Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

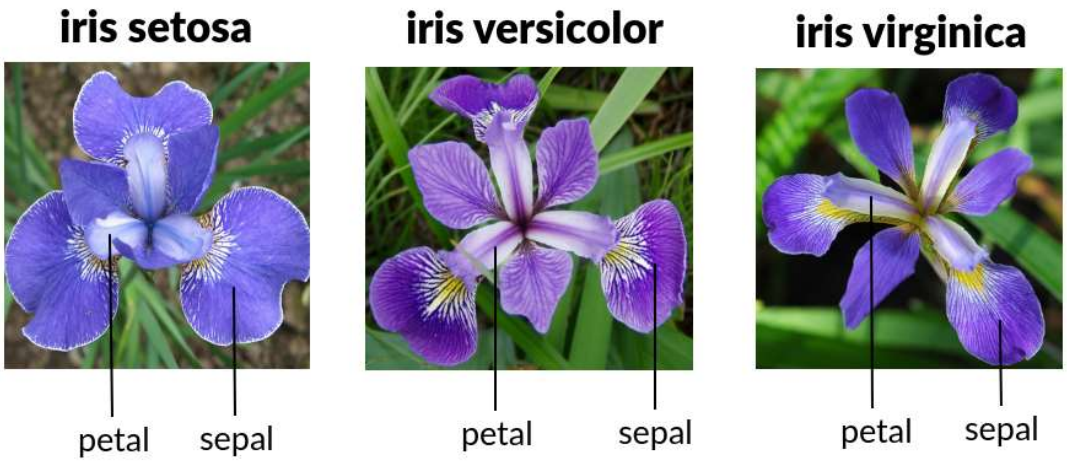
Pertemuan 8

Pertemuan 8 (delapan) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membersihkan Data dan Memvalidasi Data

DATA CLEANSING & Handling Missing Values

Value yang hilang serta tidak lengkap dari dataframe akan membuat analisis atau model prediksi yang dibuat menjadi tidak akurat dan mengakibatkan keputusan salah yang diambil. Terdapat beberapa cara untuk mengatasi data yang hilang/tidak lengkap tersebut.

Kali ini, kita akan menggunkan Dataset Iris yang kotor / terdapat nilai NaN dan outliers



Info dataset: Dataset ini berisi ukuran/measures 3 spesies iris

Pada Tugas Mandiri Pertemuan 8

silakan Anda kerjakan Latihan 1 s/d 20. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

Latihan (1)

Melakukan import library yang dibutuhkan

In [14]:
```
# import library pandas
import pandas as pd

# import library numpy
import numpy as np

# import library matplotlib
import matplotlib.pyplot as plt

# import library seaborn
import seaborn as sns

# me non aktifkan peringatan pada python dengan import warning -> 'ignore'
import warnings
warnings.filterwarnings("ignore")
```

Load Dataset

In [15]:
```
#Panggil file (load file bernama Iris_unclean.csv) dan simpan dalam dataframe Lal
df = pd.read_csv("Iris_unclean.csv")
df.head(10)
```

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	NaN	3.5	1.4	0.2	Iris-setosa
1	4.9	2000.0	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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	NaN	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	-1.5	0.2	Iris-setosa
8	4.4	1500.0	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Kegiatan yang akan kita lakukan:

- Melihat bentuk data (shape) dari data
- Langkah selanjutnya, harus tahu kolom mana yang terdapat data hilang dan berapa banyak dengan cara:

    1. menerapkan method .info() pada dataframe yang dapat diikuti dari kode berikut ini
    2. mengetahui berapa banyak nilai hilang dari tiap kolom di dataset tersebut dengan menerapkan chaining method pada dataframe yaitu .isna().sum().

- Cek data NaN, bila ada maka hapus/drop data NaN tsb
- Cek outliers, bila ada maka hapus/drop outliers tsb

Latihan (2)

Review Dataset

In [16]:
```
# menghasilkan jumlah baris dan jumlah kolom (bentuk data) pada data df dengan fu
print(df.dtypes)
```

```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [17]:
```
# fungsi describe() untuk mengetahui statistika data untuk data numeric seperti c
df.describe()
```

Out[17]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	148.000000	150.000000	150.000000	150.000000
mean	5.856757	26.348000	3.721333	1.198667
std	0.824964	203.117929	1.842364	0.763161
min	4.300000	2.000000	-1.500000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.375000	5.100000	1.800000
max	7.900000	2000.000000	6.900000	2.500000

In [18]:
```
# Informasi lebih detail mengenai struktur DataFrame dapat dilihat menggunakan fu
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  148 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [19]:
```
#cek nilai yang hilang / missing values di dalam data
df.dropna(inplace=True)
```

In [20]:
```
df.head()
```

Out[20]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	4.9	2000.0	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
5	5.4	3.9	1.7	0.4	Iris-setosa

Missing values adalah nilai yang tidak terdefinisi di dataset. Bentuknya beragam, bisa berupa blank cell, ataupun simbol-simbol tertentu seperti NaN (Not a Number), NA (Not Available), ?, -, dan sebagainya. Missing values dapat menjadi masalah dalam analisis data serta tentunya dapat mempengaruhi hasil modelling machine learning. Dari hasil diatas dataset tsb mengandung 2 data missing values pada kolom/field 'SepalLengthCm' dan beberapa outliers!

Periksa dan Cleansing setiap kolom pada data

dalam kasus ini hint nya adalah: hanya kolom/field **'SepalLengthCm' 'SepalWidthCm' 'PetalLengthCm'** yang bermasalah dan kita hanya akan berfokus cleansing pada kolom/field tsb

1. Kolom SepalLengthCm

Latihan (3)

periksa statistik data kolom SepalLengthCm

In [21]:
```
df.describe(include="all")
```

Out[21]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	148.000000	148.000000	148.000000	148.000000	148
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.856757	26.657432	3.752703	1.211486	NaN
std	0.824964	204.477337	1.834716	0.760194	NaN
min	4.300000	2.000000	-1.500000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.400000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	2000.000000	6.900000	2.500000	NaN

Latihan (4)

periksa jumlah nilai NaN pada kolom SepalLengthCm

In [24]:
```
print('Nilai NaN pada kolom SepalLengthCm berjumlah :', df['SepalLengthCm'].head(
```

```
Nilai NaN pada kolom SepalLengthCm berjumlah : 24.6
```

Latihan (5)

cetak index dari nilai NaN kolom SepalLengthCm dengan function np.where

In [25]:
```
df.dtypes.value_counts()
```

Out[25]:
```
float64    4
object     1
dtype: int64
```

In [26]:
```
index_nan =df.set_index('SepalLengthCm')
index_nan.head()
```

Out[26]:

SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
4.9	2000.0	1.4	0.2	Iris-setosa
4.7	3.2	-1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa

In [27]:
```
df['SepalLengthCm'].head(10)
```

Out[27]:
```
1     4.9
2     4.7
3     4.6
4     5.0
5     5.4
7     5.0
8     4.4
9     4.9
10    5.4
11    4.8
Name: SepalLengthCm, dtype: float64
```

In [28]:
```
df.loc[11]
```

Out[28]:
```
SepalLengthCm            4.8
SepalWidthCm             3.4
PetalLengthCm            1.6
PetalWidthCm             0.2
Species          Iris-setosa
Name: 11, dtype: object
```

Latihan (6)

1. Cetak ukuran/dimensi dari dataframe
2. Drop baris jika ada satu saja data yang missing dan ukuran/dimensi dari dataframe setelah di drop

In [29]:
```
# Drop baris jika ada satu saja data yang missing dengan function dropna() dan ce
df.dropna()
```

Out[29]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	4.9	2000.0	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
5	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

148 rows × 5 columns

2. Kolom SepalWidthCm

Latihan (7)

periksa statistik data kolom SepalWidthCm

In [30]:
```
df.mean()
```

Out[30]:
```
SepalLengthCm     5.856757
SepalWidthCm     26.657432
PetalLengthCm     3.752703
PetalWidthCm      1.211486
dtype: float64
```

In [31]:
```
df.max()['SepalWidthCm']
```

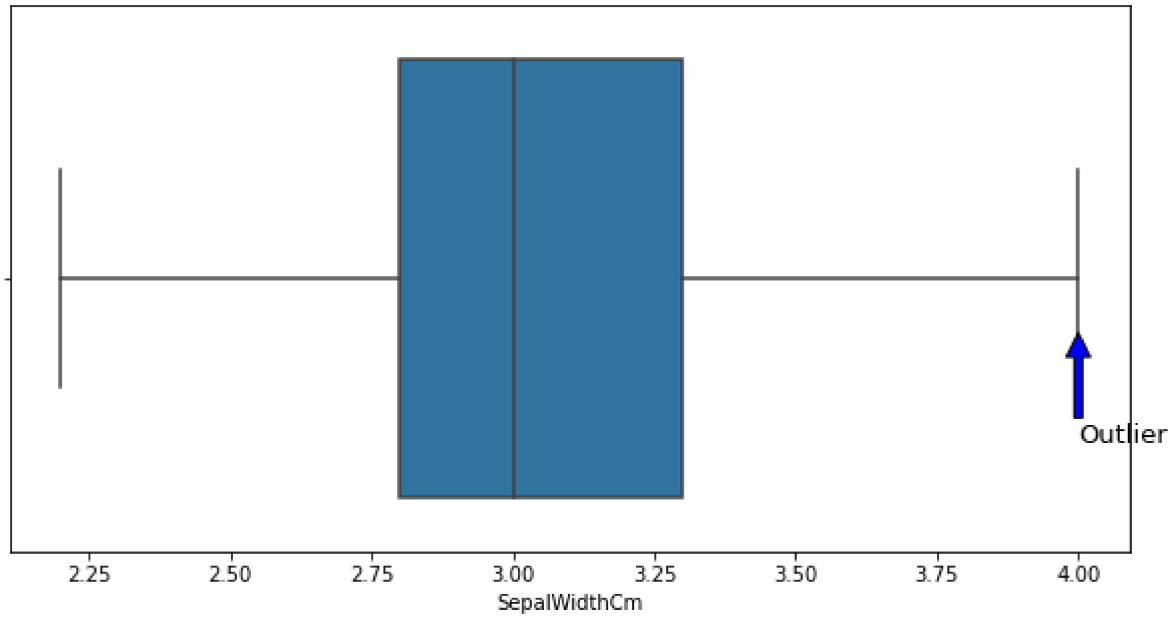
Out[31]:
```
2000.0
```

Dari data diatas terlihat pada terdapat kejanggalan pada nilai max yaitu 2000cm, sedangkan Sepal Width/ lebar Kelopak bunga nampaknya tidak masuk akal bila berukuran hingga 2000cm. Sehingga dapat dipastikan ini merupakan outliers

Latihan (8)

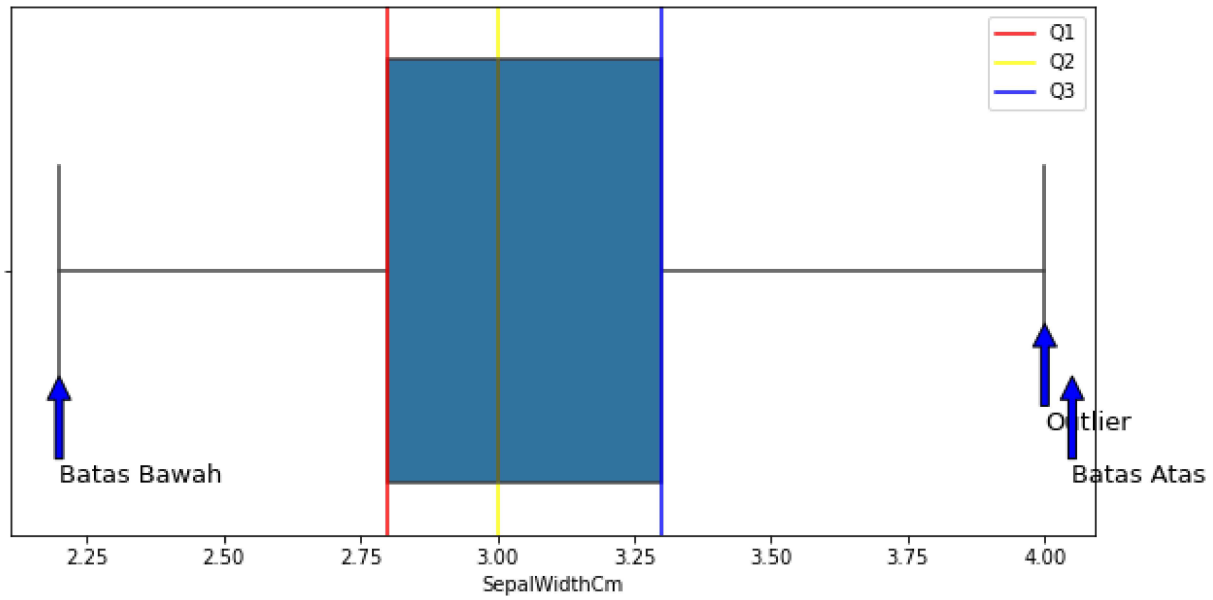
mendeteksi outlier dengan menggunakan boxplot pada kolom SepalWidthCm

In [57]:
```
plt.figure(figsize = (10, 5))
sns.boxplot(df['SepalWidthCm'])
plt.annotate('Outlier', (df['SepalWidthCm'].describe()['max'],0.1), xytext = (df[
             arrowprops = dict(facecolor = 'blue'), fontsize = 13 )
IQR = df['SepalWidthCm'].describe()['75%'] - df['SepalWidthCm'].describe()['25%']
```



In [58]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize = (10, 5))
sns.boxplot(df['SepalWidthCm'])
plt.axvline(df['SepalWidthCm'].describe()['25%'], color = 'red', label = 'Q1')
plt.axvline(df['SepalWidthCm'].describe()['50%'], color = 'yellow', label = 'Q2')
plt.axvline(df['SepalWidthCm'].describe()['75%'], color = 'blue', label = 'Q3')
plt.annotate('Outlier', (df['SepalWidthCm'].describe()['max'],0.1), xytext = (df[
             arrowprops = dict(facecolor = 'blue'), fontsize = 13 )
IQR = df['SepalWidthCm'].describe()['75%'] - df['SepalWidthCm'].describe()['25%']
plt.annotate('Batas Atas', (df['SepalWidthCm'].describe()['75%'] + 1.5*IQR, 0.2),
             xytext = (df['SepalWidthCm'].describe()['75%'] + 1.5*IQR, 0.4),
             arrowprops = dict(facecolor = 'blue'), fontsize = 13 )
plt.annotate('Batas Bawah', (df['SepalWidthCm'].describe()['min'], 0.2),
             xytext = (df['SepalWidthCm'].describe()['min'], 0.4),
             arrowprops = dict(facecolor = 'blue'), fontsize = 13 )
plt.legend()
```

Out[58]:
```
<matplotlib.legend.Legend at 0x2a3562f5c10>
```



Latihan (9-10)

membuat fungsi melihat data outlier dengan rumus IQR = Q3-Q1

In [59]:
```
def detect_outliers(df, x):
    Q1 = df[x].describe['25%']
    Q3 = df[x].describe['75%']
    IQR = Q3-Q1
    return df[(df[x] < Q1-1.5*IQR) | (df[x] > Q3+1.5*IQR)]
```

In [60]:
```
df.shape
```

Out[60]:
```
(142, 5)
```

Latihan (11)

hapus data outlier dari kolom SepalWidthCm

In [51]:
```
df = df.drop((df[df['SepalWidthCm']>4]).index, axis=0)
```

In [52]:
```
df = df.drop((df[df['SepalWidthCm']<2.1]).index, axis=0)
```

In [63]:
```
help(df.fillna)
```

```
Help on method fillna in module pandas.core.frame:

fillna(value=None, method=None, axis=None, inplace=False, limit=None, downca
st=None) -> 'Optional[DataFrame]' method of pandas.core.frame.DataFrame inst
ance
    Fill NA/NaN values using the specified method.

    Parameters
    ----------
    value : scalar, dict, Series, or DataFrame
        Value to use to fill holes (e.g. 0), alternately a
        dict/Series/DataFrame of values specifying which value to use for
        each index (for a Series) or column (for a DataFrame).  Values not
        in the dict/Series/DataFrame will not be filled. This value cannot
        be a list.
    method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None
        Method to use for filling holes in reindexed Series
        pad / ffill: propagate last valid observation forward to next valid
        backfill / bfill: use next valid observation to fill gap.
    axis : {0 or 'index', 1 or 'columns'}
        Axis along which to fill missing values.
    inplace : bool, default False
        If True, fill in-place. Note: this will modify any
        other views on this object (e.g., a no-copy slice for a column in a
        DataFrame).
    limit : int, default None
        If method is specified, this is the maximum number of consecutive
        NaN values to forward/backward fill. In other words, if there is
        a gap with more than this number of consecutive NaNs, it will only
        be partially filled. If method is not specified, this is the
        maximum number of entries along the entire axis where NaNs will be
        filled. Must be greater than 0 if not None.
    downcast : dict, default is None
        A dict of item->dtype of what to downcast if possible,
        or the string 'infer' which will try to downcast to an appropriate
        equal type (e.g. float64 to int64 if possible).

    Returns
    -------
    DataFrame or None
        Object with missing values filled or None if ``inplace=True``.

    See Also
    --------
    interpolate : Fill NaN values using interpolation.
    reindex : Conform object to new index.
    asfreq : Convert TimeSeries to specified frequency.

    Examples
    --------
    >>> df = pd.DataFrame([[np.nan, 2, np.nan, 0],
    ...                    [3, 4, np.nan, 1],
    ...                    [np.nan, np.nan, np.nan, 5],
    ...                    [np.nan, 3, np.nan, 4]],
```

```
...                    columns=list('ABCD'))
>>> df
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  NaN NaN  5
3  NaN  3.0 NaN  4

Replace all NaN elements with 0s.

>>> df.fillna(0)
    A   B   C   D
0   0.0 2.0 0.0 0
1   3.0 4.0 0.0 1
2   0.0 0.0 0.0 5
3   0.0 3.0 0.0 4

We can also propagate non-null values forward or backward.

>>> df.fillna(method='ffill')
    A   B   C   D
0   NaN 2.0 NaN 0
1   3.0 4.0 NaN 1
2   3.0 4.0 NaN 5
3   3.0 3.0 NaN 4

Replace all NaN elements in column 'A', 'B', 'C', and 'D', with 0, 1,
2, and 3 respectively.

>>> values = {'A': 0, 'B': 1, 'C': 2, 'D': 3}
>>> df.fillna(value=values)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 2.0 1
2   0.0 1.0 2.0 5
3   0.0 3.0 2.0 4

Only replace the first NaN element.

>>> df.fillna(value=values, limit=1)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 NaN 1
2   NaN 1.0 NaN 5
3   NaN 3.0 NaN 4
```

In [64]:
```
df['SepalWidthCm'].fillna(df['SepalWidthCm'].median(), inplace = True)
```

In [65]:
```
df['SepalWidthCm'].isnull().sum()
```

Out[65]:
```
0
```

Latihan (12)

cek ulang outliers dengan fungsi yang telah dibuat

In [53]:
```
df.head()
```

Out[53]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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
5	5.4	3.9	1.7	0.4	Iris-setosa
7	5.0	3.4	-1.5	0.2	Iris-setosa

In [66]:
```
help(df.fillna)
```

```
Help on method fillna in module pandas.core.frame:

fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=
None) -> 'Optional[DataFrame]' method of pandas.core.frame.DataFrame instance
    Fill NA/NaN values using the specified method.

    Parameters
    ----------
    value : scalar, dict, Series, or DataFrame
        Value to use to fill holes (e.g. 0), alternately a
        dict/Series/DataFrame of values specifying which value to use for
        each index (for a Series) or column (for a DataFrame).  Values not
        in the dict/Series/DataFrame will not be filled. This value cannot
        be a list.
    method : {'backfill', 'bfill', 'pad', 'ffill', None}, default None
        Method to use for filling holes in reindexed Series
        pad / ffill: propagate last valid observation forward to next valid
        backfill / bfill: use next valid observation to fill gap.
    axis : {0 or 'index', 1 or 'columns'}
        Axis along which to fill missing values.
    inplace : bool, default False
        If True, fill in-place. Note: this will modify any
        other views on this object (e.g., a no-copy slice for a column in a
        DataFrame).
    limit : int, default None
        If method is specified, this is the maximum number of consecutive
        NaN values to forward/backward fill. In other words, if there is
        a gap with more than this number of consecutive NaNs, it will only
        be partially filled. If method is not specified, this is the
        maximum number of entries along the entire axis where NaNs will be
        filled. Must be greater than 0 if not None.
    downcast : dict, default is None
        A dict of item->dtype of what to downcast if possible,
        or the string 'infer' which will try to downcast to an appropriate
        equal type (e.g. float64 to int64 if possible).

    Returns
    -------
    DataFrame or None
        Object with missing values filled or None if ``inplace=True``.

    See Also
    --------
    interpolate : Fill NaN values using interpolation.
    reindex : Conform object to new index.
    asfreq : Convert TimeSeries to specified frequency.

    Examples
    --------
    >>> df = pd.DataFrame([[np.nan, 2, np.nan, 0],
    ...                    [3, 4, np.nan, 1],
    ...                    [np.nan, np.nan, np.nan, 5],
    ...                    [np.nan, 3, np.nan, 4]],
    ...                   columns=list('ABCD'))
```

```
>>> df
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  NaN NaN  5
3  NaN  3.0 NaN  4

Replace all NaN elements with 0s.

>>> df.fillna(0)
    A   B   C   D
0   0.0 2.0 0.0 0
1   3.0 4.0 0.0 1
2   0.0 0.0 0.0 5
3   0.0 3.0 0.0 4

We can also propagate non-null values forward or backward.

>>> df.fillna(method='ffill')
    A   B   C   D
0   NaN 2.0 NaN 0
1   3.0 4.0 NaN 1
2   3.0 4.0 NaN 5
3   3.0 3.0 NaN 4

Replace all NaN elements in column 'A', 'B', 'C', and 'D', with 0, 1,
2, and 3 respectively.

>>> values = {'A': 0, 'B': 1, 'C': 2, 'D': 3}
>>> df.fillna(value=values)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 2.0 1
2   0.0 1.0 2.0 5
3   0.0 3.0 2.0 4

Only replace the first NaN element.

>>> df.fillna(value=values, limit=1)
    A   B   C   D
0   0.0 2.0 2.0 0
1   3.0 4.0 NaN 1
2   NaN 1.0 NaN 5
3   NaN 3.0 NaN 4
```

In [67]:
```
df['PetalWidthCm'].fillna(df['PetalWidthCm'].median(), inplace = True)
```

In [68]:
```
df['PetalWidthCm'].isnull().sum()
```

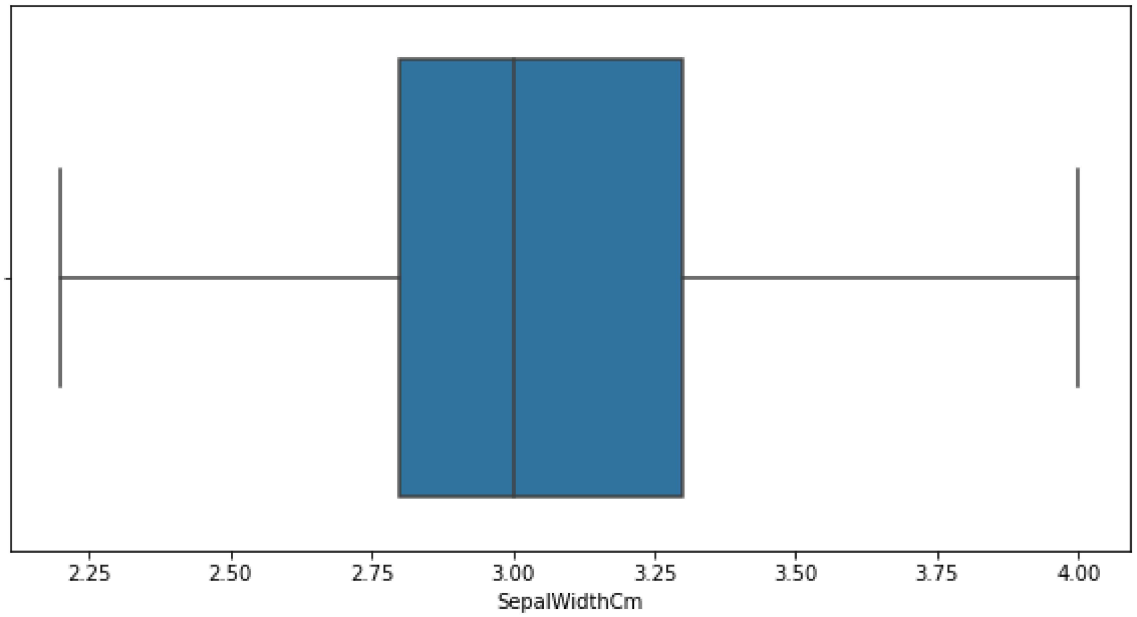
Out[68]:
```
0
```

Latihan (13)

cek ulang outliers dengan boxplot

In [37]:
```
plt.figure(figsize = (10, 5))
sns.boxplot(df['SepalWidthCm'])
```

Out[37]:
```
<AxesSubplot:xlabel='SepalWidthCm'>
```



3. Kolom PetalLengthCm

Latihan (14)

periksa statistik data kolom SepalLengthCm

In [70]:
```
df['PetalLengthCm'].head()
```

Out[70]:
```
2   -1.3
3    1.5
4    1.4
5    1.7
7   -1.5
Name: PetalLengthCm, dtype: float64
```

Dari data diatas terlihat pada terdapat kejanggalan pada nilai min yaitu bernilai minus, sedangkan Petal Length/ panjang Kelopak bunga nampaknya tidak masuk akal bila berukuran minus. Sehingga dapat dipastikan ini merupakan outliers

Latihan (15)

periksa data bernilai minus pada kolom PetalLengthCm

In [74]:
```
df[df['PetalLengthCm'] < 1]
```

Out[74]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	4.7	3.2	-1.3	0.2	Iris-setosa
7	5.0	3.4	-1.5	0.2	Iris-setosa

Latihan (16)

hapus data bernilai minus / outlier kolom PetalLengthCm

In [75]:
```
df = df.drop((df[df['PetalLengthCm']<1]).index, axis=0)
```

Latihan (17)

cek ulang outliers dengan fungsi yang telah dibuat

In [76]:
```
df[df['PetalLengthCm'] < 1]
```

Out[76]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

CEK DATA SETELAH PROSES CLEANSING

Latihan (18)

Melihat nomor index beserta tipe datanya dengan function info()

In [77]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140 entries, 3 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  140 non-null    float64
 1   SepalWidthCm   140 non-null    float64
 2   PetalLengthCm  140 non-null    float64
 3   PetalWidthCm   140 non-null    float64
 4   Species        140 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.6+ KB
```

Latihan (19)

cek ulang nilai yang hilang / missing values di dalam data setelah proses cleansing

In [81]:
```
df.isnull().sum()
```

Out[81]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Latihan (20)

Tampilkan 10 baris dataframe setelah proses cleansing

In [80]:
```
df.head(10)
```

Out[80]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa

DATA SUDAH SIAP UNTUK KETAHAP SELANJUTNYA YAITU MODELLING :)