20장. 군집화 (Clustering)

In [1]:

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:₩users₩ghkrg₩anaconda3₩envs₩data_mining
₩lib₩site-packages (0.11.1)
Requirement already satisfied: pandas>=0.23 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mi
ning₩lib₩site-packages (from seaborn) (1.2.3)
Requirement already satisfied: numpy>=1.15 in c:₩users₩ghkrg₩anaconda3₩envs₩data_min
ing₩lib₩site-packages (from seaborn) (1.20.1)
Requirement already satisfied: matplotlib>=2.2 in c:₩users₩ghkrg₩anaconda3₩envs₩data
_mining₩lib₩site-packages (from seaborn) (3.3.4)
Requirement already satisfied: scipy>=1.0 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mini
ng₩lib₩site-packages (from seaborn) (1.6.3)
Requirement already satisfied: cycler>=0.10 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mi
ning₩lib₩site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:₩users
₩ghkrg₩anaconda3₩envs₩data_mining₩lib₩site-packages (from matplotlib>=2.2->seaborn)
(2.4.7)
Requirement already satisfied: pillow>=6.2.0 in c:₩users₩ghkrg₩anaconda3₩envs₩data_m
ining₩lib₩site-packages (from matplotlib>=2.2->seaborn) (8.1.2)
Requirement already satisfied: python-dateutil>=2.1 in c:₩users₩ghkrg₩anaconda3₩envs
₩data_mining₩lib₩site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:₩users₩ghkrg₩anaconda3₩envs₩da
ta_mining₩lib₩site-packages (from matplotlib>=2.2->seaborn) (1.3.1)
Requirement already satisfied: six in c:₩users₩ghkrg₩anaconda3₩envs₩data_mining₩lib
₩site-packages (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:₩users₩ghkrg₩anaconda3₩envs₩data_mi
ning₩lib₩site-packages (from pandas>=0.23->seaborn) (2021.1)
```

1. 데이터셋 (IRIS)

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data)

In [2]:

```
import requests
import os

data = requests.get("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")
path = os.path.join('data', 'iris.data')
with open(path, "w") as f:
  f.write(data.text)
```

1.1 데이터셋 읽기

In [3]:

```
import pandas as pd
column_names = ['sepal length', 'sepal width', 'petal length', 'petal width', 'species']
dataset = pd.read_csv(path, names=column_names)
dataset.head()
```

Out[3]:

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal length  150 non-null    float64
 1   sepal width   150 non-null    float64
 2   petal length  150 non-null    float64
 3   petal width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

2. 데이터 탐색

2.1 요약 통계량

In [5]:

```
1 dataset.describe()
```

Out[5]:

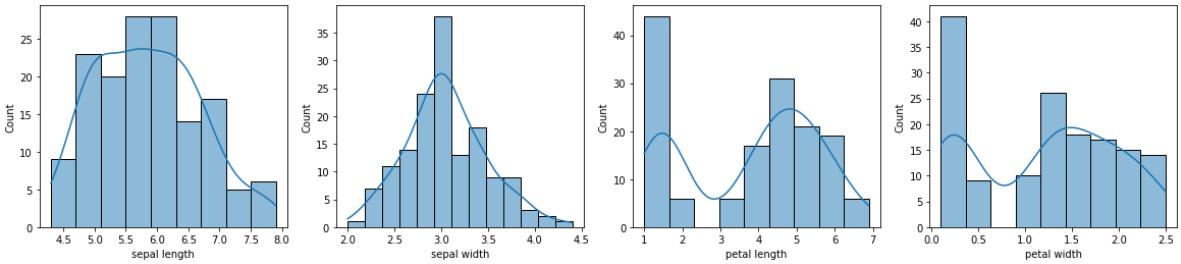
	sepal length	sepal width	petal length	petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

2.2 단일 변수 분석

2.2.1 히스토그램

In [6]:

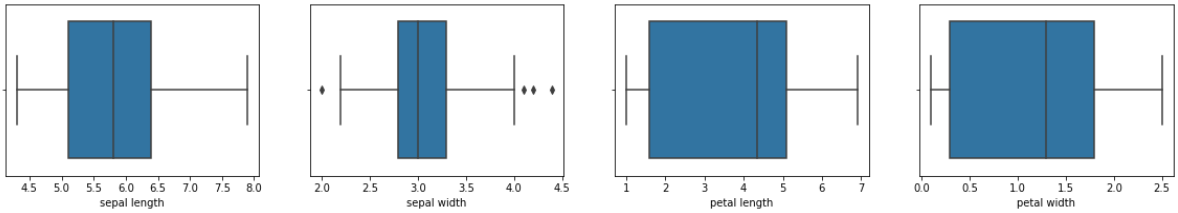
```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 plt.figure(figsize=[20,4])
5 for i, column in enumerate(dataset.describe().columns):
6     plt.subplot(1,4,i+1)
7     sns.histplot(data=dataset, x=column, kde=True)
8 plt.show()
```



2.2.2 박스 플롯

In [7]:

```
plt.figure(figsize=[20,3])
for i in enumerate(dataset.describe().columns[:4]):
    plt.subplot(1,4,i[0]+1)
    sns.boxplot(x=dataset[i[1]])

plt.show()
```

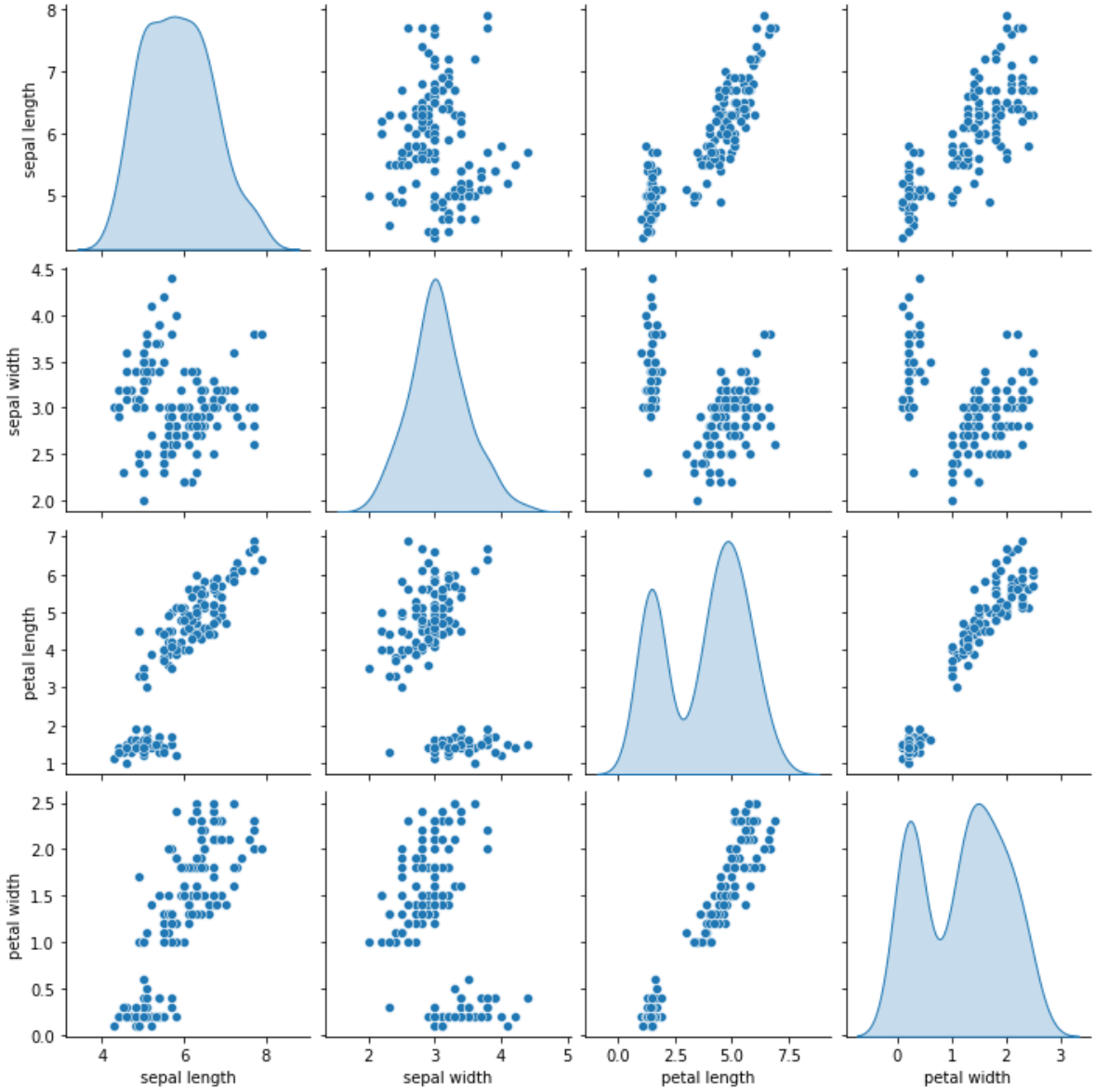


2.3 두 변수 관계 분석

2.3.1 산포도 행렬

In [8]:

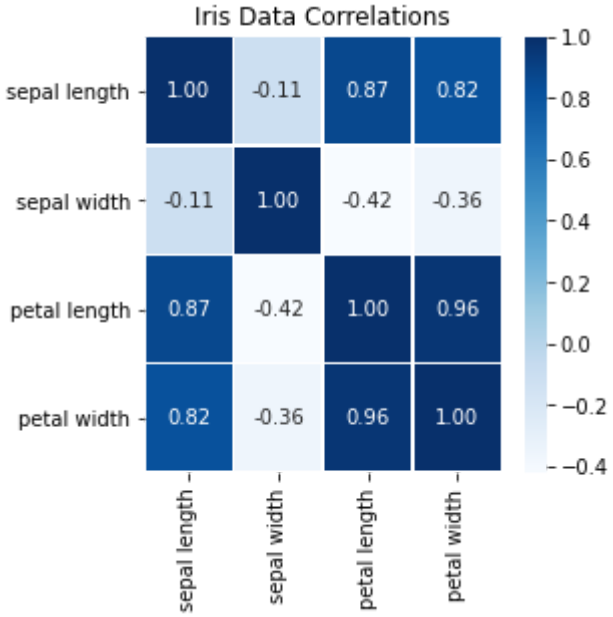
```
1 sns.pairplot(dataset, diag_kind="kde")
2 plt.show()
```



2.3.2 히트맵

In [9]:

```
fig, ax = plt.subplots(figsize=(4, 4))
sns.heatmap(dataset.corr(), linewidths=.5, annot=True, fmt=".2f", cmap='Blues')
plt.title('Iris Data Correlations')
plt.show()
```



3. 데이터 전처리

3.1 데이터 추출

In [10]:

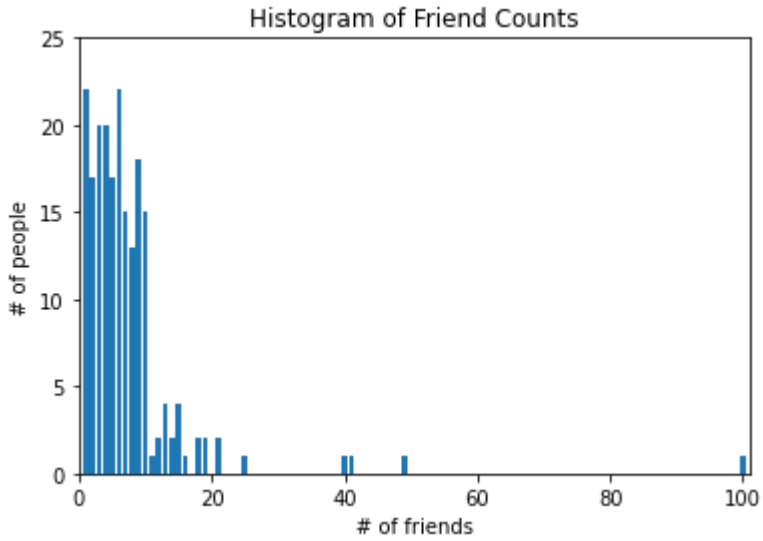
```
clusterdata = dataset.iloc[:,:-1]
inputs = clusterdata.iloc[:,:].values.tolist()
columns = clusterdata.keys().tolist()
column2index ={ column : i for i, column in enumerate(columns)}
print('columns = ', columns)
print('column2index = ', column2index)
```

```
columns =  ['sepal length', 'sepal width', 'petal length', 'petal width']
column2index =  {'sepal length': 0, 'sepal width': 1, 'petal length': 2, 'petal widt
h': 3}
```

3.2 데이터 표준화

In [11]:

```
from scratch.working_with_data import scale, rescale, Vector
from typing import List

inputs_normed = rescale(inputs)
```



4. K-평균 군집화

Q1. 손실 곡선을 보고 K 선택하기

손실을 최소화 하는 클러스터 수 K를 찾아보시오. 단, K는 20까지 확인해 보라.

In [12]:

```
from scratch.linear_algebra import Vector

def num_differences(v1: Vector, v2: Vector) -> int:
    assert len(v1) == len(v2)
    return len([x1 for x1, x2 in zip(v1, v2) if x1 != x2])
```

In [13]:

```
from typing import List
from scratch.linear_algebra import vector_mean

def cluster_means(k: int,
                  inputs: List[Vector],
                  assignments: List[int]) -> List[Vector]:
    clusters = [[] for i in range(k)]
    for input, assignment in zip(inputs, assignments):
        clusters[assignment].append(input)

    return [vector_mean(cluster) if cluster else random.choice(inputs) for cluster in cluster
```

In [14]:

```
import itertools
import random
import tqdm
from scratch.linear_algebra import squared_distance

class KMeans:
    def __init__(self, k: int) -> None:
        self.k = k
        self.means = None

    def classify(self, input: Vector) -> int:
        return min(range(self.k), key=lambda i: squared_distance(input, self.means[i]))

    def train(self, inputs: List[Vector]) -> None:
        assignments = [random.randrange(self.k) for _ in inputs]
        with tqdm.tqdm(itertools.count()) as t:
            for _ in t:
                self.means = cluster_means(self.k, inputs, assignments)
                new_assignments = [self.classify(input) for input in inputs]
                num_changed = num_differences(assignments, new_assignments)
                if num_changed == 0:
                    return

                assignments = new_assignments
                t.set_description(f"chagned: {num_changed} / {len(inputs)}")
```

In [15]:

```
# your code
import random
from matplotlib import pyplot as plt

def squared_clustering_errors(inputs: List[Vector], k: int) -> float:
    clusterer = KMeans(k)
    clusterer.train(inputs)
    means = clusterer.means
    assignments = [clusterer.classify(input) for input in inputs]

    return sum(squared_distance(input, means[cluster]) for input, cluster in zip(inputs, assig


ks = range(1, 21)
errors = [squared_clustering_errors(inputs, k) for k in ks]

optimal_k = errors.index(min(errors)) + 1

fig, zx = plt.subplots(figsize=(10, 5))
plt.plot(ks, errors)
plt.xticks(ks)
plt.xlabel(f"k (Optimal k = {optimal_k})")
plt.ylabel("total squared error")
plt.title("Total Error vs. # of Clusters")
plt.show()
```

```
0it [00:00, ?it/s]
chagned: 1 / 150: : 3it [00:00, 749.74it/s]
chagned: 1 / 150: : 10it [00:00, 769.05it/s]
chagned: 1 / 150: : 8it [00:00, 615.26it/s]
chagned: 1 / 150: : 6it [00:00, 545.35it/s]
chagned: 2 / 150: : 7it [00:00, 499.91it/s]
chagned: 3 / 150: : 11it [00:00, 439.91it/s]
chagned: 4 / 150: : 9it [00:00, 374.92it/s]
chagned: 1 / 150: : 6it [00:00, 315.70it/s]
chagned: 1 / 150: : 9it [00:00, 310.27it/s]
chagned: 2 / 150: : 8it [00:00, 296.23it/s]
chagned: 3 / 150: : 12it [00:00, 272.66it/s]
chagned: 2 / 150: : 5it [00:00, 217.34it/s]
chagned: 4 / 150: : 6it [00:00, 230.72it/s]
chagned: 1 / 150: : 9it [00:00, 224.95it/s]
chagned: 3 / 150: : 6it [00:00, 199.96it/s]
chagned: 1 / 150: : 6it [00:00, 181.77it/s]
chagned: 1 / 150: : 6it [00:00, 181.78it/s]
chagned: 1 / 150: : 7it [00:00, 174.96it/s]
chagned: 1 / 150: : 14it [00:00, 177.18it/s]
```

Q2. 군집화 및 결과 확인 (Q)

K=3으로 군집화를 해서 다음과 같이 군집화 결과를 확인해 보라.

In [16]:

```
from typing import NamedTuple, Union

class Leaf(NamedTuple):
    value: Vector
```

In [17]:

```
class Merged(NamedTuple):
    children: tuple
    order: int
```

In [18]:

```
Cluster = Union[Leaf, Merged]
```

In [19]:

```
def get_values(cluster: Cluster) -> List[Vector]:
    if isinstance(cluster, Leaf):
        return [cluster.value]
    else:
        return [value
                for child in cluster.children
                for value in get_values(child)]
```

In [20]:

```
from typing import Callable
from scratch.linear_algebra import distance

def cluster_distance(cluster1: Cluster,
                     cluster2: Cluster,
                     distance_agg: Callable = min) -> float:
    return distance_agg([distance(v1, v2)
                         for v1 in get_values(cluster1)
                         for v2 in get_values(cluster2)])
```

In [21]:

```
def get_merge_order(cluster: Cluster) -> float:
    if isinstance(cluster, Leaf):
        return float('inf')
    else:
        return cluster.order
```

In [22]:

```
from typing import Tuple

def get_children(cluster: Cluster):
    if isinstance(cluster, Leaf):
        raise TypeError("Leaf has no children")
    else:
        return cluster.children
```

In [23]:

```
def bottom_up_cluster(inputs: List[Vector],
                      distance_agg: Callable = min) -> Cluster:
    clusters: List[Cluster] = [Leaf(input) for input in inputs]
```

In [24]:

```
def pair_distance(pair: Tuple[Cluster, Cluster]) -> float:
    return cluster_distance(pair[0], pair[1], distance_agg)
```

In [25]:

```
def generate_clusters(base_cluster: Cluster,
                      num_clusters: int) -> List[Cluster]:
    clusters = [base_cluster]

    while len(clusters) < num_clusters:
        next_cluster = min(clusters, key=get_merge_order)
        clusters = [c for c in clusters if c != next_cluster]
        clusters.extend(get_children(next_cluster))
    return clusters
```

In [26]:

```
#your code
#base_cluster = bottom_up_cluster(inputs)
#three_clusters = [get_values(cluster)
 #                 for cluster in generate_clusters(base_cluster, 3)]

random.seed(0)
clusterer = KMeans(k=3)
clusterer.train(inputs)
means = sorted(clusterer.means)
assignments = [clusterer.classify(input) for input in inputs]

assert len(means) == 3
```

chagned: 1 / 150: : 11it [00:00, 785.53it/s]

dataset에 k_means 컬럼 추가

In [27]:

```
dataset["k_means"] = assignments
dataset.head()
```

Out[27]:

	sepal length	sepal width	petal length	petal width	species	k_means
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1

species와 k_means 결과 비교

In [28]:

```
dataset[dataset['k_means']==0].head()
```

Out[28]:

	sepal length	sepal width	petal length	petal width	species	k_means
50	7.0	3.2	4.7	1.4	Iris-versicolor	0
52	6.9	3.1	4.9	1.5	Iris-versicolor	0
77	6.7	3.0	5.0	1.7	Iris-versicolor	0
100	6.3	3.3	6.0	2.5	Iris-virginica	0
102	7.1	3.0	5.9	2.1	Iris-virginica	0

In [29]:

```
dataset.groupby(["k_means", "species"])['k_means'].count()
```

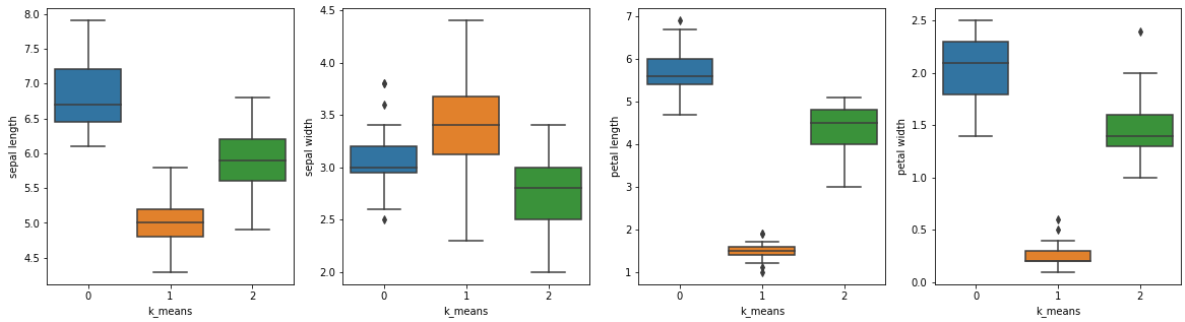
Out[29]:

```
k_means  species
0        Iris-versicolor     3
         Iris-virginica     36
1        Iris-setosa        50
2        Iris-versicolor    47
         Iris-virginica     14
Name: k_means, dtype: int64
```

군집화 결과 시각화

In [30]:

```
plt.subplots(figsize=(20, 5))
plt.subplot(1,4,1)
sns.boxplot(x='k_means', y='sepal length', data=dataset)
plt.subplot(1,4,2)
sns.boxplot(x='k_means', y='sepal width', data=dataset)
plt.subplot(1,4,3)
sns.boxplot(x='k_means', y='petal length', data=dataset)
plt.subplot(1,4,4)
sns.boxplot(x='k_means', y='petal width', data=dataset)
plt.show()
```



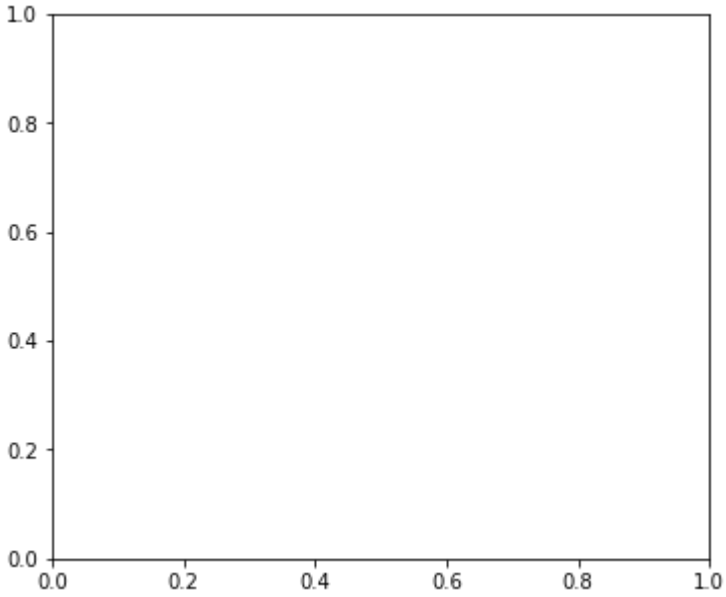
Q3. 군집화 및 결과 확인

각 군집이 구분되도록 두 변수의 산포도를 그리는 함수 plot_cluster 구현하시오.

In [31]:

```
plt.subplots(figsize=(20, 5))
plt.subplot(1,3,1)
plot_cluster(clusters, column2index["sepal length"], column2index["sepal width"])
plt.subplot(1,3,2)
plot_cluster(clusters, column2index["petal length"], column2index["petal width"])
plt.subplot(1,3,3)
plot_cluster(clusters, column2index["sepal length"], column2index["petal length"])
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-31-6fa796c3a7a0> in <module>
      1 plt.subplots(figsize=(20, 5))
      2 plt.subplot(1,3,1)
----> 3 plot_cluster(clusters, column2index["sepal length"], column2index["sepa
l width"])
      4 plt.subplot(1,3,2)
      5 plot_cluster(clusters, column2index["petal length"], column2index["petal
width"])

NameError: name 'plot_cluster' is not defined
```



5. 계층 군집화 (Hierarchical Clustering)

Q4. 군집화 및 결과 확인

K=3으로 최장 거리(max) 기준으로 군집화를 해서 다음과 같이 군집화 결과를 확인해 보라.

In [32]:

```
1 # your code
```

dataset에 h_clustering 컬럼 추가

In [33]:

```
1 dataset["h_clustering"] = h_assignments
2 dataset.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-33-4f021a770fb5> in <module>
----> 1 dataset["h_clustering"] = h_assignments
      2 dataset.head()

NameError: name 'h_assignments' is not defined
```

h_clustering과 species 비교

In [34]:

```
1 dataset.groupby(["h_clustering", "species"])['h_clustering'].count()
```

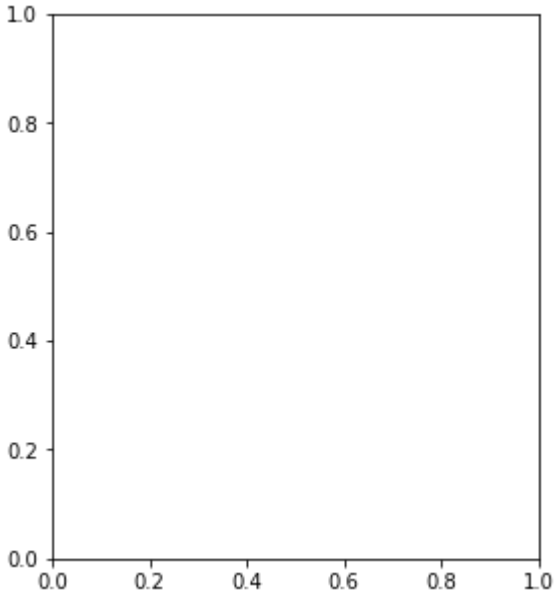
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-34-534101fc94bb> in <module>
----> 1 dataset.groupby(["h_clustering", "species"])['h_clustering'].count()

~Wanaconda3Wenvs₩data_miningWlibWsite-packagesWpandasWcoreWframe.py in grou
pby(self, by, axis, level, as_index, sort, group_keys, squeeze, observed,
 dropna)
   6715         axis = self._get_axis_number(axis)
   6716
-> 6717         return DataFrameGroupBy(
   6718             obj=self,
   6719             keys=by,

~Wanaconda3Wenvs₩data_miningWlibWsite-packagesWpandasWcoreWgroupbyWgroupb
y.py in __init__(self, obj, keys, axis, level, grouper, exclusions, selecti
on, as_index, sort, group_keys, squeeze, observed, mutated, dropna)
    558             from pandas.core.groupby.grouper import get_grouper
    559
--> 560             grouper, exclusions, obj = get_grouper(
    561                 obj,
    562                 keys,

~Wanaconda3Wenvs₩data_miningWlibWsite-packagesWpandasWcoreWgroupbyWgroupe
r.py in get_grouper(obj, key, axis, level, sort, observed, mutated, validat
e, dropna)
    809                 in_axis, name, level, gpr = False, None, gpr, None
    810             else:
--> 811                 raise KeyError(gpr)
    812         elif isinstance(gpr, Grouper) and gpr.key is not None:
    813             # Add key to exclusions

KeyError: 'h_clustering'
```

군집화 결과 시각화

In [35]:

```
plt.subplots(figsize=(20, 5))
plt.subplot(1,4,1)
sns.boxplot(x = 'h_clustering', y = 'sepal length', data= dataset)
plt.subplot(1,4,2)
sns.boxplot(x = 'h_clustering', y = 'sepal width', data= dataset)
plt.subplot(1,4,3)
sns.boxplot(x = 'h_clustering', y = 'petal length', data= dataset)
plt.subplot(1,4,4)
sns.boxplot(x = 'h_clustering', y = 'petal width', data= dataset)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-35-49390ab93351> in <module>
      1 plt.subplots(figsize=(20, 5))
      2 plt.subplot(1,4,1)
----> 3 sns.boxplot(x = 'h_clustering', y = 'sepal length', data= dataset)
      4 plt.subplot(1,4,2)
      5 sns.boxplot(x = 'h_clustering', y = 'sepal width', data= dataset)

~Wanaconda3WenvsWdata_miningWlibWsite-packagesWseabornW_decorators.py in i
nner_f(*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)})
---> 46         return f(**kwargs)
     47     return inner_f
     48

~Wanaconda3WenvsWdata_miningWlibWsite-packagesWseabornWcategorical.py in b
oxplot(x, y, hue, data, order, hue_order, orient, color, palette, saturati
on, width, dodge, fliersize, linewidth, whis, ax, **kwargs)
   2238 ):
   2239
-> 2240     plotter = _BoxPlotter(x, y, hue, data, order, hue_order,
   2241                           orient, color, palette, saturation,
   2242                           width, dodge, fliersize, linewidth)

~Wanaconda3WenvsWdata_miningWlibWsite-packagesWseabornWcategorical.py in _
_init__(self, x, y, hue, data, order, hue_order, orient, color, palette, s
aturation, width, dodge, fliersize, linewidth)
    404                  width, dodge, fliersize, linewidth):
    405
--> 406         self.establish_variables(x, y, hue, data, orient, order, hue_orde
r)
    407         self.establish_colors(color, palette, saturation)
    408

~Wanaconda3WenvsWdata_miningWlibWsite-packagesWseabornWcategorical.py in e
stablish_variables(self, x, y, hue, data, orient, order, hue_order, units)
    151                 if isinstance(var, str):
    152                     err = "Could not interpret input '{}'".format(var)
--> 153                     raise ValueError(err)
    154
    155             # Figure out the plotting orientation

ValueError: Could not interpret input 'h_clustering'
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [36]:

```
plt.subplots(figsize=(20, 5))
plt.subplot(1,3,1)
plot_cluster(h_clusters, column2index["sepal length"], column2index["sepal width"])
plt.subplot(1,3,2)
plot_cluster(h_clusters, column2index["petal length"], column2index["petal width"])
plt.subplot(1,3,3)
plot_cluster(h_clusters, column2index["sepal length"], column2index["petal length"])
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-36-94660df5af95> in <module>
      1 plt.subplots(figsize=(20, 5))
      2 plt.subplot(1,3,1)
----> 3 plot_cluster(h_clusters, column2index["sepal length"], column2index["sepal width"])
      4 plt.subplot(1,3,2)
      5 plot_cluster(h_clusters, column2index["petal length"], column2index["petal width"])

NameError: name 'plot_cluster' is not defined
```