Praktikum 4 - Hierachical Clustering

Fayza Aulia

24060120120010

Tugas Ke-4

Praktikum Machine Learning A2

Import library

In [ ]:
```
import numpy as np
import pandas as pd
from scipy import ndimage
from scipy.cluster import hierarchy
from scipy.spatial import distance_matrix
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets import make_blobs
%matplotlib inline
```

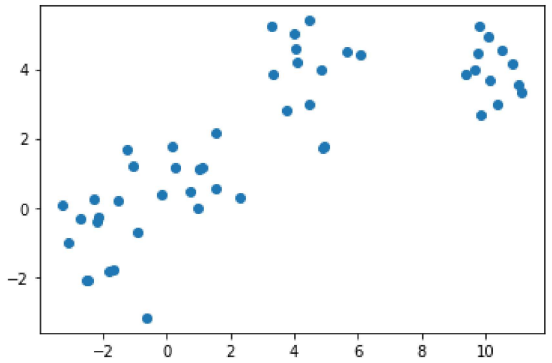
1 - Random Dataset

Generate Random Dataset

In [ ]:
```
# Generate random datasets
X1, y1 = make_blobs(n_samples=50, centers=[[4, 4], [-2, -1], [1, 1], [10, 4]], cluster_std=0.9)
```

In [ ]:
```
plt.scatter(X1[:, 0], X1[:, 1], marker='o')
```

Out[900]:
```
<matplotlib.collections.PathCollection at 0x7fb74d438490>
```



Agglomerative Clustering

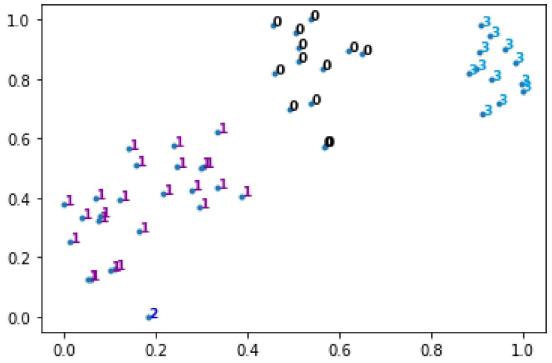
1. Agglomerative Clustering "Single"

In [ ]:
```
agglom_single = AgglomerativeClustering(n_clusters = 4, linkage = 'single')
```

In [ ]:
```
agglom_single.fit(X1,y1)
```

Out[902]:
```
AgglomerativeClustering(linkage='single', n_clusters=4)
```

In [ ]:
```
# Plot the scatter plot of the randomly generated data
plt.figure(figsize=(6,4))

# Terdapat 2 baris kode yang digunakan untuk skala dari data poin yang menurun
# Atau sebaliknya dari data poins yang akan di scatter setiap bagiannya

#Membuat range dari X1 secara minimum dan maximum
x_min, x_max = np.min(X1, axis=0), np.max(X1, axis=0)

# Mendapatkan rata-rata jarak untuk X1
X1 = (X1 - x_min) / (x_max - x_min)
for i in range(X1.shape[0]):
    plt.text(X1[i, 0], X1[i, 1], str(agglom_single.labels_[i]),
             color=plt.cm.nipy_spectral(agglom_single.labels_[i] / 10.),
             fontdict={'weight': 'bold', 'size': 9})
#Memindahkan x tcks, y ticks, serta axis dari x dan y
# plt.xticks([])
# plt.yticks([])

# Melakukan display plot dari data asli sebelum dilakukan kluster
plt.scatter(X1[:, 0], X1[:, 1], marker='.')
#Melakukan display plot
plt.show()
```



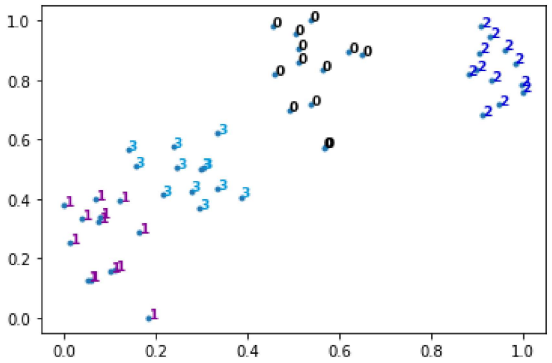
2. Agglomerative Clustering "Average"

In [ ]:
```
agglom_average = AgglomerativeClustering(n_clusters = 4, linkage = 'average')
```

In [ ]:
```
agglom_average.fit(X1,y1)
```

Out[905]:
```
AgglomerativeClustering(linkage='average', n_clusters=4)
```

In [ ]:
```
# Plot the scatter plot of the randomly generated data
plt.figure(figsize=(6,4))

# Terdapat 2 baris kode yang digunakan untuk skala dari data poin yang menurun
# Atau sebaliknya dari data poins yang akan di scatter setiap bagiannya

#Membuat range dari X1 secara minimum dan maximum
x_min, x_max = np.min(X1, axis=0), np.max(X1, axis=0)

# Mendapatkan rata-rata jarak untuk X1
X1 = (X1 - x_min) / (x_max - x_min)
for i in range(X1.shape[0]):
    plt.text(X1[i, 0], X1[i, 1], str(agglom_average.labels_[i]),
             color=plt.cm.nipy_spectral(agglom_average.labels_[i] / 10.),
             fontdict={'weight': 'bold', 'size': 9})
# #Memindahkan x tcks, y ticks, serta axis dari x dan y
# plt.xticks([])
# plt.yticks([])

# Melakukan display plot dari data asli sebelum dilakukan kluster
plt.scatter(X1[:, 0], X1[:, 1], marker='.')
#Melakukan display plot
plt.show()
```



In [ ]:
```
dist_matrix = distance_matrix(X1, X1)
print(dist_matrix)
```

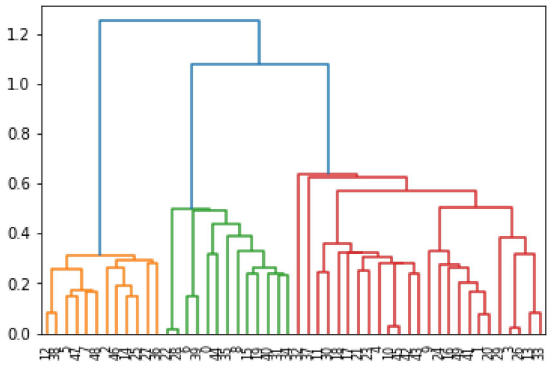
```
[[0.         0.5482367  0.4586803  ... 0.50241903 0.51176241 0.58654121]
 [0.5482367  0.         0.99649846 ... 1.04457775 1.04676926 0.0893565 ]
 [0.4586803  0.99649846 0.         ... 0.0562437  0.05471632 1.04245595]
 ...
 [0.50241903 1.04457775 0.0562437  ... 0.         0.05207259 1.08824023]
 [0.51176241 1.04676926 0.05471632 ... 0.05207259 0.         1.09416801]
 [0.58654121 0.0893565  1.04245595 ... 1.08824023 1.09416801 0.        ]]
```

In [ ]:
```
# Hierarchical Clustering
a = hierarchy.linkage(dist_matrix, 'single')
b = hierarchy.linkage(dist_matrix, 'average')
```

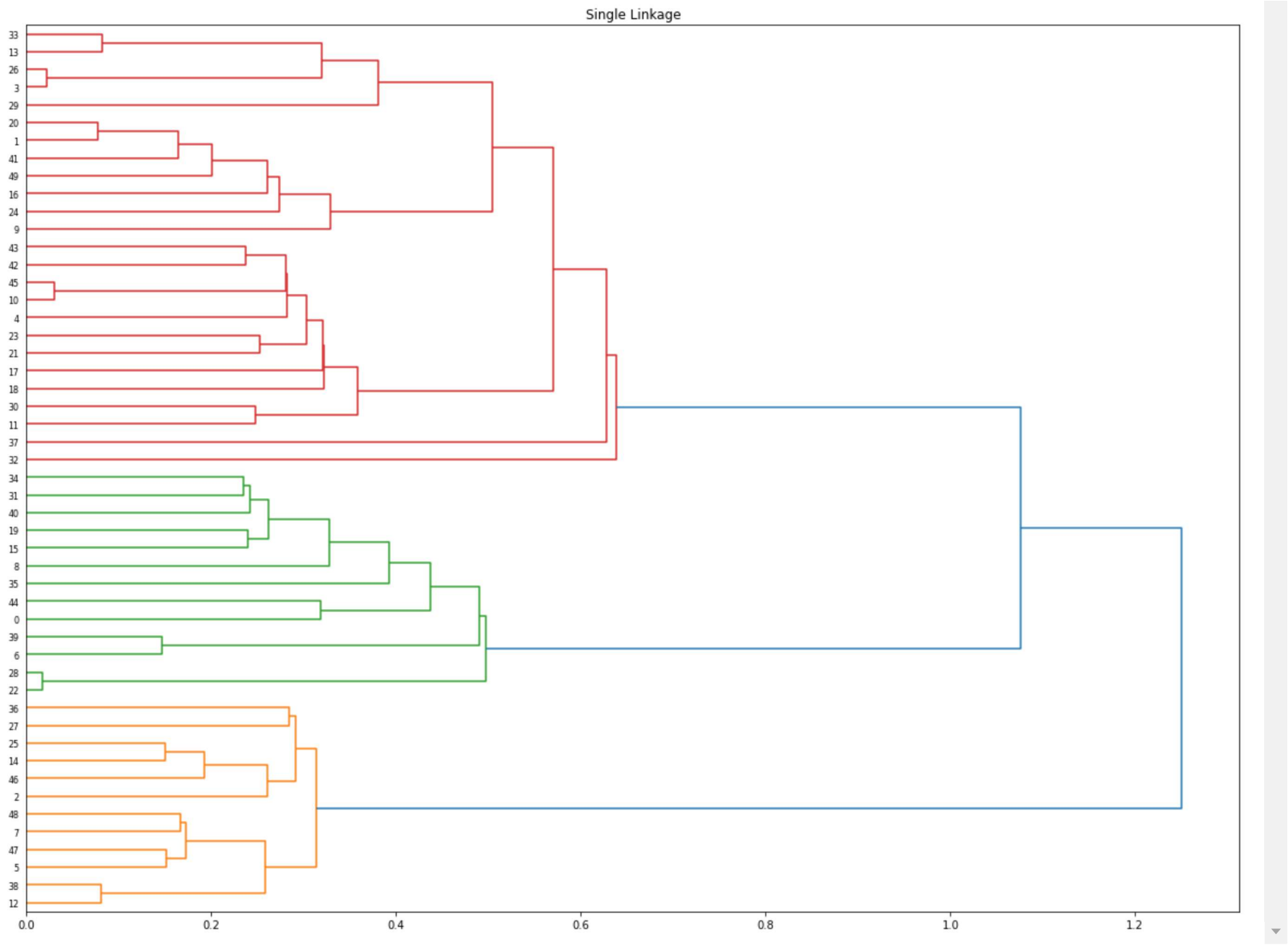
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix looks
suspiciously like an uncondensed distance matrix

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix looks
suspiciously like an uncondensed distance matrix
  This is separate from the ipykernel package so we can avoid doing imports until
```

Plot Dendogram

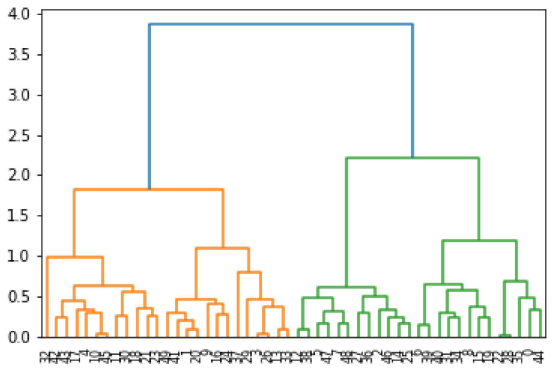
In [ ]:
```
# Dendogram untuk Prameter a
dendro = hierarchy.dendrogram(a)
```



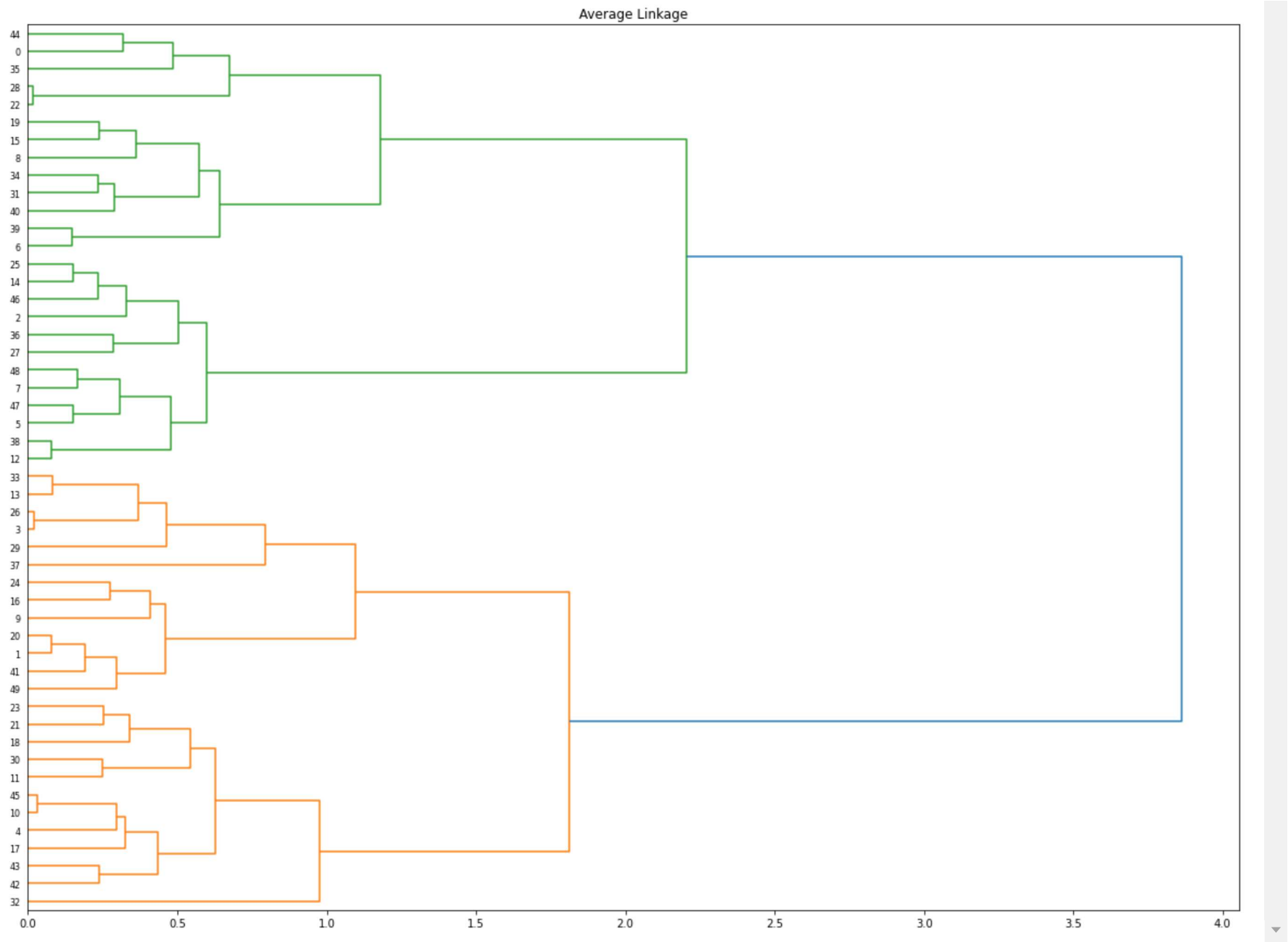
In [ ]:
```
# Single Linkage
plt.figure(figsize=(20,15))
plt.title('Single Linkage')
Z1 = hierarchy.dendrogram(a, orientation='right')

plt.show()
```


Single Linkage
33
13
26
3
29
20
1
41
49
16
24
9
43
42
45
10
4
23
21
17
18
30
11
37
32
34
31
40
19
15
8
35
44
0
39
6
28
22
36
27
25
14
46
2
48
7
47
5
38
12
0.0
0.2
0.4
0.6
0.8
1.0
1.2

In [ ]:
```
# Dendogram untuk Prameter b
dendro = hierarchy.dendrogram(b)
```



In [ ]:
```
# Average Linkage
plt.figure(figsize=(20,15))
Z2 = hierarchy.dendrogram(b, orientation='right')
plt.title('Average Linkage')

plt.show()
```



Tugas 1

Perbedaan clustering agglomerative linkage single dengan linkage average

Berdasarkan clustering menggunakan agglomerative clustering untuk dataset random yang tersedia dengan single linkage dan average linkage ditemukan beberapa perbedaan berikut.

1. Single linkage menggabungkan cluster-cluster menurut jarak antara anggota-anggota terdekat di antara dua cluster. Sedangkan average linkage menggabungkan cluster-cluster menurut jarak rata-rata pasangan anggota masing-masing pada himpunan antara dua cluster.
2. Jadi single linkage perhitungan berdasarkan jarak tiap anggota apabila average menghitung jaraknya tiap pasangan. Pada single linkage pengelompokan berdasarkan jarak terdekat sedangkan average berdasarkan jarak rata-rata.
3. Misalkan kita menginginkan jumlah cluster sebanyak 4 cluster untuk masing-masing linkage maka dapat dilihat pada dendrogram bahwa hasil cluster yang menggunakan linkage single dan linkage average berbeda.

2 - Cars Dataset

Understanding the data

In [ ]:
```
# Import dataset
pdf = pd.read_csv('cars_clus.csv')
print("Shape of dataset: ", pdf.shape)
pdf.head()
```

```
Shape of dataset:  (159, 16)
```

Out[913]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	width	length	curb_wgt	fuel_cap	mpg	lnsales	partition
0	Acura	Integra	16.919	16.36	0	21.5	1.8	140	101.2	67.3	172.4	2.639	13.2	28	2.828	0
1	Acura	TL	39.384	19.875	0	28.4	3.2	225	108.1	70.3	192.9	3.517	17.2	25	3.673	0
2	Acura	CL	14.114	18.225	0	*null*	3.2	225	106.9	70.6	192	3.47	17.2	26	2.647	0
3	Acura	RL	8.588	29.725	0	42	3.5	210	114.6	71.4	196.6	3.85	18	22	2.15	0
4	Audi	A4	20.397	22.255	0	23.99	1.8	150	102.6	68.2	178	2.998	16.4	27	3.015	0

Preprocessing Data

In [ ]:
```
# Data Cleaning
# Menghapus dataset dengan membuang baris yang memiliki nilai null
pdf[['sales','resale','type','price','engine_s','horsepow','wheelbas','width','length','curb_wgt','fuel_cap','mpg','lnsales']] = pdf[['sales','resale','ty
pdf = pdf.dropna()
pdf = pdf.reset_index(drop=True)
print("Shape of dataset after cleaning: ", pdf.shape)
pdf.head()
```

```
Shape of dataset after cleaning:  (117, 16)
```

Out[914]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	width	length	curb_wgt	fuel_cap	mpg	lnsales	partition
0	Acura	Integra	16.919	16.360	0.0	21.50	1.8	140.0	101.2	67.3	172.4	2.639	13.2	28.0	2.828	0
1	Acura	TL	39.384	19.875	0.0	28.40	3.2	225.0	108.1	70.3	192.9	3.517	17.2	25.0	3.673	0
2	Acura	RL	8.588	29.725	0.0	42.00	3.5	210.0	114.6	71.4	196.6	3.850	18.0	22.0	2.150	0
3	Audi	A4	20.397	22.255	0.0	23.99	1.8	150.0	102.6	68.2	178.0	2.998	16.4	27.0	3.015	0
4	Audi	A6	18.780	23.555	0.0	33.95	2.8	200.0	108.7	76.1	192.0	3.561	18.5	22.0	2.933	0

In [ ]:
```
# Feature selection
featureset = pdf[['engine_s', 'horsepow', 'wheelbas', 'width', 'length', 'curb_wgt', 'fuel_cap', 'mpg']]
featureset.head()
```

Out[915]:

	engine_s	horsepow	wheelbas	width	length	curb_wgt	fuel_cap	mpg
0	1.8	140.0	101.2	67.3	172.4	2.639	13.2	28.0
1	3.2	225.0	108.1	70.3	192.9	3.517	17.2	25.0
2	3.5	210.0	114.6	71.4	196.6	3.850	18.0	22.0
3	1.8	150.0	102.6	68.2	178.0	2.998	16.4	27.0
4	2.8	200.0	108.7	76.1	192.0	3.561	18.5	22.0

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler

# Normalize the dataset
x = featureset.values
min_max_scaler = MinMaxScaler()
feature_mtx = min_max_scaler.fit_transform(x)
feature_mtx[0:5]
```

Out[916]:
```
array([[0.11428571, 0.21518987, 0.18655098, 0.28143713, 0.30625832,
        0.2310559 , 0.13364055, 0.43333333],
       [0.31428571, 0.43037975, 0.3362256 , 0.46107784, 0.5792277 ,
        0.50372671, 0.31797235, 0.33333333],
       [0.35714286, 0.39240506, 0.47722343, 0.52694611, 0.62849534,
        0.60714286, 0.35483871, 0.23333333],
       [0.11428571, 0.24050633, 0.21691974, 0.33532934, 0.38082557,
        0.34254658, 0.28110599, 0.4       ],
       [0.25714286, 0.36708861, 0.34924078, 0.80838323, 0.56724368,
        0.5173913 , 0.37788018, 0.23333333]])
```

Agglomerative Clustering with Scipy

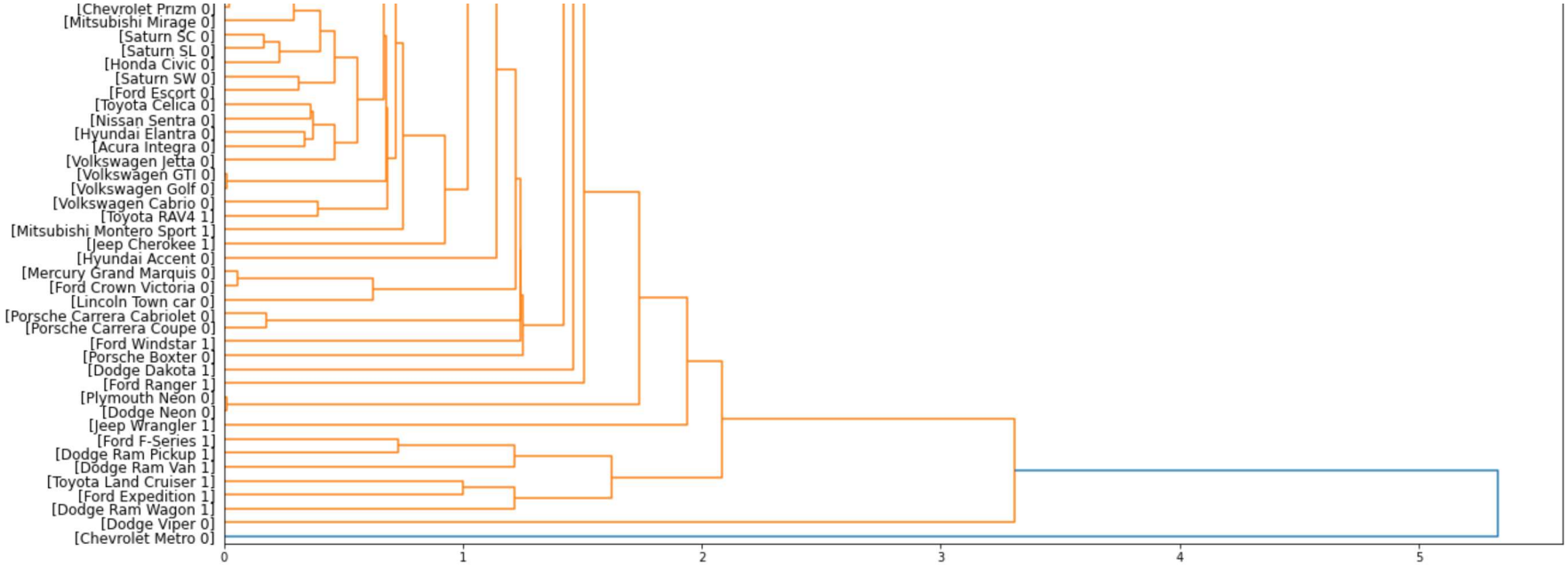
In [ ]:
```
# Agglomerative Clustering with scipy
import scipy
import scipy.cluster.hierarchy

leng = feature_mtx.shape[0]
D = scipy.zeros([leng,leng])
for i in range(leng):
    for j in range(leng):
        D[i,j] = scipy.spatial.distance.euclidean(feature_mtx[i], feature_mtx[j])

# Single Linkage
cars_single = hierarchy.linkage(D, 'single')

# Average Linkage
cars_average = hierarchy.linkage(D, 'average')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: DeprecationWarning: scipy.zeros is deprecated and will be removed in SciPy 2.0.0, use num
py.zeros instead

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix look
s suspiciously like an uncondensed distance matrix
  if sys.path[0] == '':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix look
s suspiciously like an uncondensed distance matrix
  from ipykernel import kernelapp as app
```

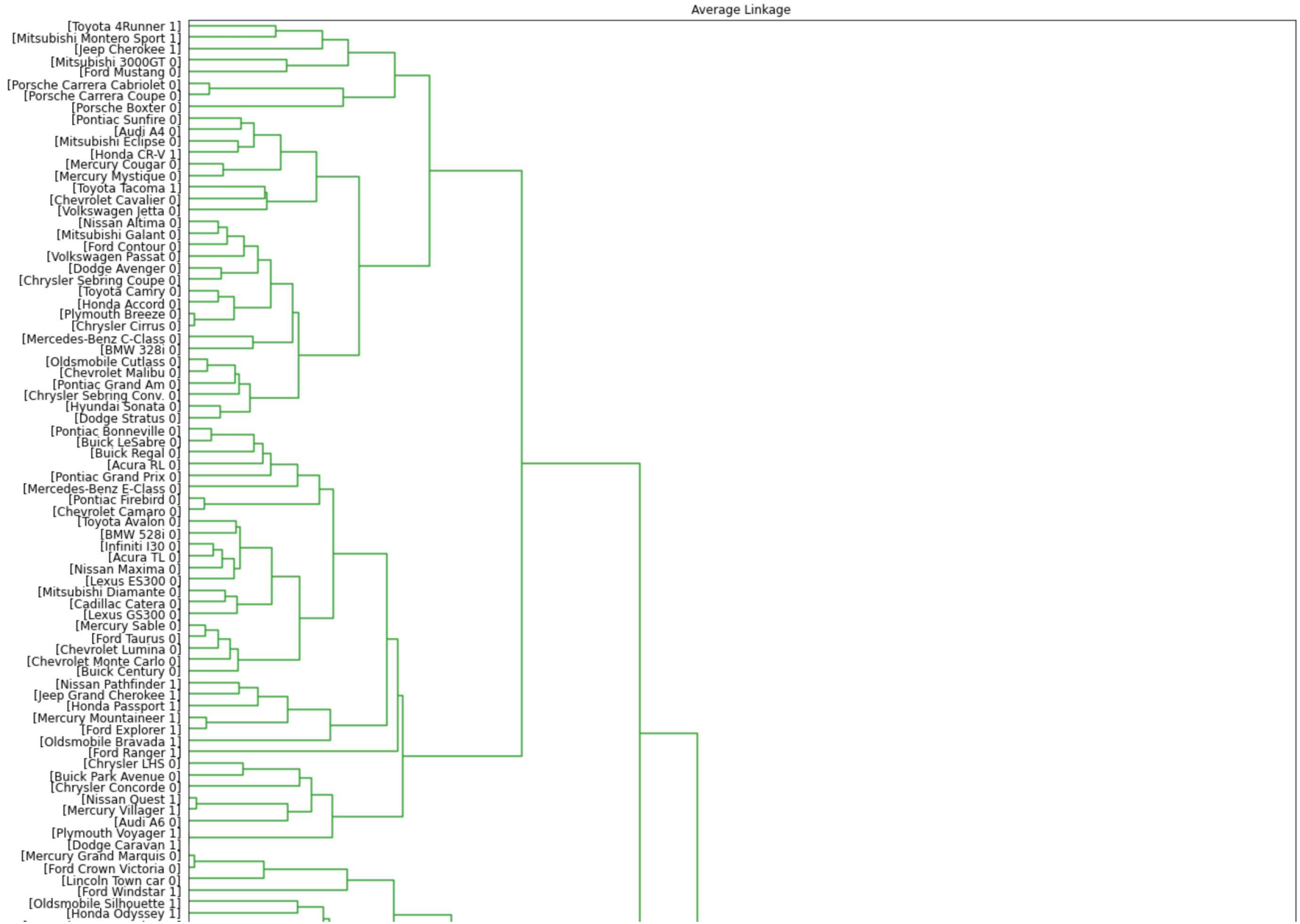
In [ ]:
```
# Fcluster
from scipy.cluster.hierarchy import fcluster
max_d = 3
clusters = fcluster(cars_single, max_d, criterion='distance')
clusters
```

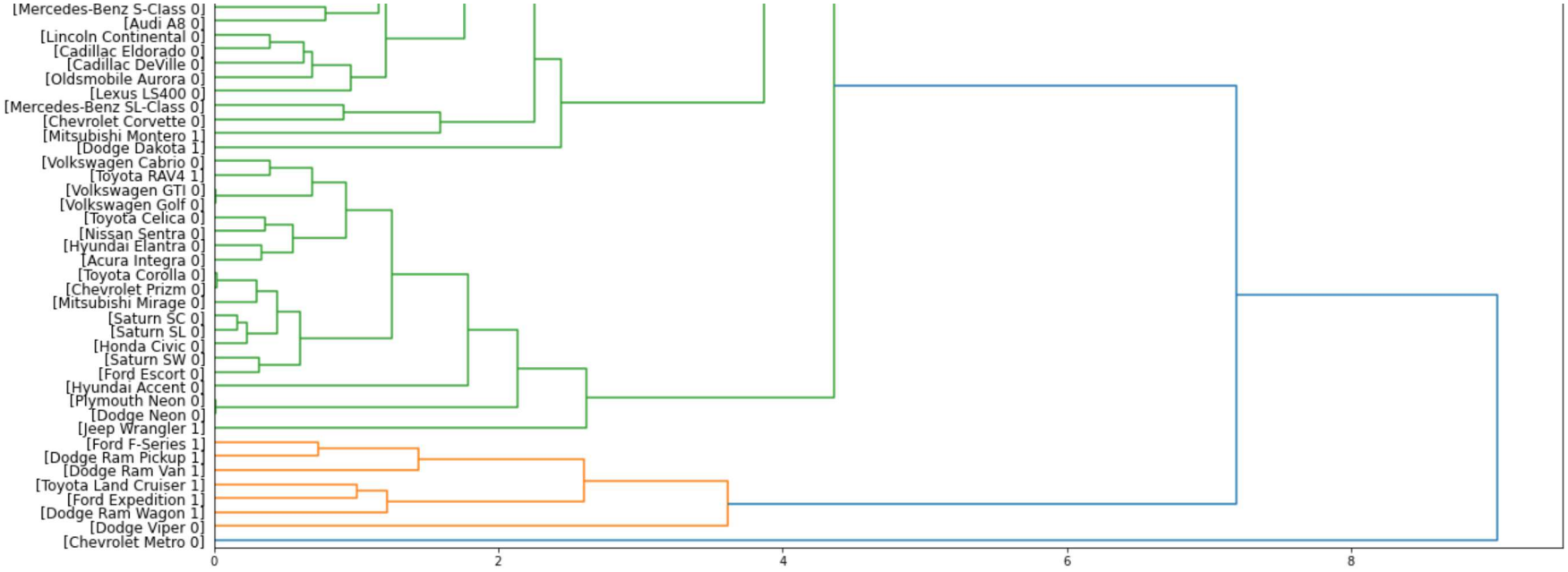
Out[918]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       3, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

In [ ]:
```
# Plot the dendogram for Single Linkage
fig = plt.figure(figsize=(20,25))
def llf(id):
    return '[%s %s %s]' % (pdf['manufact'][id], pdf['model'][id], int(float(pdf['type'][id])) )

plt.title('Single Linkage')
dendro = hierarchy.dendrogram(cars_single,  leaf_label_func=llf, leaf_rotation=0, leaf_font_size =12, orientation = 'right')
```


[Chevrolet Prizm 0]
[Mitsubishi Mirage 0]
[Saturn SC 0]
[Saturn SL 0]
[Honda Civic 0]
[Saturn SW 0]
[Ford Escort 0]
[Toyota Celica 0]
[Nissan Sentra 0]
[Hyundai Elantra 0]
[Acura Integra 0]
[Volkswagen Jetta 0]
[Volkswagen GTI 0]
[Volkswagen Golf 0]
[Volkswagen Cabrio 0]
[Toyota RAV4 1]
[Mitsubishi Montero Sport 1]
[Jeep Cherokee 1]
[Hyundai Accent 0]
[Mercury Grand Marquis 0]
[Ford Crown Victoria 0]
[Lincoln Town car 0]
[Porsche Carrera Cabriolet 0]
[Porsche Carrera Coupe 0]
[Ford Windstar 1]
[Porsche Boxter 0]
[Dodge Dakota 1]
[Ford Ranger 1]
[Plymouth Neon 0]
[Dodge Neon 0]
[Jeep Wrangler 1]
[Ford F-Series 1]
[Dodge Ram Pickup 1]
[Dodge Ram Van 1]
[Toyota Land Cruiser 1]
[Ford Expedition 1]
[Dodge Ram Wagon 1]
[Dodge Viper 0]
[Chevrolet Metro 0]
0
1
2
3
4
5

In [ ]:
```
# Plot the dendogram for Average Linkage
fig = plt.figure(figsize=(20,25))
def llf(id):
    return '[%s %s %s]' % (pdf['manufact'][id], pdf['model'][id], int(float(pdf['type'][id])) )

plt.title('Average Linkage')
dendro = hierarchy.dendrogram(cars_average,  leaf_label_func=llf, leaf_rotation=0, leaf_font_size =12, orientation = 'right')
```





Tugas 2

Perbedaan menggunakan metode single linkage dan average linkage terlihat pada bagaimana cara data mengelompok dalam sebuah kluster. Ketika menggunakan single linkage, data menumpuk pada cluster 1. Sedangkan ketika menggunakan average linkage, data cenderung lebih menyebar.

Agglomerative Clustering with scikit-learn

In [ ]:
```
# Get matrix distance
dist_matrix = distance_matrix(feature_mtx,feature_mtx)
print(dist_matrix)
```

```
[[0.         0.57777143 0.75455727 ... 0.28530295 0.24917241 0.18879995]
 [0.57777143 0.         0.22798938 ... 0.36087756 0.66346677 0.62201282]
 [0.75455727 0.22798938 0.         ... 0.51727787 0.81786095 0.77930119]
 ...
 [0.28530295 0.36087756 0.51727787 ... 0.         0.41797928 0.35720492]
 [0.24917241 0.66346677 0.81786095 ... 0.41797928 0.         0.15212198]
 [0.18879995 0.62201282 0.77930119 ... 0.35720492 0.15212198 0.        ]]
```

In [ ]:
```
# Hierarchical Clustering
single_agg = AgglomerativeClustering(n_clusters = 6, linkage = 'single').fit(feature_mtx)
avg_agg = AgglomerativeClustering(n_clusters = 6, linkage = 'average').fit(feature_mtx)
```

In [ ]:
```
single_agg.labels_
```

Out[923]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       3, 0, 0, 0, 0, 0, 0, 0, 0, 5, 1, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 2, 0, 0, 0, 0, 0])
```

In [ ]:
```
avg_agg.labels_
```

Out[924]:
```
array([0, 4, 4, 0, 4, 4, 0, 4, 4, 4, 4, 4, 4, 4, 4, 0, 0, 4, 4, 4, 1, 0,
       3, 0, 0, 4, 0, 4, 0, 0, 0, 1, 5, 2, 2, 4, 4, 0, 4, 0, 4, 4, 4, 4,
       2, 4, 5, 0, 0, 0, 4, 4, 0, 0, 0, 4, 0, 0, 4, 4, 4, 4, 4, 4, 0, 0,
       0, 4, 0, 4, 0, 0, 0, 4, 4, 4, 4, 0, 4, 4, 1, 0, 0, 4, 4, 4, 0, 4,
       4, 4, 0, 0, 4, 0, 0, 4, 4, 4, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0,
       0, 2, 0, 0, 0, 0, 0])
```

In [ ]:
```
# Copy the dataframe
single_df = pdf.copy(deep=True)
avg_df = pdf.copy(deep=True)

# Add the new class to the dataframe
single_df['cluster_'] = single_agg.labels_
avg_df['cluster_'] = avg_agg.labels_
```

In [ ]:
```
single_df.head()
```

Out[926]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	width	length	curb_wgt	fuel_cap	mpg	lnsales	partition	cluster_
0	Acura	Integra	16.919	16.360	0.0	21.50	1.8	140.0	101.2	67.3	172.4	2.639	13.2	28.0	2.828	0	0
1	Acura	TL	39.384	19.875	0.0	28.40	3.2	225.0	108.1	70.3	192.9	3.517	17.2	25.0	3.673	0	0
2	Acura	RL	8.588	29.725	0.0	42.00	3.5	210.0	114.6	71.4	196.6	3.850	18.0	22.0	2.150	0	0
3	Audi	A4	20.397	22.255	0.0	23.99	1.8	150.0	102.6	68.2	178.0	2.998	16.4	27.0	3.015	0	0
4	Audi	A6	18.780	23.555	0.0	33.95	2.8	200.0	108.7	76.1	192.0	3.561	18.5	22.0	2.933	0	0

In [ ]:
```
single_df.groupby(['cluster_', 'type'])['cluster_'].count()
```

Out[927]:
```
cluster_  type
0         0.0     86
          1.0     23
1         1.0      2
2         1.0      3
3         0.0      1
4         1.0      1
5         0.0      1
Name: cluster_, dtype: int64
```

In [ ]:
```
avg_df.head()
```

Out[928]:

	manufact	model	sales	resale	type	price	engine_s	horsepow	wheelbas	width	length	curb_wgt	fuel_cap	mpg	lnsales	partition	cluster_
0	Acura	Integra	16.919	16.360	0.0	21.50	1.8	140.0	101.2	67.3	172.4	2.639	13.2	28.0	2.828	0	0
1	Acura	TL	39.384	19.875	0.0	28.40	3.2	225.0	108.1	70.3	192.9	3.517	17.2	25.0	3.673	0	4
2	Acura	RL	8.588	29.725	0.0	42.00	3.5	210.0	114.6	71.4	196.6	3.850	18.0	22.0	2.150	0	4
3	Audi	A4	20.397	22.255	0.0	23.99	1.8	150.0	102.6	68.2	178.0	2.998	16.4	27.0	3.015	0	0
4	Audi	A6	18.780	23.555	0.0	33.95	2.8	200.0	108.7	76.1	192.0	3.561	18.5	22.0	2.933	0	4

In [ ]:
```
avg_df.groupby(['cluster_', 'type'])['cluster_'].count()
```

Out[929]:
```
cluster_  type
0         0.0     47
          1.0      7
1         0.0      3
2         1.0      4
3         0.0      1
4         0.0     37
          1.0     16
5         1.0      2
Name: cluster_, dtype: int64
```

In [ ]:
```
single_df.groupby(['cluster_', 'type'])['horsepow','engine_s','wheelbas','fuel_cap','mpg'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  """Entry point for launching an IPython kernel.
```

Out[930]:

		horsepow	engine_s	wheelbas	fuel_cap	mpg
cluster_	**type**					
0	**0.0**	181.197674	2.902326	105.808140	16.583721	25.358140
	1.0	167.478261	3.095652	109.704348	20.091304	21.000000
1	**1.0**	225.000000	4.900000	138.600000	25.550000	17.500000
2	**1.0**	215.000000	4.400000	119.466667	27.800000	15.666667
3	**0.0**	55.000000	1.000000	93.100000	10.300000	45.000000
4	**1.0**	175.000000	3.900000	109.600000	32.000000	15.000000
5	**0.0**	450.000000	8.000000	96.200000	19.000000	16.000000

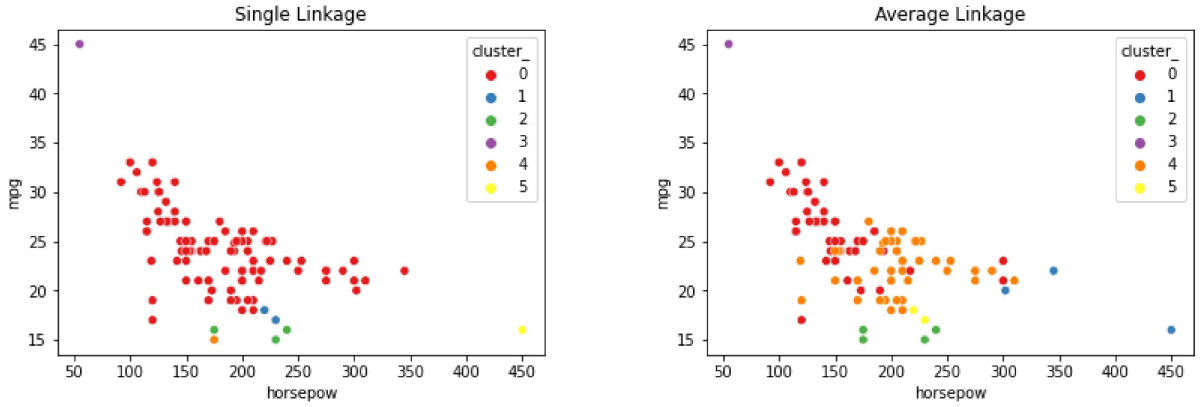
In [ ]:
```
avg_df.groupby(['cluster_', 'type'])['horsepow','engine_s','wheelbas','fuel_cap','mpg'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will
be deprecated, use a list instead.
  """Entry point for launching an IPython kernel.
```

Out[931]:

		horsepow	engine_s	wheelbas	fuel_cap	mpg
cluster_	**type**					
0	**0.0**	146.531915	2.246809	102.491489	14.980851	27.021277
	1.0	149.714286	2.657143	101.257143	17.528571	22.000000
1	**0.0**	365.666667	6.233333	99.900000	19.733333	19.333333
2	**1.0**	205.000000	4.275000	117.000000	28.850000	15.500000
3	**0.0**	55.000000	1.000000	93.100000	10.300000	45.000000
4	**0.0**	217.540541	3.602703	110.240541	18.429730	23.481081
	1.0	175.250000	3.287500	113.400000	21.212500	20.562500
5	**1.0**	225.000000	4.900000	138.600000	25.550000	17.500000

In [ ]:
```
# Import sns
import seaborn as sns

# Plot single_df and avg_df side-by-side
fig = plt.figure(figsize=(15,5))

# Single Linkage
# Give title and show the dendogram
ax1 = fig.add_axes([0.1,0.1,0.3,0.6])
sns.scatterplot(x='horsepow', y='mpg', hue='cluster_', data=single_df, palette='Set1')
ax1.set_title('Single Linkage')

# Average Linkage
# Give title and show the dendogram
ax2 = fig.add_axes([0.5,0.1,0.3,0.6])
sns.scatterplot(x='horsepow', y='mpg', hue='cluster_', data=avg_df, palette='Set1')
ax2.set_title('Average Linkage')

plt.show()
```



In [ ]:

Perbedaan single linkage dengan average linkage

Berdasarkan clustering di atas terdapat perbedaan pada single linkage dan average linkage apabila menggunakan clustering with Scipy dan scikit learn.

1. Single linkage menggabungkan cluster-cluster menurut jarak antara anggota-anggota terdekat di antara dua cluster. Sedangkan average linkage menggabungkan cluster-cluster menurut jarak rata-rata pasangan anggota masing-masing pada himpunan antara dua cluster.
2. Jadi single linkage perhitungan berdasarkan jarak tiap anggota apabila average menghitung jaraknya tiap pasangan. Pada single linkage pengelompokan berdasarkan jarak terdekat sedangkan average berdasarkan jarak rata-rata.

3 - Iris Dataset

Import dataset

In [ ]:
```
# Import iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
```

Understanding and preprocessing the data

In [ ]:
```
# Convert to df
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
df.head()
```

Out[934]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [ ]:
```
# Normalize the dataset
x = df.values
min_max_scaler = MinMaxScaler()
feature_mtx = min_max_scaler.fit_transform(x)
feature_mtx[0:5]
```

Out[935]:
```
array([[0.22222222, 0.625     , 0.06779661, 0.04166667, 0.        ],
       [0.16666667, 0.41666667, 0.06779661, 0.04166667, 0.        ],
       [0.11111111, 0.5       , 0.05084746, 0.04166667, 0.        ],
       [0.08333333, 0.45833333, 0.08474576, 0.04166667, 0.        ],
       [0.19444444, 0.66666667, 0.06779661, 0.04166667, 0.        ]])
```

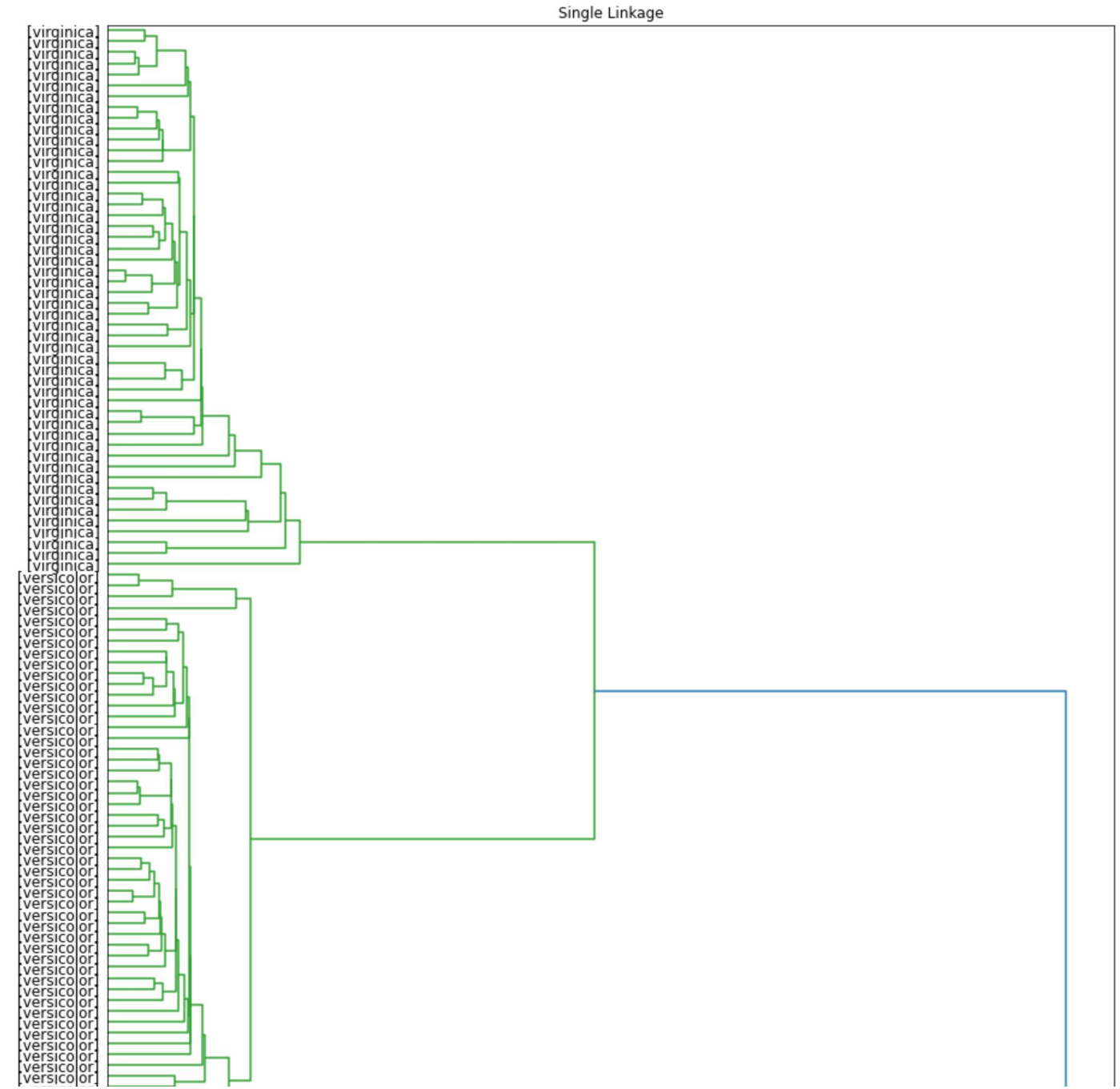
Agglomerative Clustering with scipy

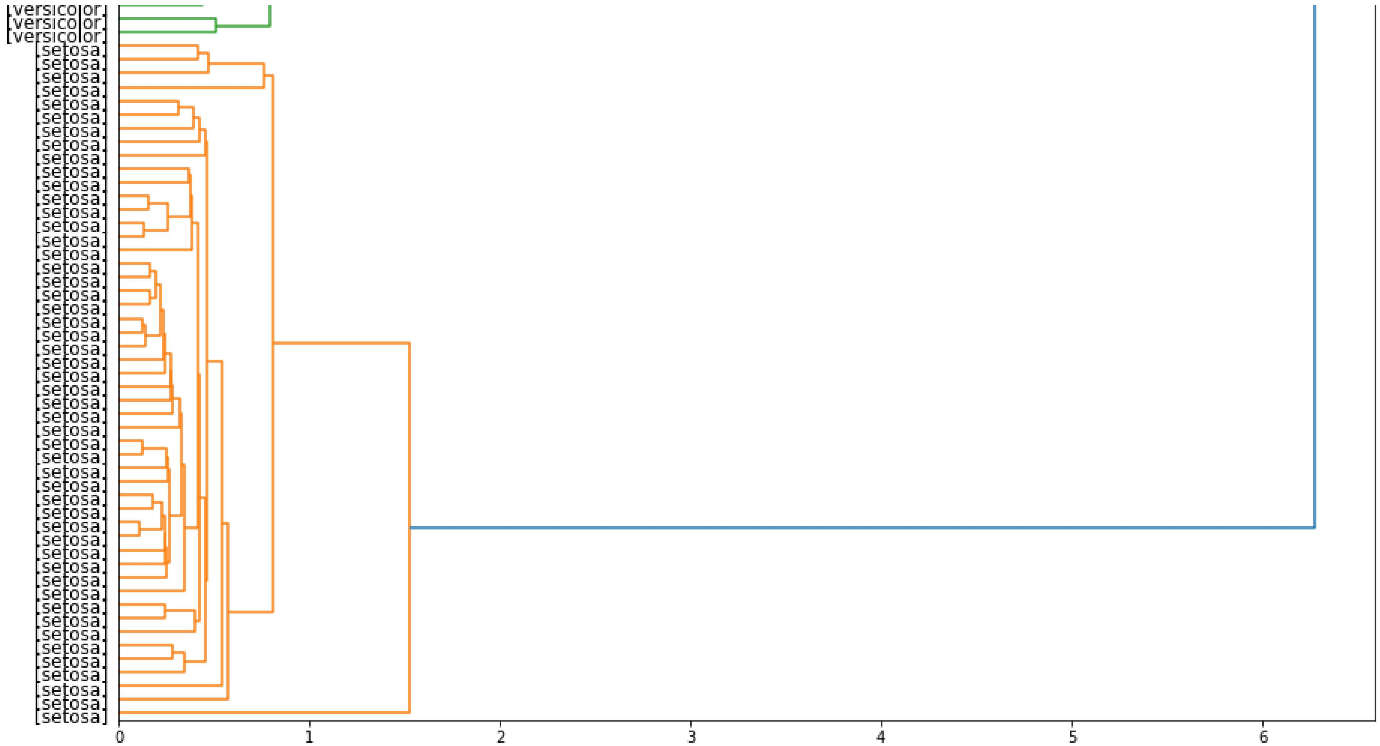
In [ ]:
```
# Agglomerative Clustering with scipy
import scipy
import scipy.cluster.hierarchy

leng = feature_mtx.shape[0]
D = scipy.zeros([leng,leng])
for i in range(leng):
    for j in range(leng):
        D[i,j] = scipy.spatial.distance.euclidean(feature_mtx[i], feature_mtx[j])

# Single Linkage
Z_single = hierarchy.linkage(D, 'single')

# Average Linkage
Z_avg = hierarchy.linkage(D, 'average')

# Compelte Linkage
Z_complete = hierarchy.linkage(D, 'complete')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: DeprecationWarning: scipy.zeros is deprecated and will be removed in SciPy 2.0.0, use num
py.zeros instead

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix look
s suspiciously like an uncondensed distance matrix
  if sys.path[0] == '':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix look
s suspiciously like an uncondensed distance matrix
  from ipykernel import kernelapp as app
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18: ClusterWarning: scipy.cluster: The symmetric non-negative hollow observation matrix look
s suspiciously like an uncondensed distance matrix
```

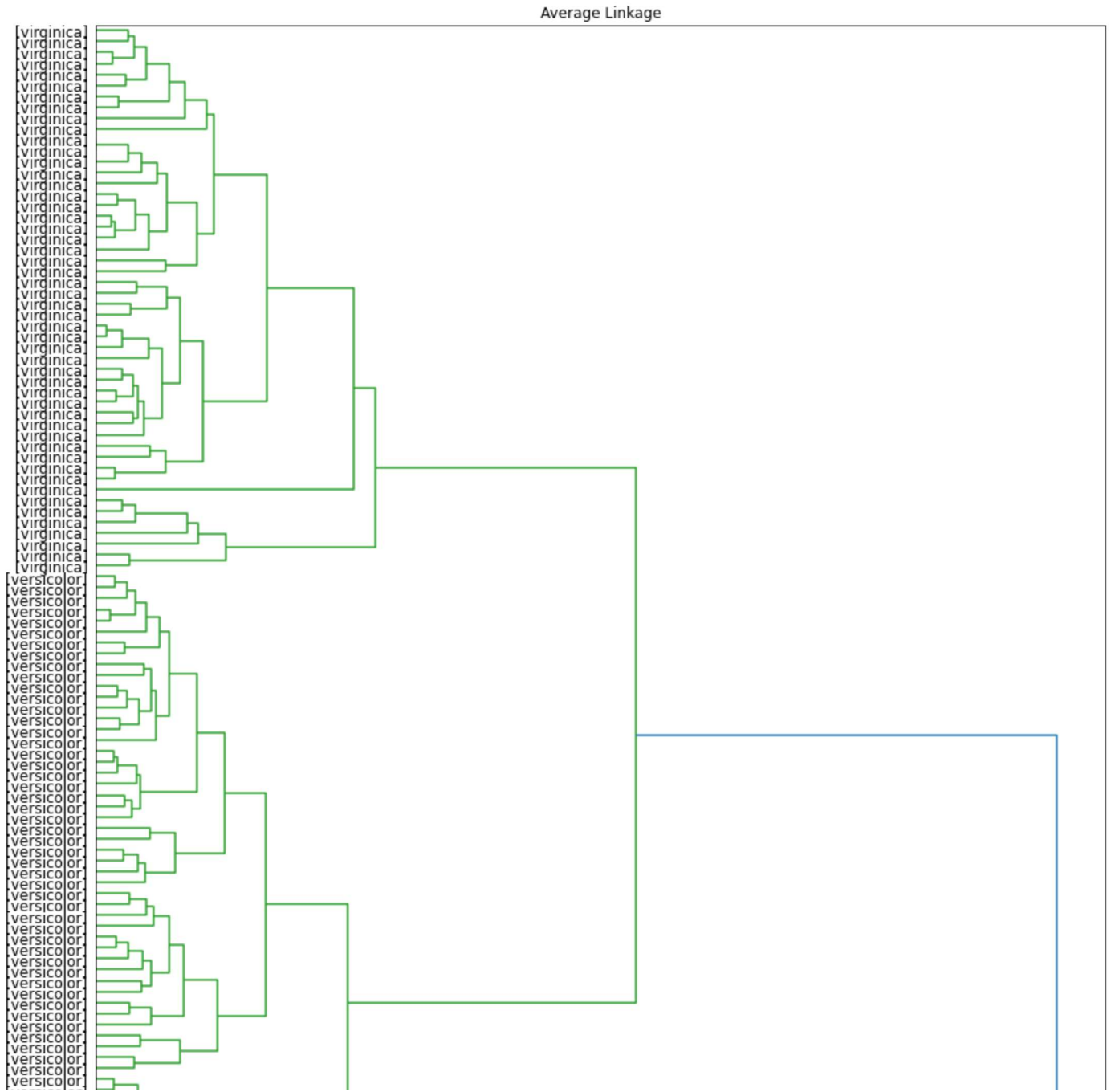
In [ ]:
```
# Plot the dendogram
def llf(id):
    return '[%s]' % ( iris.target_names[df['target'][id]] )
```

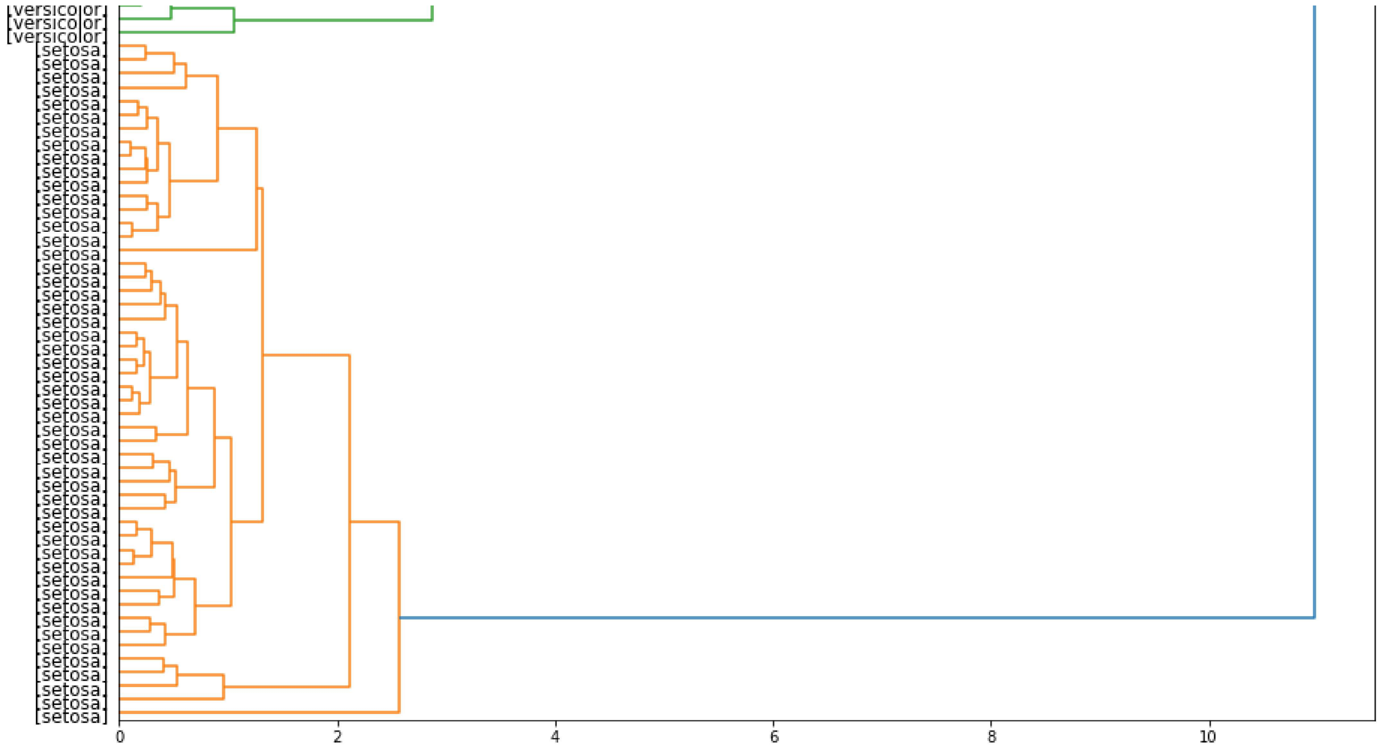
In [ ]:
```
plt.figure(figsize=(15,25))
plt.title('Single Linkage')
dendro = hierarchy.dendrogram(Z_single,  leaf_label_func=llf, leaf_rotation=0, leaf_font_size =12, orientation = 'right')
```




[versicolor]
[versicolor]
[versicolor]
[setosa]
0
1
2
3
4
5
6

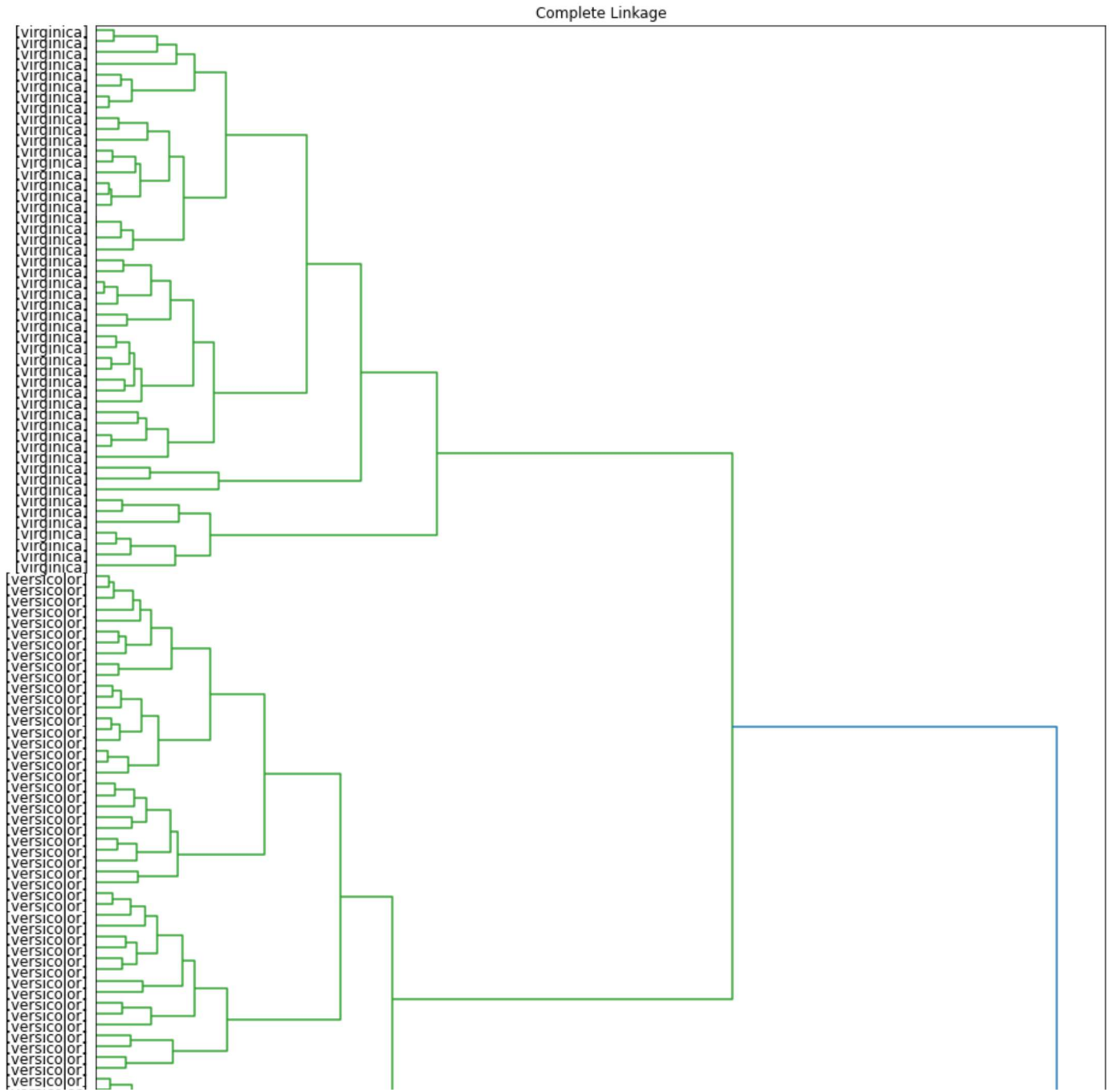
In [ ]:
```
plt.figure(figsize=(15,25))
plt.title('Average Linkage')
dendro = hierarchy.dendrogram(Z_avg,  leaf_label_func=llf, leaf_rotation=0, leaf_font_size =12, orientation = 'right')
```

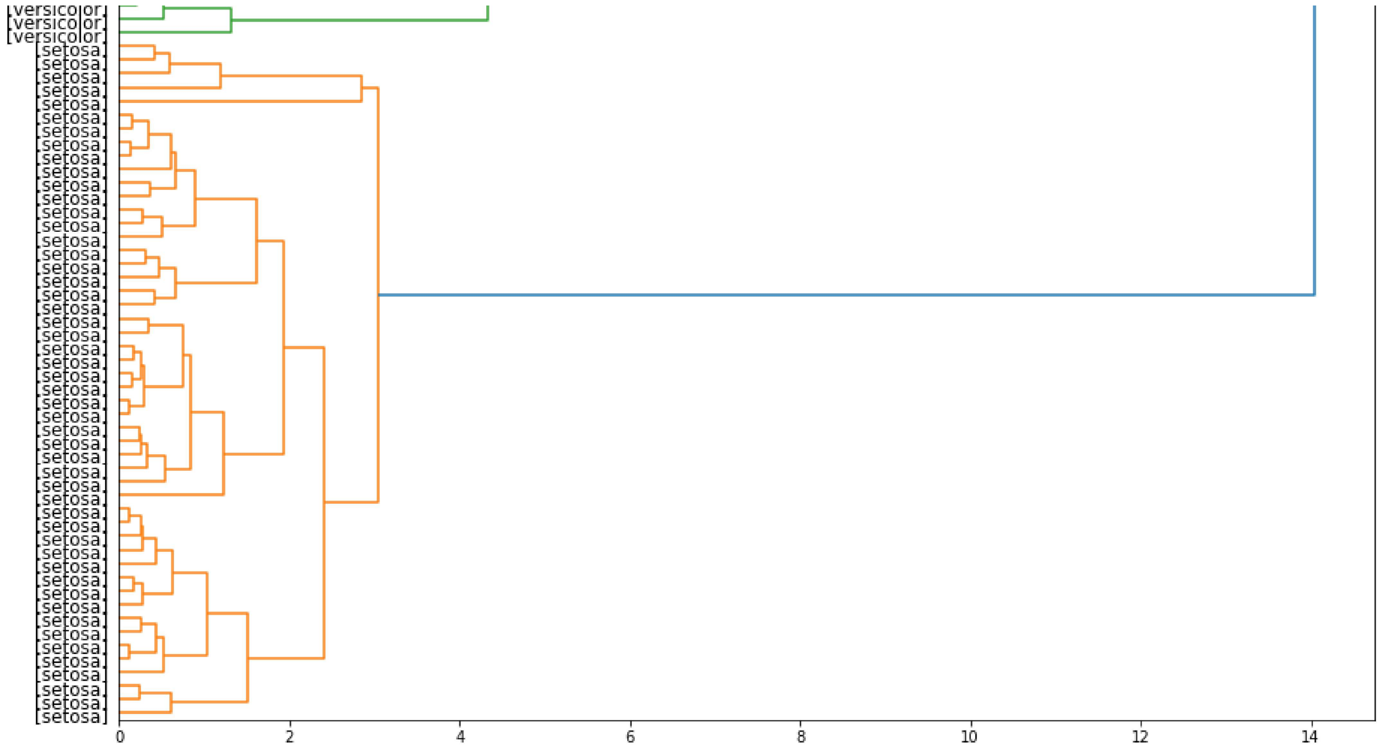




[versicolor]
[versicolor]
[versicolor]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
[setosa]
0
2
4
6
8
10

In [ ]:
```
plt.figure(figsize=(15,25))
plt.title('Complete Linkage')
dendro = hierarchy.dendrogram(Z_complete,  leaf_label_func=llf, leaf_rotation=0, leaf_font_size =12, orientation = 'right')
```





Agglomerative Clustering with sklearn

In [ ]:
```
# Agglomerative Clustering with sklearn
from sklearn.cluster import AgglomerativeClustering

# Single Linkage
single_agglo_clust = AgglomerativeClustering(n_clusters = 3, linkage = 'single')
single_agglo_clust.fit(feature_mtx)

# Average Linkage
avg_agglo_clust = AgglomerativeClustering(n_clusters = 3, linkage = 'average')
avg_agglo_clust.fit(feature_mtx)

# Complete Linkage
complete_agglo_clust = AgglomerativeClustering(n_clusters = 3, linkage = 'complete')
complete_agglo_clust.fit(feature_mtx)

print("Single Linkage")
print(single_agglo_clust.labels_)
print("-"*75)
print("Average Linkage")
print(avg_agglo_clust.labels_)
print("-"*75)
print("Complete Linkage")
print(complete_agglo_clust.labels_)
```

```
Single Linkage
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0]
---------------------------------------------------------------------------
Average Linkage
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0]
---------------------------------------------------------------------------
Complete Linkage
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 2 1 2 2 1 2 2 2
 1 2 1 1 2 1 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 1 1 2 2 1 1 2 2 2 1 2 2 2 1 2 2
 2 1]
```

In [ ]:
```
# Copy the dataframe
single_df = df.copy(deep=True)
avg_df = df.copy(deep=True)
complete_df = df.copy(deep=True)

# Add the new class to the dataframe
single_df['cluster_'] = single_agglo_clust.labels_
avg_df['cluster_'] = avg_agglo_clust.labels_
complete_df['cluster_'] = complete_agglo_clust.labels_
```

In [ ]:
```
avg_df.head()
```

Out[943]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	cluster_
0	5.1	3.5	1.4	0.2	0	1
1	4.9	3.0	1.4	0.2	0	1
2	4.7	3.2	1.3	0.2	0	1
3	4.6	3.1	1.5	0.2	0	1
4	5.0	3.6	1.4	0.2	0	1

In [ ]:
```
complete_df.head()
```

Out[944]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	cluster_
0	5.1	3.5	1.4	0.2	0	0
1	4.9	3.0	1.4	0.2	0	0
2	4.7	3.2	1.3	0.2	0	0
3	4.6	3.1	1.5	0.2	0	0
4	5.0	3.6	1.4	0.2	0	0

In [ ]:
```
single_df.groupby(['cluster_', 'target'])['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  """Entry point for launching an IPython kernel.
```

Out[945]:

		sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
cluster_	**target**				
0	**2**	6.588	2.974	5.552	2.026
1	**0**	5.006	3.428	1.462	0.246
2	**1**	5.936	2.770	4.260	1.326

In [ ]:
```
single_df.groupby(['cluster_', 'target'])['cluster_'].count()
```

Out[946]:
```
cluster_  target
0         2         50
1         0         50
2         1         50
Name: cluster_, dtype: int64
```

In [ ]:
```
avg_df.groupby(['cluster_', 'target'])['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will
be deprecated, use a list instead.
  """Entry point for launching an IPython kernel.
```

Out[947]:

		sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
cluster_	**target**				
0	**2**	6.588	2.974	5.552	2.026
1	**0**	5.006	3.428	1.462	0.246
2	**1**	5.936	2.770	4.260	1.326

In [ ]:
```
avg_df.groupby(['cluster_', 'target'])['cluster_'].count()
```

Out[948]:
```
cluster_  target
0         2         50
1         0         50
2         1         50
Name: cluster_, dtype: int64
```

In [ ]:
```
complete_df.groupby(['cluster_', 'target'])['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)'].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will
be deprecated, use a list instead.
  """Entry point for launching an IPython kernel.
```

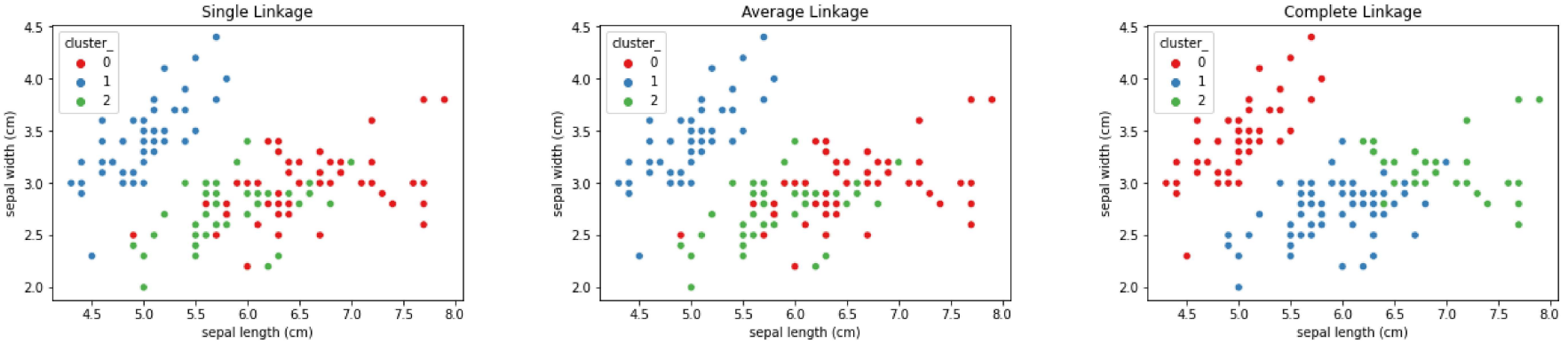
Out[949]:

		sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
cluster_	**target**				
0	**0**	5.006000	3.428	1.462000	0.246000
1	**1**	5.936000	2.770	4.260000	1.326000
	2	6.055000	2.740	5.130000	1.815000
2	**2**	6.943333	3.130	5.833333	2.166667

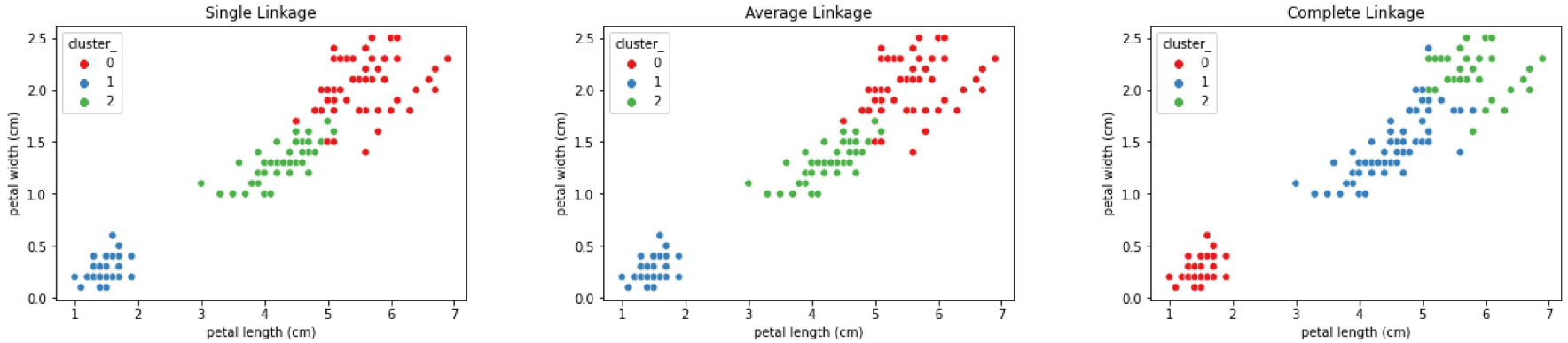
In [ ]:
```
complete_df.groupby(['cluster_', 'target'])['cluster_'].count()
```

Out[950]:
```
cluster_  target
0         0         50
1         1         50
          2         20
2         2         30
Name: cluster_, dtype: int64
```

In [ ]:
```
# Plot cluster
fig = plt.figure(figsize=(15,5))

# Compare Single, Average, and Complete Linkage
# Single Linkage
ax1 = fig.add_axes([0.1,0.1,0.3,0.6])
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', hue='cluster_', data=single_df, palette='Set1')
ax1.set_title('Single Linkage')

# Average Linkage
ax2 = fig.add_axes([0.5,0.1,0.3,0.6])
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', hue='cluster_', data=avg_df, palette='Set1')
ax2.set_title('Average Linkage')

# Complete Linkage
ax3 = fig.add_axes([0.9,0.1,0.3,0.6])
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', hue='cluster_', data=complete_df, palette='Set1')
ax3.set_title('Complete Linkage')

plt.show()
```



In [ ]:
```
# Plot cluster
fig = plt.figure(figsize=(15,5))

# Compare Single, Average, and Complete Linkage
# Single Linkage
ax1 = fig.add_axes([0.1,0.1,0.3,0.6])
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='cluster_', data=single_df, palette='Set1')
ax1.set_title('Single Linkage')

# Average Linkage
ax2 = fig.add_axes([0.5,0.1,0.3,0.6])
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='cluster_', data=avg_df, palette='Set1')
ax2.set_title('Average Linkage')

# Complete Linkage
ax3 = fig.add_axes([0.9,0.1,0.3,0.6])
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='cluster_', data=complete_df, palette='Set1')
ax3.set_title('Complete Linkage')

plt.show()
```



Tugas 3

Perbedaan single linkage dengan average linkage

Berdasarkan clustering di atas terdapat perbedaan pada single linkage dan average linkage apabila menggunakan clustering with Scipy dan scikit learn.

1. Pada Single linkage menggabungkan cluster-cluster menurut jarak antara anggota-anggota terdekat di antara dua cluster. Sedangkan average linkage menggabungkan cluster-cluster menurut jarak rata-rata pasangan anggota masing-masing pada himpunan antara dua cluster.
2. Sehingga pada single linkage, perhitungan berdasarkan jarak tiap anggota apabila average menghitung jaraknya tiap pasangan. Pada single linkage pengelompokan berdasarkan jarak terdekat sedangkan average berdasarkan jarak rata-rata.