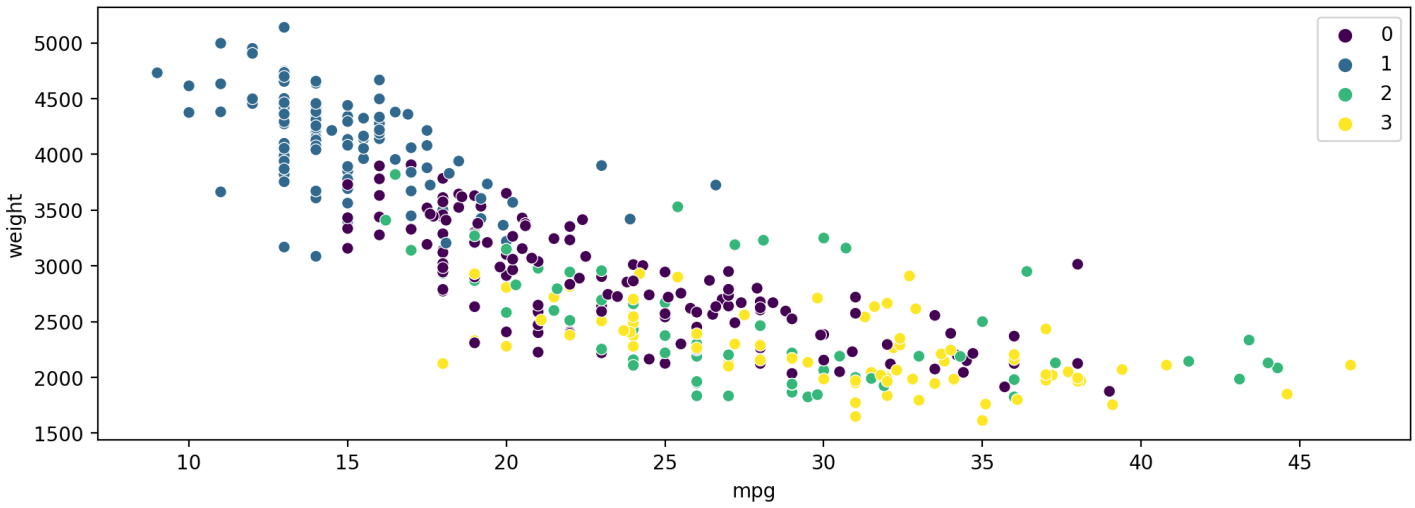
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 0, 0, 0, 3, 2, 2, 2,
       2, 2, 0, 1, 1, 1, 1, 3, 0, 3, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 2, 2, 2, 3, 3, 2, 0, 3, 0, 2, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 1, 1, 1, 1, 2, 2, 2, 2, 0, 3, 3, 0, 3, 1, 1, 1, 1,
       1, 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,
       0, 3, 3, 0, 3, 0, 0, 3, 2, 2, 2, 2, 2, 3, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 3, 0, 0, 2, 1, 2, 1, 0, 0, 3, 2, 0, 0, 0, 0, 2,
       3, 0, 3, 0, 0, 0, 0, 2, 3, 3, 3, 3, 3, 0, 3, 2, 2, 2, 2, 3, 3, 2,
       3, 3, 2, 3, 0, 0, 0, 0, 0, 3, 0, 3, 3, 3, 3, 3, 0, 0, 0, 2, 3, 3,
       3, 3, 2, 2, 3, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 3, 0, 0,
       3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0], dtype=int64)
```

In [23]:

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

Out[23]:

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



Exploring Number of Clusters with Dendrograms

Make sure to read the documentation online!
https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.dendrogram.html

Assuming every point starts as its own cluster

In [24]:

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

In [25]:

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

In [26]:

```
cluster_labels
```

Out[26]:

```
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,
       226, 160, 341, 193, 320, 101, 224, 162, 243, 146,  99, 185, 119,
       219, 209, 265, 221, 335,  66, 121, 316, 319, 254, 264, 124, 336,
       304, 206, 106, 148, 368, 122, 164, 131, 142,  95, 173, 194, 152,
       138, 157, 110, 159, 107, 312, 328, 225, 150, 211, 140, 163, 242,
       116,  81,  93,  96,  72, 189, 303, 167,  73, 115, 143, 132, 181,
       141, 103, 170, 130,  49,  83, 309, 120,  82, 227, 310, 151, 117,
       104, 109,  57,  75,  79, 169,  71,  84, 153,  35,  47, 238, 180,
        74, 237, 176, 190, 139, 125, 135, 156, 108, 171, 136,  53,  23,
        67,  94, 113, 112,  41,  70, 174,  61, 102,  40,  64,  65,  60,
       118, 223, 137,  63,  86, 155, 178,  36,  31,  88,  87,  58,  54,
       114, 111, 158,  78,  92,  50,  26,  17,  85, 183,  80,  42,  69,
        32, 154,  51,  20,  76,  34, 179,  68,  39,  59,  33,  56, 126,
        19,  15,  37,  89,  62,  77,  29,  38, 105,  52,  28,  90,  46,
        55,  43,   9,  91,  18,  16,  25,   7,  45,  27,  44,   8,  30,
        22,  24,  21,  10,   4,  14,  13,  12,  11,   5,   6,   2,   3,
         1,   0], dtype=int64)
```

In [27]:

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

Linkage Model

In [28]:

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

In [29]:

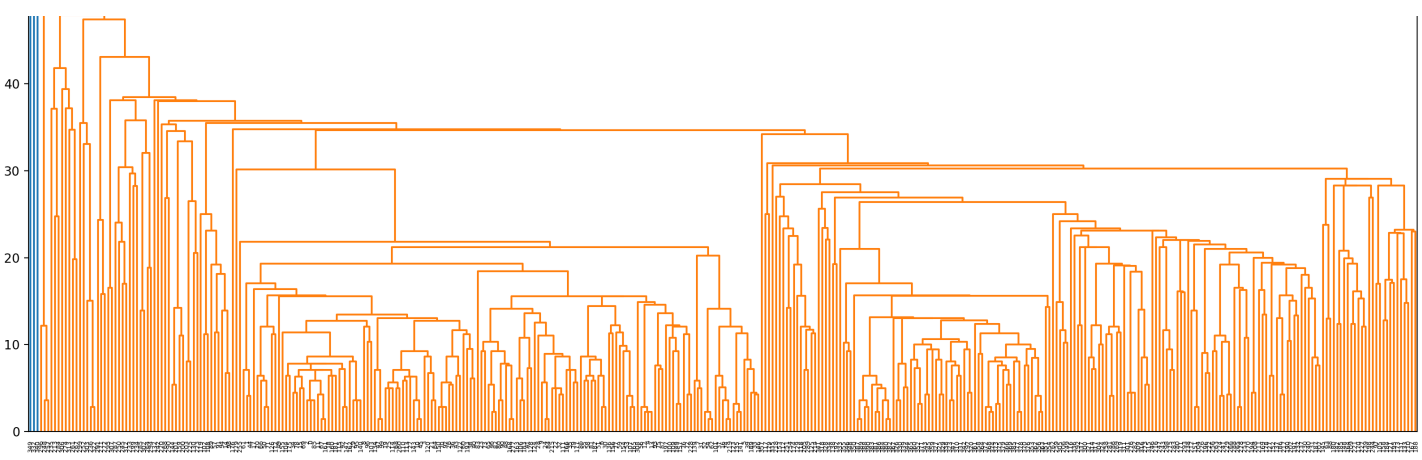
```
linkage_matrix
```

Out[29]:

```
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.        ]])
```

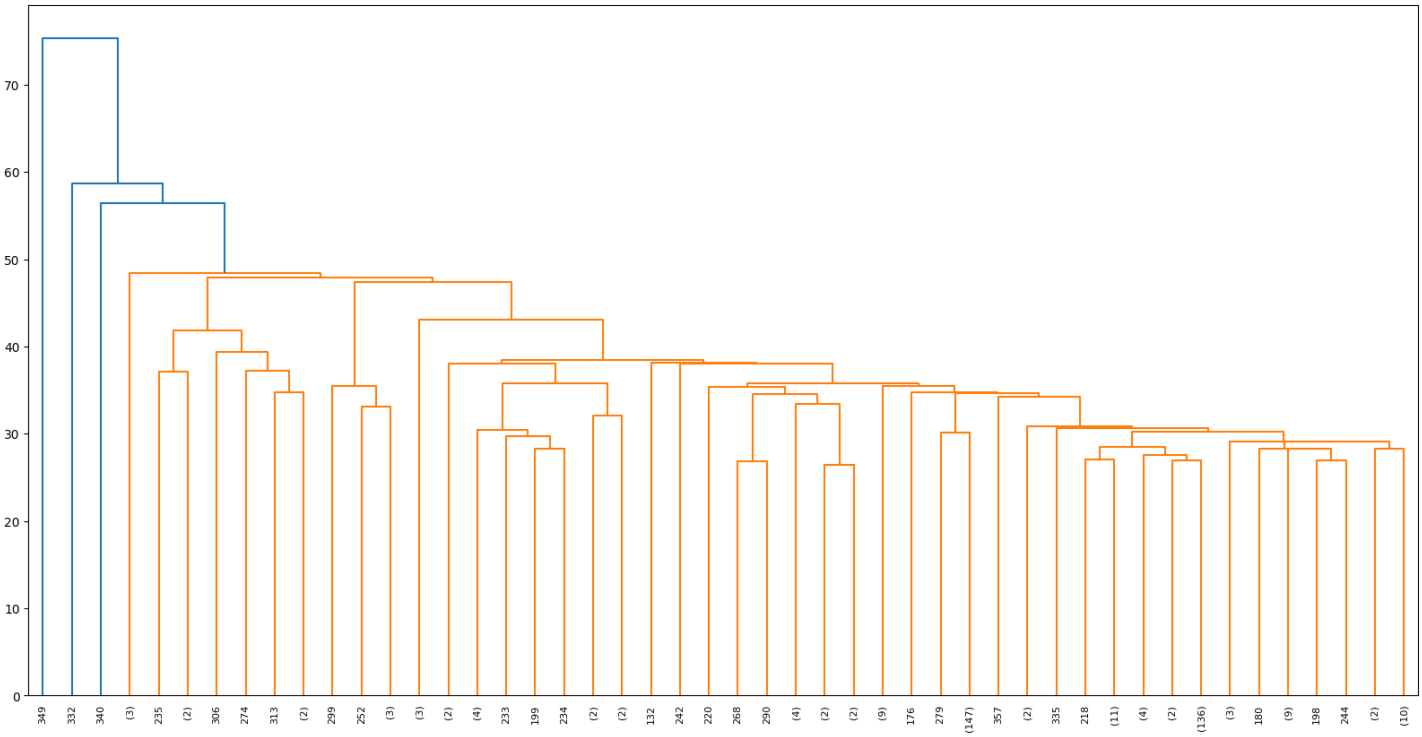
In [45]:

```
plt.figure(figsize=(20,10),dpi=200)
# Warning! This plot will take awhile!!
dn = dendrogram(linkage_matrix,)
```



In [31]:

```
plt.figure(figsize=(20,10))
dn = hierarchy.dendrogram(linkage_matrix,truncate_mode='lastp',p=48)
```



Choosing a Threshold Distance

What is the distance between two points?

In [32]:

```
scaler_df.describe()
```

Out[32]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	0.384200	0.494388	0.326646	0.317768	0.386897	0.448888	0.498299	0.173469	0.201531
std	0.207580	0.341157	0.270398	0.209191	0.240829	0.164218	0.306978	0.379136	0.401656
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.212766	0.200000	0.095607	0.157609	0.173589	0.343750	0.250000	0.000000	0.000000
50%	0.365691	0.200000	0.214470	0.258152	0.337539	0.446429	0.500000	0.000000	0.000000
75%	0.531915	1.000000	0.536822	0.434783	0.567550	0.537202	0.750000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [33]:

```
scaler_df['mpg'].idxmax()
```

Out[33]:

```
320
```

In [34]:

```
scaler_df['mpg'].idxmin()
```

Out[34]:

```
28
```

In [35]:

```
# https://stackoverflow.com/questions/1401712/how-can-the-euclidean-distance-be-calculated-with-numpy
a = scaler_df.iloc[320]
b = scaler_df.iloc[28]
dist = np.linalg.norm(a-b)
```

In [36]:

```
dist
```

Out[36]:

```
2.3852929970374714
```

Max possible distance?

Recall Euclidean distance: https://en.wikipedia.org/wiki/Euclidean_distance

In [37]:

```
np.sqrt(len(scaler_df.columns))
```

Out[37]:

```
3.1622776601683795
```

Creating a Model Based on Distance Threshold

- **distance_threshold**
  - **The linkage distance threshold above which, clusters will not be merged.**

In [38]:

```
model = AgglomerativeClustering(n_clusters=None,distance_threshold=2)
```

In [39]:

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

In [40]:

```
cluster_labels
```

Out[40]:

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

In [41]:

```
np.unique(cluster_labels)
```

Out[41]:

```
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.linkage.html#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.

In [42]:

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

In [43]:

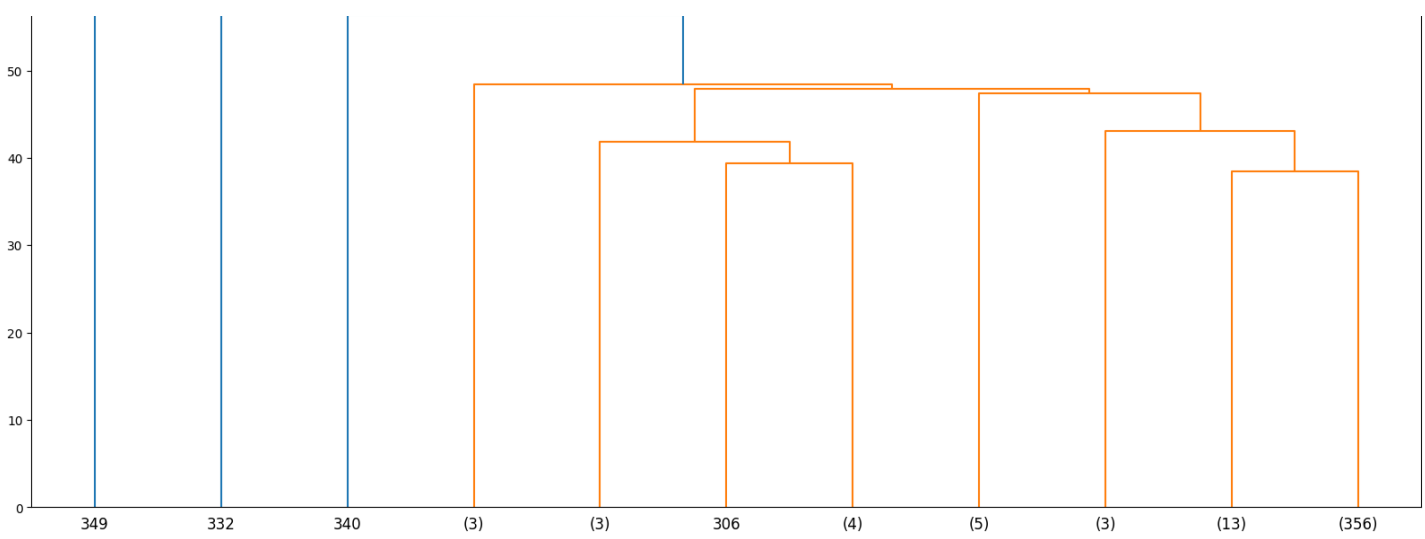
```
linkage_matrix
```

Out[43]:

```
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.        ]])
```

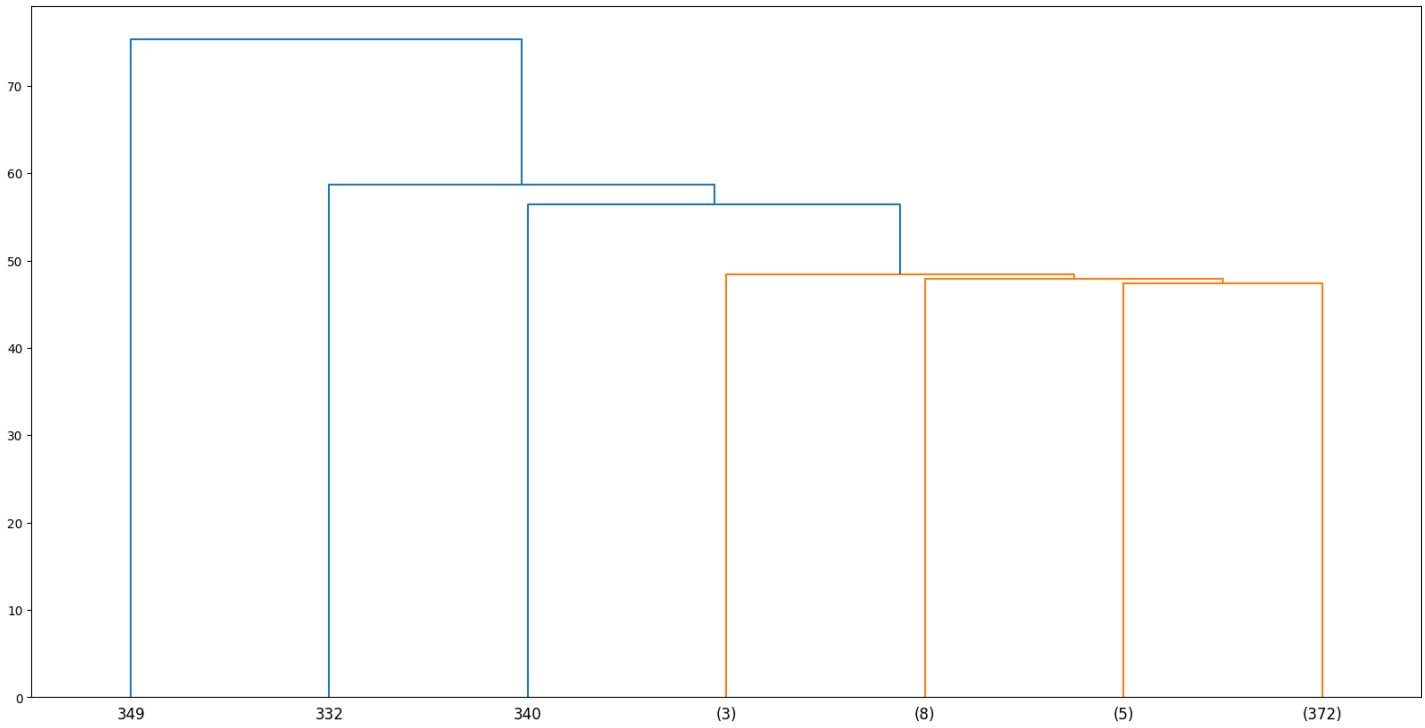
In [44]:

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



In [47]:

```
plt.figure(figsize=(20,10))
dn = hierarchy.dendrogram(linkage_matrix,truncate_mode='lastp',p=7)
```



In [ ]:

```

```