Assignment No.1

Rollno: TCO21F035

```
import pandas as pd
df=pd.read_csv("./Downloads/Uber Request Data.csv") df
```

```
      Request id Pickup point  Driver id              Status  \
0            619      Airport        1.0     Trip Completed
1            867      Airport        1.0     Trip Completed
2           1807         City        1.0     Trip Completed
3           2532      Airport        1.0     Trip Completed
4           3112         City        1.0     Trip Completed
...          ...          ...        ...                ...
6740        6745         City        NaN  No Cars Available
6741        6752      Airport        NaN  No Cars Available
6742        6751         City        NaN  No Cars Available
6743        6754         City        NaN  No Cars Available
6744        6753      Airport        NaN  No Cars Available

      Request timestamp       Drop timestamp
0          11/7/2016 11:51      11/7/2016 13:00
1          11/7/2016 17:57      11/7/2016 18:47
2           12/7/2016 9:17       12/7/2016 9:58
3          12/7/2016 21:08      12/7/2016 22:03
4       13-07-2016 08:33:16  13-07-2016 09:25:47
...                     ...                  ...
6740    15-07-2016 23:49:03                  NaN
6741    15-07-2016 23:50:05                  NaN
6742    15-07-2016 23:52:06                  NaN
6743    15-07-2016 23:54:39                  NaN
6744    15-07-2016 23:55:03                  NaN

[6745 rows x 6 columns]
```

```
df.shape
```

```
(6745, 6)
```

```
df
```

```
      Request id Pickup point  Driver id              Status  \
0            619      Airport        1.0     Trip Completed
1            867      Airport        1.0     Trip Completed
2           1807         City        1.0     Trip Completed
3           2532      Airport        1.0     Trip Completed
4           3112         City        1.0     Trip Completed
...          ...          ...        ...                ...
6740        6745         City        NaN  No Cars Available
6741        6752      Airport        NaN  No Cars Available
6742        6751         City        NaN  No Cars Available
6743        6754         City        NaN  No Cars Available
```

```
6744        6753       Airport          NaN  No Cars Available

        Request timestamp       Drop timestamp
0          11/7/2016 11:51      11/7/2016 13:00
1          11/7/2016 17:57      11/7/2016 18:47
2           12/7/2016 9:17       12/7/2016 9:58
3          12/7/2016 21:08      12/7/2016 22:03
4       13-07-2016 08:33:16  13-07-2016 09:25:47
...                     ...                  ...
6740  15-07-2016 23:49:03                  NaN
6741  15-07-2016 23:50:05                  NaN
6742  15-07-2016 23:52:06                  NaN
6743  15-07-2016 23:54:39                  NaN
6744  15-07-2016 23:55:03                  NaN

[6745 rows x 6 columns]
```

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6745 entries, 0 to 6744
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Request id         6745 non-null   int64
 1   Pickup point       6745 non-null   object
 2   Driver id          4095 non-null   float64
 3   Status             6745 non-null   object
 4   Request timestamp  6745 non-null   object
 5   Drop timestamp     2831 non-null   object
dtypes: float64(1), int64(1), object(4)
memory usage: 316.3+ KB
```

df.head()

```
   Request id Pickup point  Driver id          Status     Request
timestamp \
0         619      Airport         1.0  Trip Completed      11/7/2016
11:51
1         867      Airport         1.0  Trip Completed      11/7/2016
17:57
2        1807         City         1.0  Trip Completed       12/7/2016
9:17
3        2532      Airport         1.0  Trip Completed      12/7/2016
21:08
4        3112         City         1.0  Trip Completed  13-07-2016
08:33:16

         Drop timestamp
0       11/7/2016 13:00
```

```
1       11/7/2016 18:47
2        12/7/2016 9:58
3       12/7/2016 22:03
4  13-07-2016 09:25:47
```

```
df.head(10)
```

```
   Request id Pickup point   Driver id          Status     Request
timestamp  \
0         619      Airport         1.0  Trip Completed      11/7/2016
11:51
1         867      Airport         1.0  Trip Completed      11/7/2016
17:57
2        1807         City         1.0  Trip Completed       12/7/2016
9:17
3        2532      Airport         1.0  Trip Completed      12/7/2016
21:08
4        3112         City         1.0  Trip Completed  13-07-2016
08:33:16
5        3879      Airport         1.0  Trip Completed  13-07-2016
21:57:28
6        4270      Airport         1.0  Trip Completed  14-07-2016
06:15:32
7        5510      Airport         1.0  Trip Completed  15-07-2016
05:11:52
8        6248         City         1.0  Trip Completed  15-07-2016
17:57:27
9         267         City         2.0  Trip Completed      11/7/2016
6:46

         Drop timestamp
0       11/7/2016 13:00
1       11/7/2016 18:47
2        12/7/2016 9:58
3       12/7/2016 22:03
4  13-07-2016 09:25:47
5  13-07-2016 22:28:59
6  14-07-2016 07:13:15
7  15-07-2016 06:07:52
8  15-07-2016 18:50:51
9       11/7/2016 7:25
```

```
df.tail()
```

```
      Request id Pickup point  Driver id              Status  \
6740        6745         City        NaN  No Cars Available
6741        6752      Airport        NaN  No Cars Available
6742        6751         City        NaN  No Cars Available
6743        6754         City        NaN  No Cars Available
6744        6753      Airport        NaN  No Cars Available
```

```
        Request timestamp Drop  timestamp
6740   15-07-2016 23:49:03              NaN
6741   15-07-2016 23:50:05              NaN
6742   15-07-2016 23:52:06              NaN
6743   15-07-2016 23:54:39              NaN
6744   15-07-2016 23:55:03              NaN
```

```
df.tail(10)
```

```
      Request id Pickup point  Driver id              Status  \
6735        6737      Airport        NaN  No Cars Available
6736        6744      Airport        NaN  No Cars Available
6737        6740         City        NaN  No Cars Available
6738        6746         City        NaN  No Cars Available
6739        6739         City        NaN  No Cars Available
6740        6745         City        NaN  No Cars Available
6741        6752      Airport        NaN  No Cars Available
6742        6751         City        NaN  No Cars Available
6743        6754         City        NaN  No Cars Available
6744        6753      Airport        NaN  No Cars Available
```

```
        Request timestamp Drop  timestamp
6735   15-07-2016 23:39:15              NaN
6736   15-07-2016 23:42:51              NaN
6737   15-07-2016 23:43:54              NaN
6738   15-07-2016 23:46:03              NaN
6739   15-07-2016 23:46:20              NaN
6740   15-07-2016 23:49:03              NaN
6741   15-07-2016 23:50:05              NaN
6742   15-07-2016 23:52:06              NaN
6743   15-07-2016 23:54:39              NaN
6744   15-07-2016 23:55:03              NaN
```

```
df.index
```

```
RangeIndex(start=0, stop=6745, step=1)
```

```
df.columns
```

```
Index(['Request id', 'Pickup point', 'Driver id', 'Status',
       'Request timestamp', 'Drop timestamp'],
      dtype='object')
```

```
df.shape
```

```
(6745, 6)
```

```
df.dtypes
```

```
Request id          int64
Pickup point       object
```

```
Driver id              float64
Status                  object
Request timestamp       object
Drop timestamp          object
dtype: object
```

```
df["Request id" ].astype("float64")
```

```
0          619.0
1          867.0
2         1807.0
3         2532.0
4         3112.0
           ...
6740      6745.0
6741      6752.0
6742      6751.0
6743      6754.0
6744      6753.0
Name: Request id, Length: 6745, dtype: float64
```

```
df.dtypes
```

```
Request id              int64
Pickup point           object
Driver id             float64
Status                 object
Request timestamp      object
Drop timestamp         object
dtype: object
```

```
df["Request id"]=df["Request id"].astype("float64")
df.dtypes
```

```
Request id             float64
Pickup point            object
Driver id              float64
Status                  object
Request timestamp       object
Drop timestamp          object
dtype: object
```

```
df.columns.values
```

```
array(['Request id', 'Pickup point', 'Driver id', 'Status',
       'Request timestamp', 'Drop timestamp'], dtype=object)
```

```
df.describe()
```

```
         Request id     Driver id
count   6745.000000  4095.000000
```

```
mean    3384.644922    149.501343
std     1955.099667     86.051994
min        1.000000      1.000000
25%     1691.000000     75.000000
50%     3387.000000    149.000000
75%     5080.000000    224.000000
max     6766.000000    300.000000
```

df.isnull()

```
      Request id  Pickup point  Driver id   Status   Request
timestamp \
0          False         False       False   False              False

1          False         False       False   False              False

2          False         False       False   False              False

3          False         False       False   False              False

4          False         False       False   False              False

...          ...           ...         ...     ...                ...

6740       False         False        True   False              False

6741       False         False        True   False              False

6742       False         False        True   False              False

6743       False         False        True   False              False

6744       False         False        True   False              False


      Drop timestamp
0              False
1              False
2              False
3              False
4              False
...              ...
6740            True
6741            True
6742            True
6743            True
6744            True

[6745 rows x 6 columns]
```

df.notnull()

```
      Request id  Pickup point  Driver id   Status  Request
timestamp \
0            True          True       True    True                True

1            True          True       True    True                True

2            True          True       True    True                True

3            True          True       True    True                True

4            True          True       True    True                True

...           ...           ...        ...     ...                 ...

6740         True          True      False    True                True

6741         True          True      False    True                True

6742         True          True      False    True                True

6743         True          True      False    True                True

6744         True          True      False    True                True


      Drop  timestamp
0                True
1                True
2                True
3                True
4                True
...               ...
6740            False
6741            False
6742            False
6743            False
6744            False

[6745 rows x 6 columns]
```

df.isna()

```
      Request id  Pickup point  Driver id   Status  Request
timestamp \
0           False         False      False   False               False

1           False         False      False   False               False

2           False         False      False   False               False

3           False         False      False   False               False
```

```
4           False          False       False    False                False

...           ...            ...         ...      ...                  ...

6740        False          False        True    False                False

6741        False          False        True    False                False

6742        False          False        True    False                False

6743        False          False        True    False                False

6744        False          False        True    False                False


      Drop  timestamp
0                False
1                False
2                False
3                False
4                False
...                ...
6740              True
6741              True
6742              True
6743              True
6744              True

[6745 rows x 6 columns]
```

df.notna()

```
      Request id  Pickup point  Driver id   Status   Request
timestamp \
0             True          True        True     True                 True

1             True          True        True     True                 True

2             True          True        True     True                 True

3             True          True        True     True                 True

4             True          True        True     True                 True

...            ...           ...         ...      ...                  ...

6740          True          True       False     True                 True

6741          True          True       False     True                 True
```

```
6742          True             True        False      True                  True

6743          True             True        False      True                  True

6744          True             True        False      True                  True
```

```
      Drop timestamp
0                True
1                True
2                True
3                True
4                True
...               ...
6740            False
6741            False
6742            False
6743            False
6744            False

[6745 rows x 6 columns]
```

```
df.isnull().sum()
```

```
Request id             0
Pickup point           0
Driver id           2650
Status                 0
Request timestamp      0
Drop timestamp      3914
dtype: int64
```

```
df.isnull().any()
```

```
Request id           False
Pickup point         False
Driver id             True
Status               False
Request timestamp    False
Drop timestamp        True
dtype: bool
```

```
df.iloc[3]
```

```
Request id                    2532.0
Pickup point                 Airport
Driver id                        1.0
Status                Trip Completed
Request timestamp  12/7/2016 21:08
Drop timestamp     12/7/2016 22:03
Name: 3, dtype: object
```

```
df[0:3]
```

	Request id	Pickup point	Driver id	Status	Request timestamp
0	619.0	Airport	1.0	Trip Completed	11/7/2016 11:51
1	867.0	Airport	1.0	Trip Completed	11/7/2016 17:57
2	1807.0	City	1.0	Trip Completed	12/7/2016 9:17

	Drop timestamp
0	11/7/2016 13:00
1	11/7/2016 18:47
2	12/7/2016 9:58

```
df.describe(include='all')
```

	Request id	Pickup point	Drive	Status
count	6745.000000	6745	4095.000	6745
unique	NaN	2		3
top	NaN	City		Trip Completed
freq	NaN	3507		2831
mean	3384.644922	NaN	149.501	NaN
std	1955.099667	NaN	86.051	NaN
min	1.000000	NaN	1.000	NaN
25%	1691.000000	NaN	75.000	NaN
50%	3387.000000	NaN	149.000	NaN
75%	5080.000000	NaN	224.000	NaN
max	6766.000000	NaN	300.000	NaN

	Request timestamp	Drop timestamp
count	6745	2831
unique	5618	2598
top	11/7/2016 19:02	11/7/2016 13:00
freq	6	4
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25%	NaN	NaN
50%	NaN	NaN
75%	NaN	NaN
max	NaN	NaN

```
df.describe()
```

	Request id	Driver id
count	6745.000000	4095.000000
mean	3384.644922	149.501343
std	1955.099667	86.051994
min	1.000000	1.000000

```
25%    1691.000000     75.000000
50%    3387.000000    149.000000
75%    5080.000000    224.000000
max    6766.000000    300.000000
```

```
df['Request id']
```

```
0         619.0
1         867.0
2        1807.0
3        2532.0
4        3112.0
         ...
6740     6745.0
6741     6752.0
6742     6751.0
6743     6754.0
6744     6753.0
Name: Request id, Length: 6745, dtype: float64
```

```
df.sort_values(by="Request  id")
```

```
      Request id Pickup point  Driver id              Status  \
2700         1.0      Airport      285.0    Trip Completed
4098         2.0      Airport        NaN  No Cars Available
776          3.0      Airport       80.0    Trip Completed
4101         4.0         City        NaN  No Cars Available
2506         5.0      Airport      264.0    Trip Completed
...          ...          ...        ...                ...
2534      6762.0      Airport      267.0    Trip Completed
2137      6763.0         City      224.0    Trip Completed
2324      6764.0         City      243.0    Trip Completed
6165      6765.0      Airport        NaN  No Cars Available
1042      6766.0         City      108.0    Trip Completed

      Request timestamp       Drop timestamp
2700      11/7/2016 0:20       11/7/2016 0:51
4098      11/7/2016 0:23                  NaN
776       11/7/2016 0:24       11/7/2016 1:31
4101      11/7/2016 0:37                  NaN
2506      11/7/2016 0:36       11/7/2016 1:35
...                  ...                  ...
2534  15-07-2016 00:07:29  15-07-2016 00:52:50
2137  15-07-2016 00:04:44  15-07-2016 01:06:42
2324  15-07-2016 00:06:12  15-07-2016 01:17:53
6165  15-07-2016 00:09:09                  NaN
1042  15-07-2016 00:06:56  15-07-2016 01:10:34

[6745 rows x 6 columns]
```

```
df.sort_values(by="Driver id")
```

```
      Request id Pickup point  Driver id              Status  \
0          619.0      Airport        1.0     Trip Completed
2833      5202.0      Airport        1.0           Cancelled
2834      5927.0         City        1.0           Cancelled
2832      4805.0         City        1.0           Cancelled
8         6248.0         City        1.0     Trip Completed
...          ...          ...        ...                ...
6740      6745.0         City        NaN  No Cars Available
6741      6752.0      Airport        NaN  No Cars Available
6742      6751.0         City        NaN  No Cars Available
6743      6754.0         City        NaN  No Cars Available
6744      6753.0      Airport        NaN  No Cars Available

        Request timestamp       Drop timestamp
0          11/7/2016 11:51      11/7/2016 13:00
2833   14-07-2016 20:51:37                  NaN
2834   15-07-2016 10:12:40                  NaN
2832   14-07-2016 17:07:58                  NaN
8      15-07-2016 17:57:27  15-07-2016 18:50:51
...                    ...                  ...
6740   15-07-2016 23:49:03                  NaN
6741   15-07-2016 23:50:05                  NaN
6742   15-07-2016 23:52:06                  NaN
6743   15-07-2016 23:54:39                  NaN
6744   15-07-2016 23:55:03                  NaN

[6745 rows x 6 columns]
```

```
df.isnull().sum()
```

```
Request id                0
Pickup point              0
Driver id              2650
Status                    0
Request timestamp         0
Drop timestamp         3914
dtype: int64
```

```
2650+3914
```

```
6564
```

```
df.isnull().sum().sum()
```

```
6564
```

```
df["Request id"].isnull().sum()
```

```
0
```

```
df["Driver id"].isnull().sum()
```

2650

```
df=pd.read_csv("./Downloads/IRIS.csv")
```

```
df
```

```
     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
..            ...          ...           ...          ...
...
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

[150 rows x 5 columns]
```

```
from sklearn import preprocessing
```

```
min_max_scaler=preprocessing.MinMaxScaler()
```

```
x=df.iloc[:,:4]
```

```
x_scaled = min_max_scaler.fit_transform(x)
```

```
df_normalized = pd.DataFrame(x_scaled)
```

```
df_normalized
```

```
            0         1         2         3
0    0.222222  0.625000  0.067797  0.041667
1    0.166667  0.416667  0.067797  0.041667
2    0.111111  0.500000  0.050847  0.041667
3    0.083333  0.458333  0.084746  0.041667
4    0.194444  0.666667  0.067797  0.041667
..        ...       ...       ...       ...
```

```
145   0.666667   0.416667   0.711864   0.916667
146   0.555556   0.208333   0.677966   0.750000
147   0.611111   0.416667   0.711864   0.791667
148   0.527778   0.583333   0.745763   0.916667
149   0.444444   0.416667   0.694915   0.708333

[150 rows x 4 columns]
```

```
df
```

```
     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
..            ...          ...           ...          ...
...
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

[150 rows x 5 columns]
```

```
label_encoder = preprocessing.LabelEncoder()
```

```
df=pd.read_csv("./Downloads/IRIS.csv")
df
```

```
     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
..               ...           ...            ...           ...
...
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

[150 rows x 5 columns]
```

Assignment No.2

Rollno: TC021F035

```
import pandas as pd
import numpy as np

dict1={"math_score":[60,70,np.nan,80,56,72,77,68,np.nan,50],
       "Reading_score":
[200,80,np.nan,85,83,300,90,95,92,94],"Writing_score":
[60,80,np.nan,65,63,60,70,71,75,82],
       "Placement_score":
[75,79,np.nan,80,85,84,90,95,97,82],"Join_Date":
[2018,2019,np.nan,2020,2021,2022,2019,2022,2023,2024],"Region":
[np.nan,"Buldhana","Kothrud","Baner","Nagpur","Dadar",np.nan,"Kothrud"
,"Baner","Warje"]
       ,"Gender":
["Male","Female","Female","Male","Male","Male","Female","Male","Male",
"Female"]}

dict1
```

```
{'math_score': [60, 70, nan, 80, 56, 72, 77, 68, nan, 50],
 'Reading_score': [200, 80, nan, 85, 83, 300, 90, 95, 92, 94],
 'Writing_score': [60, 80, nan, 65, 63, 60, 70, 71, 75, 82],
 'Placement_score': [75, 79, nan, 80, 85, 84, 90, 95, 97, 82],
 'Join_Date': [2018, 2019, nan, 2020, 2021, 2022, 2019, 2022, 2023,
2024],
 'Region': [nan,
  'Buldhana',
  'Kothrud',
  'Baner',
  'Nagpur',
  'Dadar',
  nan,
  'Kothrud',
  'Baner',
  'Warje'],
 'Gender': ['Male',
  'Female',
  'Female',
  'Male',
  'Male',
  'Male',
  'Female',
  'Male',
  'Male',
  'Female']}
```

```
df=pd.DataFrame(dict1)
df
```

```
   math_score  Reading_score  Writing_score   Placement_score
Join_Date   \
```

```
0          60.0          200.0          60.0          75.0
2018.0
1          70.0           80.0          80.0          79.0
2019.0
2           NaN            NaN           NaN           NaN
NaN
3          80.0           85.0          65.0          80.0
2020.0
4          56.0           83.0          63.0          85.0
2021.0
5          72.0          300.0          60.0          84.0
2022.0
6          77.0           90.0          70.0          90.0
2019.0
7          68.0           95.0          71.0          95.0
2022.0
8           NaN           92.0          75.0          97.0
2023.0
9          50.0           94.0          82.0          82.0
2024.0
```

```
     Region  Gender
0       NaN    Male
1  Buldhana  Female
2   Kothrud  Female
3     Baner    Male
4    Nagpur    Male
5     Dadar    Male
6       NaN  Female
7   Kothrud    Male
8     Baner    Male
9     Warje  Female
```

df.isnull()

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0       False          False          False            False
False
1       False          False          False            False
False
2        True           True           True             True
True
3       False          False          False            False
False
4       False          False          False            False
False
5       False          False          False            False
False
6       False          False          False            False
```

```
False
7        False           False           False           False
False
8         True           False           False           False
False
9        False           False           False           False
False
```

```
   Region  Gender
0    True   False
1   False   False
2   False   False
3   False   False
4   False   False
5   False   False
6    True   False
7   False   False
8   False   False
9   False   False
```

df.notnull()

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date   \
0        True           True           True           True
True
1        True           True           True           True
True
2       False          False          False          False
False
3        True           True           True           True
True
4        True           True           True           True
True
5        True           True           True           True
True
6        True           True           True           True
True
7        True           True           True           True
True
8       False           True           True           True
True
9        True           True           True           True
True
```

```
   Region  Gender
0   False    True
1    True    True
2    True    True
3    True    True
```

```
4     True     True
5     True     True
6    False     True
7     True     True
8     True     True
9     True     True
```

```
df.fillna(0)
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0        60.0          200.0           60.0             75.0
2018.0
1        70.0           80.0           80.0             79.0
2019.0
2         0.0            0.0            0.0              0.0
0.0
3        80.0           85.0           65.0             80.0
2020.0
4        56.0           83.0           63.0             85.0
2021.0
5        72.0          300.0           60.0             84.0
2022.0
6        77.0           90.0           70.0             90.0
2019.0
7        68.0           95.0           71.0             95.0
2022.0
8         0.0           92.0           75.0             97.0
2023.0
9        50.0           94.0           82.0             82.0
2024.0
```

```
      Region  Gender
0          0    Male
1   Buldhana  Female
2    Kothrud  Female
3      Baner    Male
4     Nagpur    Male
5      Dadar    Male
6          0  Female
7    Kothrud    Male
8      Baner    Male
9      Warje  Female
```

```
df.fillna(20)
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0        60.0          200.0           60.0             75.0
2018.0
```

```
1          70.0          80.0          80.0          79.0
2019.0
2          20.0          20.0          20.0          20.0
20.0
3          80.0          85.0          65.0          80.0
2020.0
4          56.0          83.0          63.0          85.0
2021.0
5          72.0         300.0          60.0          84.0
2022.0
6          77.0          90.0          70.0          90.0
2019.0
7          68.0          95.0          71.0          95.0
2022.0
8          20.0          92.0          75.0          97.0
2023.0
9          50.0          94.0          82.0          82.0
2024.0
```

```
      Region  Gender
0         20    Male
1   Buldhana  Female
2    Kothrud  Female
3      Baner    Male
4     Nagpur    Male
5      Dadar    Male
6         20  Female
7    Kothrud    Male
8      Baner    Male
9      Warje  Female
```

```
df["math_score"]=df["math_score"].fillna(df["math_score"].mean())
```

```
df
```

```
    math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0       60.000          200.0           60.0             75.0
2018.0
1       70.000           80.0           80.0             79.0
2019.0
2       66.625            NaN            NaN              NaN
NaN
3       80.000           85.0           65.0             80.0
2020.0
4       56.000           83.0           63.0             85.0
2021.0
5       72.000          300.0           60.0             84.0
2022.0
6       77.000           90.0           70.0             90.0
```

```
2019.0
7       68.000           95.0           71.0           95.0
2022.0
8       66.625           92.0           75.0           97.0
2023.0
9       50.000           94.0           82.0           82.0
2024.0

      Region   Gender
0        NaN     Male
1   Buldhana   Female
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df["Reading_score"]=df["Reading_score"].fillna(df["Reading_score"].mea
n())
```

```
df
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0       60.000     200.000000           60.0             75.0
2018.0
1       70.000      80.000000           80.0             79.0
2019.0
2       66.625     124.333333            NaN              NaN
NaN
3       80.000      85.000000           65.0             80.0
2020.0
4       56.000      83.000000           63.0             85.0
2021.0
5       72.000     300.000000           60.0             84.0
2022.0
6       77.000      90.000000           70.0             90.0
2019.0
7       68.000      95.000000           71.0             95.0
2022.0
8       66.625      92.000000           75.0             97.0
2023.0
9       50.000      94.000000           82.0             82.0
2024.0

      Region  Gender
0        NaN    Male
```

```
1   Buldhana   Female
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df["Writing_score"]=df["Writing_score"].fillna(df["Writing_score"].median())
```

```
df
```

```
    math_score  Reading_score  Writing_score   Placement_score  Join_Date  \
0       60.000      200.000000           60.0              75.0     2018.0
1       70.000       80.000000           80.0              79.0     2019.0
2       66.625      124.333333           70.0               NaN        NaN
3       80.000       85.000000           65.0              80.0     2020.0
4       56.000       83.000000           63.0              85.0     2021.0
5       72.000      300.000000           60.0              84.0     2022.0
6       77.000       90.000000           70.0              90.0     2019.0
7       68.000       95.000000           71.0              95.0     2022.0
8       66.625       92.000000           75.0              97.0     2023.0
9       50.000       94.000000           82.0              82.0     2024.0

     Region   Gender
0       NaN     Male
1  Buldhana   Female
2   Kothrud   Female
3     Baner     Male
4    Nagpur     Male
5     Dadar     Male
6       NaN   Female
7   Kothrud     Male
8     Baner     Male
9     Warje   Female
```

```
df["Placement_score"]=df["Placement_score"].fillna(df["Placement_score
"].min())
```

```
df
```

```
    math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0       60.000     200.000000           60.0             75.0
2018.0
1       70.000      80.000000           80.0             79.0
2019.0
2       66.625     124.333333           70.0             75.0
NaN
3       80.000      85.000000           65.0             80.0
2020.0
4       56.000      83.000000           63.0             85.0
2021.0
5       72.000     300.000000           60.0             84.0
2022.0
6       77.000      90.000000           70.0             90.0
2019.0
7       68.000      95.000000           71.0             95.0
2022.0
8       66.625      92.000000           75.0             97.0
2023.0
9       50.000      94.000000           82.0             82.0
2024.0
```

```
      Region  Gender
0        NaN    Male
1   Buldhana  Female
2    Kothrud  Female
3      Baner    Male
4      Nagpur    Male
5      Dadar    Male
6        NaN  Female
7    Kothrud    Male
8      Baner    Male
9      Warje  Female
```

```
df
```

```
    math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0       60.000     200.000000           60.0             75.0
2018.0
1       70.000      80.000000           80.0             79.0
2019.0
2       66.625     124.333333           70.0             75.0
NaN
```

```
3       80.000       85.000000           65.0           80.0
2020.0
4       56.000       83.000000           63.0           85.0
2021.0
5       72.000      300.000000           60.0           84.0
2022.0
6       77.000       90.000000           70.0           90.0
2019.0
7       68.000       95.000000           71.0           95.0
2022.0
8       66.625       92.000000           75.0           97.0
2023.0
9       50.000       94.000000           82.0           82.0
2024.0

      Region   Gender
0        NaN     Male
1   Buldhana   Female
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

df.dropna()

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
1       70.000           80.0           80.0             79.0
2019.0
3       80.000           85.0           65.0             80.0
2020.0
4       56.000           83.0           63.0             85.0
2021.0
5       72.000          300.0           60.0             84.0
2022.0
7       68.000           95.0           71.0             95.0
2022.0
8       66.625           92.0           75.0             97.0
2023.0
9       50.000           94.0           82.0             82.0
2024.0

      Region   Gender
1   Buldhana   Female
3      Baner     Male
4     Nagpur     Male
```

```
5       Dadar      Male
7     Kothrud      Male
8       Baner      Male
9       Warje    Female
```

df

```
    math_score  Reading_score  Writing_score   Placement_score
Join_Date   \
0       60.000      200.000000           60.0              75.0
2018.0
1       70.000       80.000000           80.0              79.0
2019.0
2       66.625      124.333333           70.0              75.0
NaN
3       80.000       85.000000           65.0              80.0
2020.0
4       56.000       83.000000           63.0              85.0
2021.0
5       72.000      300.000000           60.0              84.0
2022.0
6       77.000       90.000000           70.0              90.0
2019.0
7       68.000       95.000000           71.0              95.0
2022.0
8       66.625       92.000000           75.0              97.0
2023.0
9       50.000       94.000000           82.0              82.0
2024.0
```

```
      Region   Gender
0        NaN     Male
1   Buldhana   Female
2     Kothrud  Female
3       Baner     Male
4      Nagpur     Male
5       Dadar     Male
6        NaN   Female
7     Kothrud     Male
8       Baner     Male
9       Warje   Female
```

dict1

```
{'math_score': [60, 70, nan, 80, 56, 72, 77, 68, nan, 50],
 'Reading_score': [200, 80, nan, 85, 83, 300, 90, 95, 92, 94],
 'Writing_score': [60, 80, nan, 65, 63, 60, 70, 71, 75, 82],
 'Placement_score': [75, 79, nan, 80, 85, 84, 90, 95, 97, 82],
 'Join_Date': [2018, 2019, nan, 2020, 2021, 2022, 2019, 2022, 2023,
2024],
```

```
 'Region': [nan,
  'Buldhana',
  'Kothrud',
  'Baner',
  'Nagpur',
  'Dadar',
  nan,
  'Kothrud',
  'Baner',
  'Warje'],
 'Gender': ['Male',
  "Female",
  "Female",
  'Male',
  'Male',
  'Male',
  "Female",
  'Male',
  'Male',
  "Female"]}

df=pd.DataFrame(dict1)
df
```

	math_score	Reading_score	Writing_score	Placement_score	Join_Date
0	60.0	200.0	60.0	75.0	2018.0
1	70.0	80.0	80.0	79.0	2019.0
2	NaN	NaN	NaN	NaN	NaN
3	80.0	85.0	65.0	80.0	2020.0
4	56.0	83.0	63.0	85.0	2021.0
5	72.0	300.0	60.0	84.0	2022.0
6	77.0	90.0	70.0	90.0	2019.0
7	68.0	95.0	71.0	95.0	2022.0
8	NaN	92.0	75.0	97.0	2023.0
9	50.0	94.0	82.0	82.0	2024.0

	Region	Gender
0	NaN	Male
1	Buldhana	Female

```
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df.dropna(how="all")
```

```
   math_score  Reading_score  Writing_score   Placement_score
Join_Date   \
0         60.0          200.0           60.0              75.0
2018.0
1         70.0           80.0           80.0              79.0
2019.0
2          NaN            NaN            NaN               NaN
NaN
3         80.0           85.0           65.0              80.0
2020.0
4         56.0           83.0           63.0              85.0
2021.0
5         72.0          300.0           60.0              84.0
2022.0
6         77.0           90.0           70.0              90.0
2019.0
7         68.0           95.0           71.0              95.0
2022.0
8          NaN           92.0           75.0              97.0
2023.0
9         50.0           94.0           82.0              82.0
2024.0

     Region   Gender
0       NaN     Male
1  Buldhana   Female
2    Kothrud  Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df
```

```
   math_score  Reading_score  Writing_score   Placement_score
Join_Date   \
```

```
0         60.0         200.0         60.0         75.0
2018.0
1         70.0          80.0         80.0         79.0
2019.0
2          NaN           NaN          NaN          NaN
NaN
3         80.0          85.0         65.0         80.0
2020.0
4         56.0          83.0         63.0         85.0
2021.0
5         72.0         300.0         60.0         84.0
2022.0
6         77.0          90.0         70.0         90.0
2019.0
7         68.0          95.0         71.0         95.0
2022.0
8          NaN          92.0         75.0         97.0
2023.0
9         50.0          94.0         82.0         82.0
2024.0
```

	Region	Gender
0	NaN	Male
1	Buldhana	Female
2	Kothrud	Female
3	Baner	Male
4	Nagpur	Male
5	Dadar	Male
6	NaN	Female
7	Kothrud	Male
8	Baner	Male
9	Warje	Female

```
df.dropna(axis=1)
```

	Gender
0	Male
1	Female
2	Female
3	Male
4	Male
5	Male
6	Female
7	Male
8	Male
9	Female

```
df
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0         60.0          200.0           60.0             75.0
2018.0
1         70.0           80.0           80.0             79.0
2019.0
2          NaN            NaN            NaN              NaN
NaN
3         80.0           85.0           65.0             80.0
2020.0
4         56.0           83.0           63.0             85.0
2021.0
5         72.0          300.0           60.0             84.0
2022.0
6         77.0           90.0           70.0             90.0
2019.0
7         68.0           95.0           71.0             95.0
2022.0
8          NaN           92.0           75.0             97.0
2023.0
9         50.0           94.0           82.0             82.0
2024.0

      Region  Gender
0        NaN    Male
1   Buldhana  Female
2    Kothrud  Female
3      Baner    Male
4     Nagpur    Male
5      Dadar    Male
6        NaN  Female
7    Kothrud    Male
8      Baner    Male
9      Warje  Female
```

```
df.dropna(axis=0,how="any")
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
1         70.0           80.0           80.0             79.0
2019.0
3         80.0           85.0           65.0             80.0
2020.0
4         56.0           83.0           63.0             85.0
2021.0
5         72.0          300.0           60.0             84.0
2022.0
7         68.0           95.0           71.0             95.0
2022.0
9         50.0           94.0           82.0             82.0
```

```
2024.0

      Region  Gender
1   Buldhana  Female
3      Baner    Male
4     Nagpur    Male
5      Dadar    Male
7    Kothrud    Male
9      Warje  Female

df

   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0        60.0          200.0           60.0             75.0
2018.0
1        70.0           80.0           80.0             79.0
2019.0
2         NaN            NaN            NaN              NaN
NaN
3        80.0           85.0           65.0             80.0
2020.0
4        56.0           83.0           63.0             85.0
2021.0
5        72.0          300.0           60.0             84.0
2022.0
6        77.0           90.0           70.0             90.0
2019.0
7        68.0           95.0           71.0             95.0
2022.0
8         NaN           92.0           75.0             97.0
2023.0
9        50.0           94.0           82.0             82.0
2024.0

      Region  Gender
0        NaN    Male
1   Buldhana  Female
2    Kothrud  Female
3      Baner    Male
4     Nagpur    Male
5      Dadar    Male
6        NaN  Female
7    Kothrud    Male
8      Baner    Male
9      Warje  Female

df.replace(to_replace=np.nan,value=60)
```

	math_score	Reading_score	Writing_score	Placement_score	Join_Date
0	60.0	200.0	60.0	75.0	2018.0
1	70.0	80.0	80.0	79.0	2019.0
2	60.0	60.0	60.0	60.0	60.0
3	80.0	85.0	65.0	80.0	2020.0
4	56.0	83.0	63.0	85.0	2021.0
5	72.0	300.0	60.0	84.0	2022.0
6	77.0	90.0	70.0	90.0	2019.0
7	68.0	95.0	71.0	95.0	2022.0
8	60.0	92.0	75.0	97.0	2023.0
9	50.0	94.0	82.0	82.0	2024.0

	Region	Gender
0	60	Male
1	Buldhana	Female
2	Kothrud	Female
3	Baner	Male
4	Nagpur	Male
5	Dadar	Male
6	60	Female
7	Kothrud	Male
8	Baner	Male
9	Warje	Female

df

	math_score	Reading_score	Writing_score	Placement_score	Join_Date
0	60.0	200.0	60.0	75.0	2018.0
1	70.0	80.0	80.0	79.0	2019.0
2	NaN	NaN	NaN	NaN	NaN
3	80.0	85.0	65.0	80.0	2020.0
4	56.0	83.0	63.0	85.0	2021.0
5	72.0	300.0	60.0	84.0	

```
2022.0
6         77.0           90.0           70.0           90.0
2019.0
7         68.0           95.0           71.0           95.0
2022.0
8          NaN           92.0           75.0           97.0
2023.0
9         50.0           94.0           82.0           82.0
2024.0

      Region   Gender
0        NaN     Male
1   Buldhana   Female
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df.dropna(subset=["Region","Gender"])
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
1         70.0           80.0           80.0           79.0
2019.0
2          NaN            NaN            NaN            NaN
NaN
3         80.0           85.0           65.0           80.0
2020.0
4         56.0           83.0           63.0           85.0
2021.0
5         72.0          300.0           60.0           84.0
2022.0
7         68.0           95.0           71.0           95.0
2022.0
8          NaN           92.0           75.0           97.0
2023.0
9         50.0           94.0           82.0           82.0
2024.0

      Region   Gender
1   Buldhana   Female
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
```

```
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
0        60.0          200.0           60.0             75.0
2018.0
1        70.0           80.0           80.0             79.0
2019.0
2         NaN            NaN            NaN              NaN
NaN
3        80.0           85.0           65.0             80.0
2020.0
4        56.0           83.0           63.0             85.0
2021.0
5        72.0          300.0           60.0             84.0
2022.0
6        77.0           90.0           70.0             90.0
2019.0
7        68.0           95.0           71.0             95.0
2022.0
8         NaN           92.0           75.0             97.0
2023.0
9        50.0           94.0           82.0             82.0
2024.0
```

```
     Region   Gender
0       NaN     Male
1  Buldhana   Female
2   Kothrud   Female
3     Baner     Male
4    Nagpur     Male
5     Dadar     Male
6       NaN   Female
7   Kothrud     Male
8     Baner     Male
9     Warje   Female
```

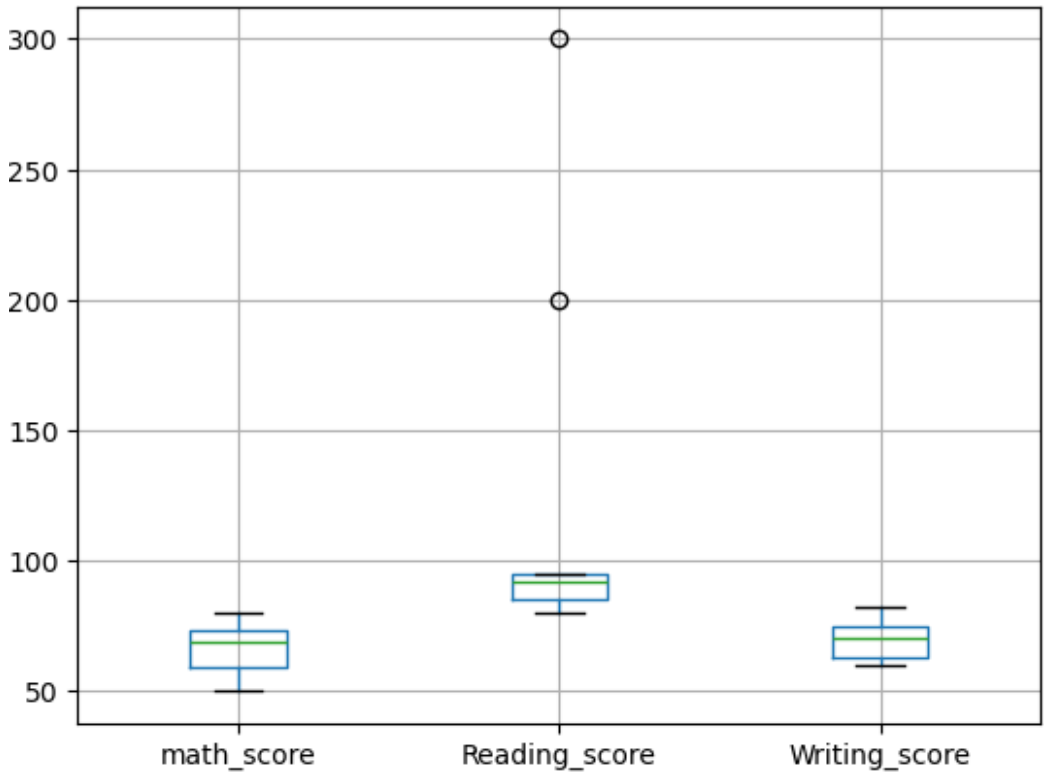
```
df["Region"].replace(to_replace=np.nan,value="Kota")
```

```
0         Kota
1     Buldhana
2     Kothrud
3       Baner
4      Nagpur
5       Dadar
6         Kota
```

```
7      Kothrud
8        Baner
9        Warje
Name: Region, dtype: object
```

```
col1=["math_score","Reading_score","Writing_score"]
df.boxplot(col1)
```

```
<Axes: >
```




```
df
```

	math_score	Reading_score	Writing_score	Placement_score	Join_Date
0	60.0	200.0	60.0	75.0	2018.0
1	70.0	80.0	80.0	79.0	2019.0
2	NaN	NaN	NaN	NaN	NaN
3	80.0	85.0	65.0	80.0	2020.0
4	56.0	83.0	63.0	85.0	2021.0
5	72.0	300.0	60.0	84.0	

```
2022.0
6         77.0             90.0             70.0             90.0
2019.0
7         68.0             95.0             71.0             95.0
2022.0
8          NaN             92.0             75.0             97.0
2023.0
9         50.0             94.0             82.0             82.0
2024.0

      Region  Gender
0        NaN    Male
1   Buldhana  Female
2    Kothrud  Female
3      Baner    Male
4     Nagpur    Male
5      Dadar    Male
6        NaN  Female
7    Kothrud    Male
8      Baner    Male
9      Warje  Female
```

df

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date   \
0         60.0          200.0           60.0             75.0
2018.0
1         70.0           80.0           80.0             79.0
2019.0
2          NaN            NaN            NaN              NaN
NaN
3         80.0           85.0           65.0             80.0
2020.0
4         56.0           83.0           63.0             85.0
2021.0
5         72.0          300.0           60.0             84.0
2022.0
6         77.0           90.0           70.0             90.0
2019.0
7         68.0           95.0           71.0             95.0
2022.0
8          NaN           92.0           75.0             97.0
2023.0
9         50.0           94.0           82.0             82.0
2024.0

      Region  Gender
0        NaN    Male
1   Buldhana  Female
```

```
2    Kothrud   Female
3      Baner     Male
4     Nagpur     Male
5      Dadar     Male
6        NaN   Female
7    Kothrud     Male
8      Baner     Male
9      Warje   Female
```

```
df.dropna(inplace=True)
df
```

```
   math_score  Reading_score  Writing_score  Placement_score
Join_Date  \
1        70.0           80.0           80.0             79.0
2019.0
3        80.0           85.0           65.0             80.0
2020.0
4        56.0           83.0           63.0             85.0
2021.0
5        72.0          300.0           60.0             84.0
2022.0
7        68.0           95.0           71.0             95.0
2022.0
9        50.0           94.0           82.0             82.0
2024.0

     Region  Gender
1  Buldhana  Female
3     Baner    Male
4    Nagpur    Male
5     Dadar    Male
7   Kothrud    Male
9     Warje  Female
```

```
coll=['math_score','Reading_score','Writing_score']
df.boxplot(coll)
```

```
<Axes: >
```

Assignment No: 3
Rollno:TC021F035

```
import pandas as pd
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-00cf07b74dcd> in <cell line: 1>()
----> 1 df

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

New section

```
df.mean()
```

```
df.loc[:,"Age"].mean()
```

```
38.85
```

```
df.mean(axis=0)
```

```
<ipython-input-78-1c43c59c9f98>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.mean(axis=0)
```

```
CustomerID                100.50
Age                        38.85
Annual Income (k$)         60.56
Spending Score (1-100)     50.20
dtype: float64
```

```
df.median()
```

```
<ipython-input-79-6d467abf240d>:1: FutureWarning: The default value of numeric_only in DataFrame.median is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.median()
```

```
CustomerID                100.5
Age                        36.0
Annual Income (k$)         61.5
```

```
Spending Score (1-100)    50.0
dtype: float64
```

```
df.mode()
```


```
{"summary":"{\n \"name\": \"df\",\n \"rows\": 200,\n \"fields\": [\n    {\n      \"column\": \"CustomerID\",\n    \"properties\": {\n \"dtype\": \"number\",\n      \"std\": 57,\n      \"min\": 1,\n \"max\": 200,\n      \"num_unique_values\": 200,\n \"samples\": [\n          96,\n          16,\n          31\n      ],\n      \"semantic_type\": \"\",\n \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Genre\",\n      \"properties\": {\n          \"dtype\": \"category\",\n      \"num_unique_values\": 1,\n      \"samples\": [\n \"Female\"\n      ],\n          \"semantic_type\": \"\",\n \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n          \"dtype\": \"number\",\n \"std\": null,\n      \"min\": 32.0,\n      \"max\": 32.0,\n \"num_unique_values\": 1,\n      \"samples\": [\n          32.0\n ],\n      \"semantic_type\": \"\",\n      \"description\": \"\"\n }\n    },\n    {\n      \"column\": \"Annual Income (k$)\",\n \"properties\": {\n          \"dtype\": \"number\",\n      \"std\": 16.97056274847714,\n    \"min\": 54.0,\n      \"max\": 78.0,\n \"num_unique_values\": 2,\n      \"samples\": [\n          78.0\n ],\n      \"semantic_type\": \"\",\n      \"description\": \"\"\n }\n    },\n    {\n      \"column\": \"Spending Score (1-100)\",\n \"properties\": {\n      \"dtype\": \"number\",\n    \"std\": null,\n      \"min\": 42.0,\n      \"max\": 42.0,\n \"num_unique_values\": 1,\n      \"samples\": [\n          42.0\n ],\n      \"semantic_type\": \"\",\n      \"description\": \"\"\n }\n    }\n ]\n}","type":"dataframe"}
```


```
df.std()
```

```
<ipython-input-81-ce97bb7eaef8>:1: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.std()
```

```
CustomerID                57.879185
Age                       13.969007
Annual Income (k$)        26.264721
Spending Score (1-100)    25.823522
dtype: float64
```

```
df.groupby(["Genre"])["Age"].mean()
```

```
Genre
Female    38.098214
Male      39.806818
Name: Age, dtype: float64
```

```
df.groupby(['Genre'])["Annual Income (k$)"].mean()
```

```
Genre
Female    59.250000
Male      62.227273
Name: Annual Income (k$), dtype: float64
```

```
df.groupby(['Genre']).mean()
```


```
{"summary":"{\n \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\":  [\n
{\n      \"column\": \"CustomerID\",\n     \"properties\": {\n
\"dtype\": \"number\",\n        \"std\":  4.720741294853369,\n
\"min\": 97.5625,\n        \"max\": 104.23863636363636,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
104.23863636363636,\n      97.5625\n       ],\n
\"semantic_type\": \"\",\n         \"description\": \"\"\n      }\
n    },\n     {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\":  1.2081654012968197,\n
\"min\":  38.098214285714285,\n      \"max\": 39.80681818181818,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
39.80681818181818,\n       38.098214285714285\n     ],\n
\"semantic_type\": \"\",\n         \"description\": \"\"\n       }\
n    },\n     {\n      \"column\": \"Annual Income (k$)\",\n
\"properties\": {\n         \"dtype\": \"number\",\n        \"std\":
2.105249734896311 5,\n     \"min\": 59.25,\n        \"max\":
62.22727272727273,\n     \"num_unique_values\": 2,\n
\"samples\": [\n          62.22727272727273,\n       59.25\n
],\n         \"semantic_type\": \"\",\n         \"description\": \"\"\n
}\n    },\n     {\n      \"column\": \"Spending Score (1-100)\",\n
\"properties\": {\n         \"dtype\": \"number\",\n        \"std\":
2.132225399438334,\n     \"min\": 48.51136363636363,\n
\"max\": 51.526785714285715,\n     \"num_unique_values\": 2,\n
\"samples\": [\n          48.51136363636363,\n
51.526785714285715\n     ],\n         \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
df.groupby(['Genre']).median()
```


```
{"summary":"{\n \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\":  [\n
{\n      \"column\": \"CustomerID\",\n     \"properties\": {\n
\"dtype\": \"number\",\n        \"std\":  8.48528137423857,\n
\"min\": 94.5,\n        \"max\": 106.5,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          106.5,\n
94.5\n       ],\n         \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n     {\n      \"column\":
\"Age\",\n      \"properties\": {\n         \"dtype\": \"number\",\n
```

\"std\": 1.4142135623730951,\n \"min\": 35.0,\n \"max\": 37.0,\n \"num_unique_values\": 2,\n \"samples\": [\n 37.0,\n 35.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Annual Income (k$)\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 1.7677669529663689,\n \"min\": 60.0,\n \"max\": 62.5,\n \"num_unique_values\": 2,\n \"samples\": [\n 62.5,\n 60.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Spending Score (1-100)\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.0,\n \"min\": 50.0,\n \"max\": 50.0,\n \"num_unique_values\": 1,\n \"samples\": [\n 50.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe"}

```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df=(enc.fit_transform(df[['Genre']]).toarray())
enc_df
x=pd.DataFrame(enc_df)
df_encode = df.join(x)
df_encode
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-e9e68a97a0e9> in <cell line: 3>()
      1 from sklearn import preprocessing
      2 enc = preprocessing.OneHotEncoder()
----> 3 enc_df=(enc.fit_transform(df[['Genre']]).toarray())
      4 enc_df
      5 x=pd.DataFrame(enc_df)

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

```
import pandas as pd

iris=pd.read_csv("/content/iris.csv")

iris
```

{"summary":"{\n \"name\": \"iris\",\n \"rows\": 149,\n \"fields\": [\n {\n \"column\": \"5.1\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.8285940572656173,\n \"min\": 4.3,\n \"max\": 7.9,\n \"num_unique_values\": 35,\n \"samples\": [\n 6.2,\n 4.5,\n 5.6\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"3.5\",\n \"properties\": {\n \"dtype\": \"number\",\n

\"std\": 0.4334988777167476,\n \"min\": 2.0,\n \"max\": 4.4,\n \"num_unique_values\": 23,\n \"samples\": [\n 2.3,\n 4.4,\n 3.0\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"1.4\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 1.7596511617753423,\n \"min\": 1.0,\n \"max\": 6.9,\n \"num_unique_values\": 43,\n \"samples\": [\n 6.7,\n 3.8,\n 3.7\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"0.2\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0.7612920413899603,\n \"min\": 0.1,\n \"max\": 2.5,\n \"num_unique_values\": 22,\n \"samples\": [\n 0.2,\n 1.2,\n 1.3\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Iris-setosa\",\n \"properties\": {\n \"dtype\": \"category\",\n \"num_unique_values\": 3,\n \"samples\": [\n \"Iris-setosa\",\n \"Iris-versicolor\",\n \"Iris-virginica\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","type":"dataframe","variable_name":"iris"}

```
iris["Iris-setosa"].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
dtype=object)
```

```
irisSet=(iris['Iris-setosa']=='setosa')
```

```
print(iris[irisSet].describe())
```

	5.1	3.5	1.4	0.2
count	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

```
irisSet1=(iris['Iris-setosa']=='versicolor')
```

```
print(iris[irisSet1].describe())
```

	5.1	3.5	1.4	0.2
count	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN

```
25%     NaN   NaN   NaN   NaN
50%     NaN   NaN   NaN   NaN
75%     NaN   NaN   NaN   NaN
max     NaN   NaN   NaN   NaN
```

```
irisSet2=(iris['Iris-setosa']=='virginica')

print(iris[irisSet2].describe())
```

```
       5.1   3.5   1.4   0.2
count  0.0   0.0   0.0   0.0
mean   NaN   NaN   NaN   NaN
std    NaN   NaN   NaN   NaN
min    NaN   NaN   NaN   NaN
25%    NaN   NaN   NaN   NaN
50%    NaN   NaN   NaN   NaN
75%    NaN   NaN   NaN   NaN
max    NaN   NaN   NaN   NaN
```

```
print("Iris-Setosa")
print(iris[irisSet].describe())
print("Iris-Versicolor")
print(iris[irisSet1].describe())
print("Iris-Virginica")
print(iris[irisSet2].describe())
```

```
Iris-Setosa
       5.1   3.5   1.4   0.2
count  0.0   0.0   0.0   0.0
mean   NaN   NaN   NaN   NaN
std    NaN   NaN   NaN   NaN
min    NaN   NaN   NaN   NaN
25%    NaN   NaN   NaN   NaN
50%    NaN   NaN   NaN   NaN
75%    NaN   NaN   NaN   NaN
max    NaN   NaN   NaN   NaN
Iris-Versicolor
       5.1   3.5   1.4   0.2
count  0.0   0.0   0.0   0.0
mean   NaN   NaN   NaN   NaN
std    NaN   NaN   NaN   NaN
min    NaN   NaN   NaN   NaN
25%    NaN   NaN   NaN   NaN
50%    NaN   NaN   NaN   NaN
75%    NaN   NaN   NaN   NaN
max    NaN   NaN   NaN   NaN
Iris-Virginica
       5.1   3.5   1.4   0.2
count  0.0   0.0   0.0   0.0
mean   NaN   NaN   NaN   NaN
```

std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

Assignment No:4

Rollno:TC021F035

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

df= pd.read_csv("/home/ubuntu/Downloads/archive (14)/HousingData.csv")

df
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX
\
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296

1    0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242

2    0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242

3    0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222

4    0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222

..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...

501  0.06263   0.0  11.93   0.0  0.573  6.593  69.1  2.4786    1  273

502  0.04527   0.0  11.93   0.0  0.573  6.120  76.7  2.2875    1  273

503  0.06076   0.0  11.93   0.0  0.573  6.976  91.0  2.1675    1  273

504  0.10959   0.0  11.93   0.0  0.573  6.794  89.3  2.3889    1  273

505  0.04741   0.0  11.93   0.0  0.573  6.030   NaN  2.5050    1  273


     PTRATIO       B  LSTAT  MEDV
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90    NaN  36.2
..       ...     ...    ...   ...
501     21.0  391.99    NaN  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

```
#df=pd.set_option("display.max_rows",None)

df= pd.read_csv("/home/ubuntu/Downloads/archive (14)/HousingData.csv")
```

```
df.shape
```

```
(506, 14)
```

```
df.isnull().sum()
```

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

```
df["CRIM"].fillna(df["CRIM"].mean(),inplace=True)
df["ZN"].fillna(df["ZN"].mean(),inplace=True)
df["INDUS"].fillna(df["INDUS"].mean(),inplace=True)
df["CHAS"].fillna(df["CHAS"].mean(),inplace=True)
df["AGE"].fillna(df["AGE"].mean(),inplace=True)
df["LSTAT"].fillna(df["LSTAT"].mean(),inplace=True)
df
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	\
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.200000	4.0900	1	296	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.900000	4.9671	2	242	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.100000	4.9671	2	242	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.800000	6.0622	3	222	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.200000	6.0622	3	222	
..	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.100000	2.4786	1	273	
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.700000	2.2875	1	273	
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.000000	2.1675	1	273	

```
504  0.10959   0.0   11.93   0.0  0.573  6.794  89.300000  2.3889     1
273
505  0.04741   0.0   11.93   0.0  0.573  6.030  68.518519  2.5050     1
273

     PTRATIO       B       LSTAT  MEDV
0       15.3  396.90    4.980000  24.0
1       17.8  396.90    9.140000  21.6
2       17.8  392.83    4.030000  34.7
3       18.7  394.63    2.940000  33.4
4       18.7  396.90   12.715432  36.2
..       ...     ...         ...   ...
501     21.0  391.99   12.715432  22.4
502     21.0  396.90    9.080000  20.6
503     21.0  396.90    5.640000  23.9
504     21.0  393.45    6.480000  22.0
505     21.0  396.90    7.880000  11.9

[506 rows x 14 columns]
```

```
df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

```
x = df.drop(['MEDV'], axis = 1)
x
```

```
        CRIM    ZN  INDUS  CHAS    NOX     RM       AGE     DIS  RAD
TAX  \
0    0.00632  18.0   2.31   0.0  0.538  6.575  65.200000  4.0900    1
296
1    0.02731   0.0   7.07   0.0  0.469  6.421  78.900000  4.9671    2
242
2    0.02729   0.0   7.07   0.0  0.469  7.185  61.100000  4.9671    2
242
```

```
3    0.03237    0.0    2.18    0.0  0.458  6.998  45.800000  6.0622     3
222
4    0.06905    0.0    2.18    0.0  0.458  7.147  54.200000  6.0622     3
222
..       ...    ...     ...    ...    ...    ...        ...     ...   ...
...
501  0.06263    0.0   11.93    0.0  0.573  6.593  69.100000  2.4786     1
273
502  0.04527    0.0   11.93    0.0  0.573  6.120  76.700000  2.2875     1
273
503  0.06076    0.0   11.93    0.0  0.573  6.976  91.000000  2.1675     1
273
504  0.10959    0.0   11.93    0.0  0.573  6.794  89.300000  2.3889     1
273
505  0.04741    0.0   11.93    0.0  0.573  6.030  68.518519  2.5050     1
273

     PTRATIO       B      LSTAT
0       15.3  396.90   4.980000
1       17.8  396.90   9.140000
2       17.8  392.83   4.030000
3       18.7  394.63   2.940000
4       18.7  396.90  12.715432
..       ...     ...        ...
501     21.0  391.99  12.715432
502     21.0  396.90   9.080000
503     21.0  396.90   5.640000
504     21.0  393.45   6.480000
505     21.0  396.90   7.880000

[506 rows x 13 columns]
```

```
y=df['MEDV']
y
```

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size
=0.2,random_state = 0)

from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(xtrain, ytrain)
```

```
LinearRegression()
```

```
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)

ytrain_pred
```

```
array([32.81627321, 22.44810156, 28.03766232, 23.75733198,
6.50947414,
       14.03444098, 22.08820099, 29.25415603, 32.5690748 ,
13.05640904,
       20.22623633, 21.50935141, 13.130115  , 23.97459357,
5.94369526,
       19.18872715, 9.22208539, 45.332114 , 30.74904711,
17.36508271,
       17.89690787, 21.8538289 , 23.41427455, 19.26643055,
35.0106092 ,
       13.80465069, 20.97652211, 35.5947274 , 19.11750028,
13.64565313,
       13.99390551, 22.3513967 , 14.97018808, 31.41563776,
25.42200597,
       16.12261841, 24.8151741 ,  9.77305866, 14.99985459,
21.7069503 ,
       33.07056401, 28.27079451, 24.99059687, 15.53957856,
31.8131073 ,
       25.31962943, 14.20026892,  7.85538779, 27.95901931,
25.43216539,
        5.06592749, 28.31123002, 16.96468964, 29.9148188 ,
19.39108637,
       16.30580326, 18.42774446, 12.95370034,  8.93495761,
19.25886782,
       34.02016271, 32.86420157, 23.95497256, 20.14492197,
22.95083775,
       26.67615398, 21.10735155, 17.74979344, 32.38278077,
10.58254731,
       19.13318567, 31.64516362, 18.84520418, 15.78422296,
18.94046927,
       15.11678414, 24.07458559, 23.66247081, 17.31713844,
13.21910258,
       20.51857201, 24.22417338, 17.56807076, 25.39478716,
22.84498687,
       28.06459103, 36.75378827, 16.43162613, 12.07850424,
```

```
35.04746739,
       31.32063269, 20.48029146, 39.84323922, 28.61226369,
28.52233259,
       17.5436698 , 26.81225348, 40.46530719, 27.56913544,
17.03883567,
       37.52638916, 35.8229176 , 14.0961138 , 27.81731446,
21.94869655,
       24.8818496 , 21.19719131, 23.45740081, 28.00487668,
29.57431557,
       13.98506722, 26.16081663, 22.9901289 , 13.49729741,
14.06839936,
       25.56540702, 19.52524407, 30.57606788, 10.04770818,
24.37160039,
       17.17149942, 17.0112022 , 22.84937882, 21.67962795,
12.22063591,
       25.20446182, 28.42197799, 20.75309338, 12.37886993,
25.11306051,
       26.49461989, 25.70733316, 23.55013679, 25.89336895,
19.39553691,
       20.85778701, 36.13738934, 21.1272548 , 36.28019204,
25.81708451,
       20.90978431, 15.58319854, 31.98314685, 21.54080142,
28.05396355,
       14.844027, 32.73726538, 14.35524691,  1.67919971,
19.44711142,
       13.87157102, 37.66832351, 16.30974204, 14.5549353 ,
26.76951017,
       23.52976973, 17.9629898 , 31.18728628, 25.18377746,
27.6157619 ,
       24.80190806, 22.65264492, 22.46031495, 11.15172696,
20.91842234,
       11.68647636, 17.73974341, 12.46032947, 27.77373401,
15.19251431,
       15.99965125, 28.65756198, 14.46036826, 21.62746997,
12.64254329,
       14.45555483, 23.36423324, 21.29797228, 14.93440596,
17.48649966,
       13.45046456, 24.24295419, 12.50361345, 35.3211587 ,
14.03760167,
       43.21873439, 31.66742609, 34.8841818 , 22.01139076,
15.82180035,
       26.85423351, 29.2313597 , 13.77457177, 26.727208  ,
36.27945842,
       16.83686285, 11.36251837, 34.42259379, 35.99230127,
17.96133076,
       21.19847959, 18.52795439, 24.4993728 , 19.58258696,
27.1900666 ,
       -4.30707212, 20.86540926, 32.56708668, 35.64347839,
25.17123763,
```

```
       26.89080764,  20.27656175,  21.38857599,  15.98382948,
17.86132517,
       21.07756281, 28.04291179, 19.96986817,   6.97597498,
16.23743192,
       32.36451304,  35.49074744,  16.44786434,  18.93927412,
22.33854817,
        6.31311559,  21.58666499,  23.55661668,  15.92257792,
18.47627952,
       23.05203916,  27.21603015,  25.92177059,  32.83874569,
14.87693591,
       28.97948914,  25.26104179,  21.07941467,  38.66910342,
20.48523852,
       23.62439463,  22.76124186,  11.92347536,  19.99770627,
33.42769853,
       24.80122075,  17.81400458,  33.31685947,  22.07230652,
28.71857835,
       32.15452084,  36.60591303,  21.92092982,  24.10484055,
23.11676177,
       32.01346165,  22.26638068,  18.36298427,  21.92771527,
29.11853042,
       22.86612836,  22.06878986,  17.24027069,  17.3597754 ,
16.97472858,
       16.93337092,  16.77794732,  32.03226006,  23.32756224,
17.49983857,
       19.30181049,  34.16756391,  14.14434098,  25.93018821,
17.01947728,
       30.72513401,  30.03828448,  21.25506147,  20.3298965 ,
36.06896982,
       20.55702264,  33.38547011,  21.24908409,  31.57709481,
30.27897298,
       37.42356288,  25.97972218,  21.09103073,  29.15810192,
15.96108991,
       26.18276066,  21.50175208,  30.02080424,  10.53454993,
31.34892003,
        6.17985967,  15.25781318,  20.54161826,  35.68172035,
31.93778897,
       12.06580266,  13.62606496,  21.96053599,  34.80135807,
18.76201757,
       18.62293267,  14.91881869,  25.48278612,  41.01931145,
25.34871174,
       42.07887857,  25.45001621,  21.13187064,  12.07131249,
15.86997446,
       14.20602197, 18.51957266,   3.01474407, 27.73065016,
26.41221664,
       41.70198443,  21.64461258,  21.05335926,  34.0757984 ,
32.95054801,
        9.64398016,  24.85070793,  43.79024787,  21.78807344,
17.6920574 ,
       26.16218871, 18.58628594,   6.34645098, 18.91507749,
```

```
35.66319094,
        16.30328017,  23.75087974,  13.14594653,  24.39977842,
18.271     ,
        17.18171595,  18.47353031,  33.00268863,  19.48068366,
29.83962214,
        31.93425646,  41.60423999,  18.49648542,  16.12293921,
38.25510656,
        17.76581487,  10.55196656, 14.7688579 , 25.35067163,
19.46964974,
        16.46915179,  26.67604749, 13.37142686,  5.91891273,
18.71816701,
        10.83757858,  28.50859921,  4.96207098, 28.66572375,
32.80875045,
        22.61360755,  16.44226312,  17.93061892,  21.13503675,
34.02654674,
        28.24386741,  19.33997782, 20.46592732,  6.76629059,
28.97480848,
        25.03253899,  22.44360225, 13.7417905 , 24.80011236,
19.46303725,
         8.97368226,  26.78806235,  16.0120091 ,  31.5636985 ,
31.99530544,
        25.19164941,  18.43188624,  30.61644449,  21.28067267,
25.9311395 ,
        24.31670018,  31.19585054,  24.60225016,  31.38261045,
17.64408955,
        19.80887282,  18.65517922,  41.33253844,  25.56986444,
19.34047326,
        33.40599414,  23.70130494,  18.3221544 ,  23.25156216])
```

```
lm.predict([[0.006320,18.000000,2.310000,0.000000,0.5380,6.575,65.200000,4.0900,1,296,15.3,396.90,4.980000]])
```

```
/home/ubuntu/.local/lib/python3.8/site-packages/sklearn/base.py:420: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

```
array([30.72513401])
```

```
ytest_pred
```

```
array([26.175296 , 22.64747588, 29.1456294 , 11.52971235,
21.65312134,
        19.42320699,  20.18413017,  21.46914355,  19.1985363 ,
19.98228162,
         4.32483046, 16.16891668, 16.87682404,  5.31232373,
39.36827861,
        33.09358732,  21.9152876 ,  36.61918436,  31.52676377,
23.52713482,
        24.96022461,  23.69866912,  20.88033802,  30.55074901,
```

```
22.74081741,
         8.66805959, 17.65119072, 17.93088633, 36.01223185,
21.16299556,
        17.83464361, 17.43306603, 19.5240167 , 23.50605522,
28.97262793,
        19.21808862, 11.23997435, 23.94256597, 17.86786717,
15.40849806,
        26.3630836 , 21.5193299 , 23.78733694, 14.84041522,
23.9445175 ,
        24.97067627, 20.11366175, 23.08636158, 10.42208266,
24.52832122,
        21.60847326, 18.66228165, 24.53362832, 31.03502944,
12.97457826,
        22.38536236, 21.34822822, 16.10928673, 12.37477824,
22.78596712,
        18.28714824, 21.91802045, 32.49771603, 31.21256855,
17.47867791,
        33.18861907, 19.17896285, 19.94662594, 20.17142015,
23.90228857,
        22.81288844, 24.17911208, 30.83402844, 28.87481037,
25.14581721,
         5.55072029, 37.0183454 , 24.15428003, 27.67587636,
19.63884644,
        28.74874123, 18.83204358, 17.63305678, 37.97947167,
39.49507972,
        24.17228966, 25.33605088, 16.75044819, 25.43224687,
16.65089426,
        16.49186628, 13.37283452, 24.81689254, 31.21188699,
22.0891919 ,
        20.49360168,  0.8229737 , 25.5004737 , 15.5481509 ,
17.72901193,
        25.77663998, 22.43131323])
```

ytrain

```
220    26.7
71     21.7
240    22.0
6      22.9
417    10.4
       ...
323    18.5
192    36.4
117    19.2
47     16.6
172    23.1
Name: MEDV, Length: 404, dtype: float64
```

ytest

```
329    22.6
371    50.0
219    23.0
403     8.3
78     21.2
       ...
56     24.7
455    14.1
60     18.7
213    28.1
108    19.8
Name: MEDV, Length: 102, dtype: float64
```

```
df1=pd.DataFrame(ytrain_pred,ytrain)
df2=pd.DataFrame(ytest_pred,ytest)
```

df1

```
                0
MEDV
26.7   32.816273
21.7   22.448102
22.0   28.037662
22.9   23.757332
10.4    6.509474
...          ...
18.5   19.340473
36.4   33.405994
19.2   23.701305
16.6   18.322154
23.1   23.251562

[404 rows x 1 columns]
```

df2

```
                0
MEDV
22.6   26.175296
50.0   22.647476
23.0   29.145629
8.3    11.529712
21.2   21.653121
...          ...
24.7   25.500474
14.1   15.548151
18.7   17.729012
28.1   25.776640
19.8   22.431313

[102 rows x 1 columns]
```

```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

```
34.98738954423878
```

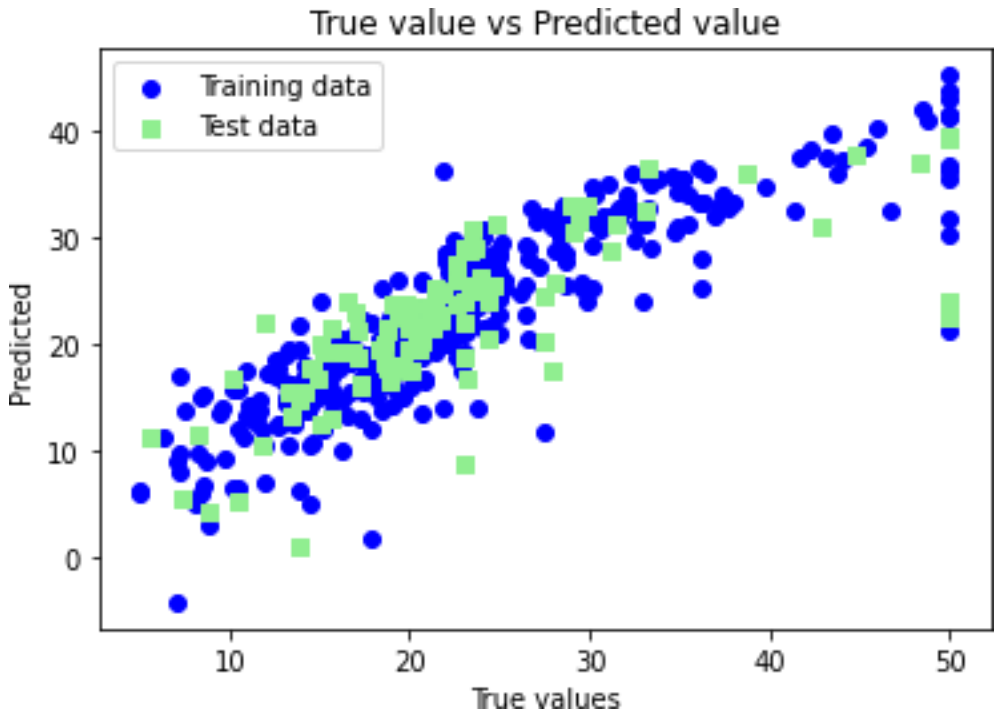
```
mse = mean_squared_error(ytrain_pred,ytrain)
print(mse)
```

```
20.019115913036593
```

```
r2=r2_score(ytest, ytest_pred)
r2
```

```
0.5703296053895557
```

```
plt.scatter(ytrain ,ytrain_pred,c="blue",marker="o",label="Training data')
plt.scatter(ytest,ytest_pred ,c="lightgreen",marker="s",label="Test data')
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left')
#plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```



Assignment No:5

Rollno:TC021F035

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

df=pd.read_csv("/home/ubuntu/Downloads/Soci
                al_Network_Ads.csv")
df
```

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
..	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

[400 rows x 3 columns]

```
df.isnull().sum()
```

```
Age                0
EstimatedSalary    0
Purchased          0
```

dtype: int64

```
x=df.drop("Purchased",axis=1)
x
```

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
..	...	...
395	46	41000
396	51	23000
397	50	20000
398	36	33000
399	49	36000

[400 rows x 2 columns]

```
y=df.Purchased
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size
=0.2,random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)

xtest
```

```
array([[-7.98950822e-01,  4.94607583e-01],
       [-2.12648508e-02, -5.77359062e-01], [-
       3.12897090e-01, 1.46942725e-01], [-
       7.98950822e-01, 2.62831011e-01], [-
       3.12897090e-01,  -5.77359062e-01],  [-
       1.09058306e+00, -1.44652121e+00],  [-
       7.01740076e-01, -1.59138156e+00],  [-
       2.15686344e-01, 2.14601566e+00], [-
       1.96547978e+00,  -5.58617754e-02],  [
       8.53631867e-01,  -7.80163563e-01],
       [-7.98950822e-01, -6.06331134e-01], [-
       9.93372315e-01,  -4.32498705e-01],  [-
       1.18475597e-01,  -4.32498705e-01],  [
       7.59458956e-02,   2.04886868e-01],
       [-1.77105829e+00,  4.65635512e-01],
       [-6.04529329e-01,  1.36376973e+00],
       [-1.18475597e-01,  2.04886868e-01],
       [-1.86826903e+00,  4.36663440e-01],
       [ 1.63131784e+00,  1.74040666e+00],
       [-3.12897090e-01,   -1.38857706e+00],
       [-3.12897090e-01,  -6.64275277e-01],  [
       8.53631867e-01,  2.14601566e+00], [
       2.70367388e-01, -5.48386991e-01], [
       8.53631867e-01,   1.01610487e+00],
       [-1.47942605e+00,  -1.21474464e+00],
       [ 1.04805336e+00,  2.05909944e+00],
```

```
        [-9.93372315e-01,  4.94607583e-01],
        [-8.96161568e-01,  2.91803083e-01],
        [-1.18475597e-01, -2.29694204e-01], [-
        6.04529329e-01, 4.65635512e-01], [-
        1.67384754e+00, 5.23579655e-01], [-
        1.18475597e-01,  2.62831011e-01], [
        1.82573933e+00,  -2.87638347e-01],  [-
        1.18475597e-01,  -4.90442848e-01],  [-
        1.38221530e+00, -3.45582490e-01],  [-
        1.96547978e+00, -5.19414919e-01],  [-
        1.57663679e+00, 3.20775154e-01], [-
        4.10107836e-01,  -7.80163563e-01],  [-
        7.01740076e-01,   -1.04091221e+00],  [
        1.04805336e+00,  -9.82968063e-01],  [-
        1.09058306e+00,  5.23579655e-01], [
        2.70367388e-01,   -5.19414919e-01],
        [-1.09058306e+00,  4.07691369e-01],
        [-3.12897090e-01,  -1.44652121e+00], [
        4.64788881e-01,   1.21890937e+00],
        [-1.09058306e+00,    -3.45582490e-01],
        [-1.18475597e-01,  2.91803083e-01],
        [ 1.33968560e+00,  5.81523798e-01],
        [-1.18779381e+00,  -1.15680049e+00], [
        1.04805336e+00,  4.65635512e-01], [
        1.82573933e+00,   1.50863009e+00],
        [-4.10107836e-01,    -1.30166085e+00],
        [-3.12897090e-01, -3.74554562e-01], [-
        4.10107836e-01,  1.30582558e+00], [
        2.02016082e+00,  5.23579655e-01], [
        6.59210374e-01,  -1.09885635e+00],  [-
        8.96161568e-01, 3.78719297e-01], [-
        1.18779381e+00,  2.91803083e-01], [
        1.04805336e+00, -1.21474464e+00],  [-
        1.47942605e+00,  -1.44652121e+00],  [-
        6.04529329e-01,   -1.50446535e+00],  [
        2.11737157e+00,  -8.09135634e-01],  [-
        1.86826903e+00, 1.75914797e-01], [-
        2.15686344e-01, 8.42272441e-01], [-
        1.86826903e+00,   -1.27268878e+00],  [
        2.11737157e+00,    3.78719297e-01],
        [-1.38221530e+00,  5.52551726e-01],
        [-1.09058306e+00,  -3.45582490e-01], [
        1.73156642e-01, -6.64275277e-01], [
        3.67578135e-01,    2.08236764e-03],
        [-6.04529329e-01,  2.31984809e+00],
        [-3.12897090e-01,  2.04886868e-01],
        [-1.57663679e+00,  -2.00722133e-01], [
        6.59210374e-01,  -1.38857706e+00],  [-
        1.09058306e+00,    5.52551726e-01],
```

```
       [-1.96547978e+00,  3.49747226e-01],
       [ 3.67578135e-01,  2.62831011e-01], [
       1.73156642e-01, -2.87638347e-01], [
       1.43689635e+00, -1.04091221e+00], [
       8.53631867e-01,  1.07404901e+00]])
```

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(xtrain, ytrain)
```

```
LogisticRegression()
```

```
ytrain_pred = lr.predict(xtrain)
ytest_pred = lr.predict(xtest)
```

```
ytrain_pred
```

```
array([1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
1,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
1,
       0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1,
0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0,
0,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
ytest_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
1,
```

```
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1])
```

```
print(lr.predict( [[19,19000]]))
```

```
[0]
```

```
/home/ubuntu/.local/lib/python3.8/site-packages/sklearn/base.py:420:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```

```
lr.predict([[-7.98950822e-01,4.94607583e-01]])
```

```
array([0])
```

```
lr.predict([[-2.15686344e-01,2.14601566e+00]])
```

```
array([1])
```

```
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score
matrix = confusion_matrix(ytest,ytest_pred)
print(matrix)
```

```
[[57  1]
 [ 5 17]]
```

```
score=accuracy_score(ytest,ytest_pred)
score
```

```
0.925
```

```
cr=classification_report(ytest,ytest_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95        58
           1       0.94      0.77      0.85        22

    accuracy                           0.93        80
   macro avg       0.93      0.88      0.90        80
weighted avg       0.93      0.93      0.92        80
```

```
pip install seaborn
```

```
Defaulting to user installation because normal site-packages is not
writeable
```

```
Requirement already satisfied: seaborn in
/home/ubuntu/.local/lib/python3.8/site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in
/home/ubuntu/.local/lib/python3.8/site-packages (from seaborn)
(1.24.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in
/home/ubuntu/.local/lib/python3.8/site-packages (from seaborn) (3.7.0)
Requirement already satisfied: pandas>=0.25 in
/home/ubuntu/.local/lib/python3.8/site-packages (from seaborn) (1.5.3)
Requirement already satisfied: fonttools>=4.22.0 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (4.38.0)
Requirement already satisfied: cycler>=0.10 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (5.7.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (3.0.8)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (21.3)
Requirement already satisfied: contourpy>=1.0.1 in
/home/ubuntu/.local/lib/python3.8/site-packages (from matplotlib!
=3.6.1,>=3.1->seaborn) (1.0.7)
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (7.0.0)
Requirement already satisfied: pytz>=2020.1 in
/home/ubuntu/.local/lib/python3.8/site-packages (from pandas>=0.25-
>seaborn) (2022.7.1)
Requirement already satisfied: zipp>=3.1.0 in
/home/ubuntu/.local/lib/python3.8/site-packages (from importlib-
resources>=3.2.0->matplotlib!=3.6.1,>=3.1->seaborn) (3.8.0)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn)
(1.14.0)
WARNING: You are using pip version 22.0.4; however, version 23.0.1 is
available.
You should consider upgrading via the '/usr/bin/python3 -m pip install
--upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

```
import seaborn as sns
sns.heatmap(matrix,annot=True)
```

```
<Axes: >
```


```
tn, fp, fn, tp = confusion_matrix(ytest,ytest_pred).ravel()
print(tn, fp, fn, tp)
```

```
57 1 5 17
```

```
print("Error rate:",(fp+fn)/(tn+fp+fn+tp))
```

```
Error rate: 0.075
```

Assignment No:6

Rollno:TC021F035

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, precision_score, recall_score, f1_score
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

```
ModuleNotFoundError                       Traceback (most recent call
last)
Cell In[3], line 9
      7 from sklearn.model_selection import train_test_split
      8 from sklearn.naive_bayes import GaussianNB
----> 9 from mlxtend.plotting import plot_confusion_matrix
     10 from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, precision_score, recall_score, f1_score
     11 import warnings

ModuleNotFoundError: No module named 'mlxtend'
```

```
pip install sklearn
```

```
Collecting sklearn
  Downloading sklearn-0.0.post12.tar.gz (2.6 kB)
  Preparing metadata (setup.py) ... error: subprocess-exited-with-
error

  × python setup.py egg_info did not run successfully.
  | exit code: 1
  ╰─> [15 lines of output]
      The 'sklearn' PyPI package is deprecated, use 'scikit-learn'
      rather than 'sklearn' for pip commands.

      Here is how to fix this error in the main use cases:
      - use 'pip install scikit-learn' rather than 'pip install
sklearn'
      - replace 'sklearn' by 'scikit-learn' in your pip requirements
files
        (requirements.txt, setup.py, setup.cfg, Pipfile, etc ...)
      - if the 'sklearn' package is used by one of your dependencies,
        it would be great if you take some time to track which package
```

```
uses
        'sklearn' instead of 'scikit-learn' and report it to their
issue tracker
      - as a last resort, set the environment variable
        SKLEARN_ALLOW_DEPRECATED_SKLEARN_PACKAGE_INSTALL=True to avoid
this error

      More information is available at
      https://github.com/scikit-learn/sklearn-pypi-package
      [end of output]

  note: This error originates from a subprocess, and is likely not a
problem with pip.
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
ay need to restart the kernel to use updated packages.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, precision_score, recall_score, f1_score
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline

---------------------------------------------------------------------------
------
ModuleNotFoundError                       Traceback (most recent call
last)
Cell In[4], line 9
      7 from sklearn.model_selection import train_test_split
      8 from sklearn.naive_bayes import GaussianNB
----> 9 from mlxtend.plotting import plot_confusion_matrix
     10 from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, precision_score, recall_score, f1_score
     11 import warnings

ModuleNotFoundError: No module named 'mlxtend'
```

```
pip install mlxtend

Collecting mlxtend
  Downloading mlxtend-0.23.1-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in
/Applications/anaconda3/lib/python3.11/site-packages (from mlxtend)
(1.11.4)
Requirement already satisfied: numpy>=1.16.2 in
/Applications/anaconda3/lib/python3.11/site-packages (from mlxtend)
(1.26.4)
Requirement already satisfied: pandas>=0.24.2 in
/Applications/anaconda3/lib/python3.11/site-packages (from mlxtend)
(2.1.4)
Requirement already satisfied: scikit-learn>=1.0.2 in
/Applications/anaconda3/lib/python3.11/site-packages (from mlxtend)
(1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in
/Applications/anaconda3/lib/python3.11/site-packages (from mlxtend)
(3.8.0)
Requirement already satisfied: joblib>=0.13.2 in
/Applications/anaconda3/lib/python3.11/site-packages (from mlxtend)
(1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (23.1)
Requirement already satisfied: pillow>=6.2.0 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/Applications/anaconda3/lib/python3.11/site-packages (from
matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/Applications/anaconda3/lib/python3.11/site-packages (from
pandas>=0.24.2->mlxtend) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
```

```
/Applications/anaconda3/lib/python3.11/site-packages (from
pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Applications/anaconda3/lib/python3.11/site-packages (from scikit-
learn>=1.0.2->mlxtend) (2.2.0)
Requirement already satisfied: six>=1.5 in
/Applications/anaconda3/lib/python3.11/site-packages (from python-
dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Downloading mlxtend-0.23.1-py3-none-any.whl (1.4 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 212.9 kB/s eta
0:00:0000:0100:01
lxtend
Successfully installed mlxtend-0.23.1
Note: you may need to restart the kernel to use updated packages.
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report, precision_score, recall_score, f1_score
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

```
iris = load_iris()
iris.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR',
'feature_names', 'filename', 'data_module'])
```

```
x = pd.DataFrame(iris['data'], columns=iris['feature_names'])
y = pd.DataFrame(iris['target'], columns=['target'])
```

```
x.head()
```

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

```
4                  5.0                3.6                1.4
0.2
```

```
x.shape, y.shape
```

```
((150, 4), (150, 1))
```

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #    Column             Non-Null Count  Dtype  
---   ------             --------------  -----  
 0    sepal length (cm)  150 non-null    float64
 1    sepal width (cm)   150 non-null    float64
 2    petal length (cm)  150 non-null    float64
 3    petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

```
y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 1 columns):
 #    Column  Non-Null Count  Dtype
---   ------  --------------  -----
 0    target  150 non-null    int64
dtypes: int64(1)
memory usage: 1.3 KB
```

```
x.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	\
count	150.000000	150.000000	150.000000	
mean	5.843333	3.057333	3.758000	
std	0.828066	0.435866	1.765298	
min	4.300000	2.000000	1.000000	
25%	5.100000	2.800000	1.600000	
50%	5.800000	3.000000	4.350000	
75%	6.400000	3.300000	5.100000	
max	7.900000	4.400000	6.900000	

	petal width (cm)
count	150.000000
mean	1.199333
std	0.762238
min	0.100000
25%	0.300000
50%	1.300000

```
75%                 1.800000
max                 2.500000
```

```
scaler = StandardScaler()
x = scaler.fit_transform(x.values)

x_train, x_test, y_train, y_test = train_test_split(x, y.values,
test_size=0.2, random_state=42)

x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((120, 4), (30, 4), (120, 1), (30, 1))
```

```
model = GaussianNB()

model.fit(x_train, y_train)
```

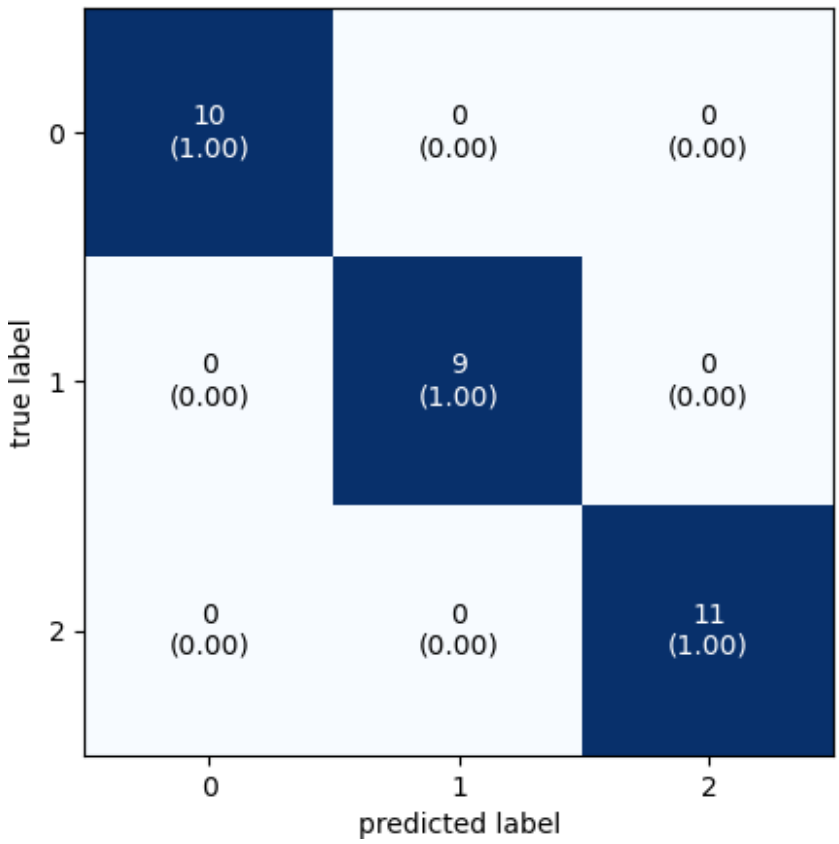
```
GaussianNB()
```

```
y_pred = model.predict(x_test)

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
plot_confusion_matrix(conf_mat=cm, figsize=(5,5), show_normed=True)
plt.show()
```




```
print(f"TP value is {cm[0,0]}")
print(f"TN value is {cm[1,1] + cm[2,2]}")
print(f"FP value is {cm[0,1] + cm[0,2]}")
print(f"FN value is {cm[1,0] + cm[2,0]}")
```

```
TP value is 10
TN value is 20
FP value is 0
FN value is 0
```

```
print(f"Accuracy score is {accuracy_score(y_test, y_pred)}")
```

```
Accuracy score is 1.0
```

```
print(f"Error rate is {1 - accuracy_score(y_test, y_pred)}")
```

```
Error rate is 0.0
```

```
print(f"Precision score is {precision_score(y_test, y_pred,
average='macro')}")
```

```
Precision score is 1.0
```

```
print(f"Recall score is {recall_score(y_test, y_pred,
average='macro')}")
```

```
Recall score is 1.0
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Assignment No:7

Rollno:TC021F035

```
import nltk
nltk.download("punkt")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("averaged_perceptron_tagger")
```

```
[nltk_data] Downloading package punkt to /Users/Saroj/nltk_data...
[nltk_data]     Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /Users/Saroj/nltk_data...
[nltk_data]     Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /Users/Saroj/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]       /Users/Saroj/nltk_data...
[nltk_data]     Unzipping taggers/averaged_perceptron_tagger.zip.
```

```
True
```

```
from nltk import word_tokenize, sent_tokenize

corpus = "Sachin was the GOAT of the previous generation. Virat is the GOAT of this generation. Shubman will be the GOAT of the next generation"

print(word_tokenize(corpus))
print(sent_tokenize(corpus))
```

```
['Sachin', 'was', 'the', 'GOAT', 'of', 'the', 'previous', 'generation', '.', 'Virat', 'is', 'the', 'GOAT', 'of', 'this', 'generation', '.', 'Shubman', 'will', 'be', 'the', 'GOAT', 'of', 'the', 'next', 'generation']
['Sachin was the GOAT of the previous generation.', 'Virat is the GOAT of this generation.', 'Shubman will be the GOAT of the next generation']
```

```
from nltk import pos_tag

tokens = word_tokenize(corpus)
print(pos_tag(tokens))
```

```
[('Sachin', 'NNP'), ('was', 'VBD'), ('the', 'DT'), ('GOAT', 'NNP'), ('of', 'IN'), ('the', 'DT'), ('previous', 'JJ'), ('generation', 'NN'), ('.', '.'), ('Virat', 'NNP'), ('is', 'VBZ'), ('the', 'DT'), ('GOAT', 'NNP'), ('of', 'IN'), ('this', 'DT'), ('generation', 'NN'), ('.', '.'), ('Shubman', 'NNP'), ('will', 'MD'), ('be', 'VB'), ('the', 'DT'), ('GOAT', 'NNP'), ('of', 'IN'), ('the', 'DT'), ('next', 'JJ'), ('generation', 'NN')]
```

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))

tokens = word_tokenize(corpus)
cleaned_tokens = []
```

```
for token in tokens:
    if (token not in stop_words):
        cleaned_tokens.append(token)
print(cleaned_tokens)
```

```
['Sachin', 'GOAT', 'previous', 'generation', '.', 'Virat', 'GOAT',
'generation', '.', 'Shubman', 'GOAT', 'next', 'generation']
```

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

stemmed_tokens = []
for token in cleaned_tokens:
    stemmed = stemmer.stem(token)
    stemmed_tokens.append(stemmed)
print(stemmed_tokens)
```

```
['sachin', 'goat', 'previou', 'gener', '.', 'virat', 'goat', 'gener',
'.', 'shubman', 'goat', 'next', 'gener']
```

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

lemmatized_tokens = []
for token in cleaned_tokens:
    lemmatized = lemmatizer.lemmatize(token)
    lemmatized_tokens.append(lemmatized)
print(lemmatized_tokens)
```

```
['Sachin', 'GOAT', 'previous', 'generation', '.', 'Virat', 'GOAT',
'generation', '.', 'Shubman', 'GOAT', 'next', 'generation']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
    "Sachin was the GOAT of the previous generation",
    "Virat is the GOAT of the this generation",
    "Shubman will be the GOAT of the next generation"
]

vectorizer = TfidfVectorizer()

matrix = vectorizer.fit(corpus)
matrix.vocabulary_
```

```
{'sachin': 7,
 'was': 12,
 'the': 9,
 'goat': 2,
 'of': 5,
```

```
'previous': 6,
'generation': 1,
'virat': 11,
'is': 3,
'this': 10,
'shubman': 8,
'will': 13,
'be': 0,
'next': 4}
```

Assignment No:8

Rollno:TC021F035

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

dataset = sns.load_dataset('titanic')
dataset.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

    who  adult_male deck  embark_town alive  alone
0   man        True  NaN  Southampton    no  False
1  woman       False    C  Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
dataset.shape
```

```
(891, 15)
```

```
dataset.isnull()
```

```
     survived  pclass    sex    age  sibsp  parch   fare  embarked
class  \
0       False   False  False  False  False  False  False     False
False
1       False   False  False  False  False  False  False     False
False
2       False   False  False  False  False  False  False     False
False
3       False   False  False  False  False  False  False     False
False
4       False   False  False  False  False  False  False     False
False
..        ...     ...    ...    ...    ...    ...    ...       ...
...
886     False   False  False  False  False  False  False     False
False
887     False   False  False  False  False  False  False     False
```

```
False
888     False   False  False   True  False  False  False     False
False
889     False   False  False  False  False  False  False     False
False
890     False   False  False  False  False  False  False     False
False

       who  adult_male   deck  embark_town  alive  alone
0    False        False   True        False  False  False
1    False        False  False        False  False  False
2    False        False   True        False  False  False
3    False        False  False        False  False  False
4    False        False   True        False  False  False
..     ...          ...    ...          ...    ...    ...
886  False        False   True        False  False  False
887  False        False  False        False  False  False
888  False        False   True        False  False  False
889  False        False  False        False  False  False
890  False        False   True        False  False  False

[891 rows x 15 columns]
```

```
dataset.isnull().sum()
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

Assignment No:9

Rollno:TC021F035

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

dataset = sns.load_dataset('titanic')
dataset.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

    who  adult_male deck  embark_town alive  alone
0   man        True  NaN  Southampton    no  False
1  woman       False    C  Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
dataset.shape
```

```
(891, 15)
```

```
dataset.isnull()
```

```
     survived  pclass    sex    age  sibsp  parch   fare  embarked
class  \
0       False   False  False  False  False  False  False     False
False
1       False   False  False  False  False  False  False     False
False
2       False   False  False  False  False  False  False     False
False
3       False   False  False  False  False  False  False     False
False
4       False   False  False  False  False  False  False     False
False
..        ...     ...    ...    ...    ...    ...    ...       ...
...
886     False   False  False  False  False  False  False     False
False
887     False   False  False  False  False  False  False     False
```

```
False
888     False   False  False   True  False  False  False     False
False
889     False   False  False  False  False  False  False     False
False
890     False   False  False  False  False  False  False     False
False

       who  adult_male   deck  embark_town  alive  alone
0    False        False   True        False  False  False
1    False        False  False        False  False  False
2    False        False   True        False  False  False
3    False        False  False        False  False  False
4    False        False   True        False  False  False
..     ...          ...    ...          ...    ...    ...
886  False        False   True        False  False  False
887  False        False  False        False  False  False
888  False        False   True        False  False  False
889  False        False  False        False  False  False
890  False        False   True        False  False  False

[891 rows x 15 columns]
```

```
dataset.isnull().sum()
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

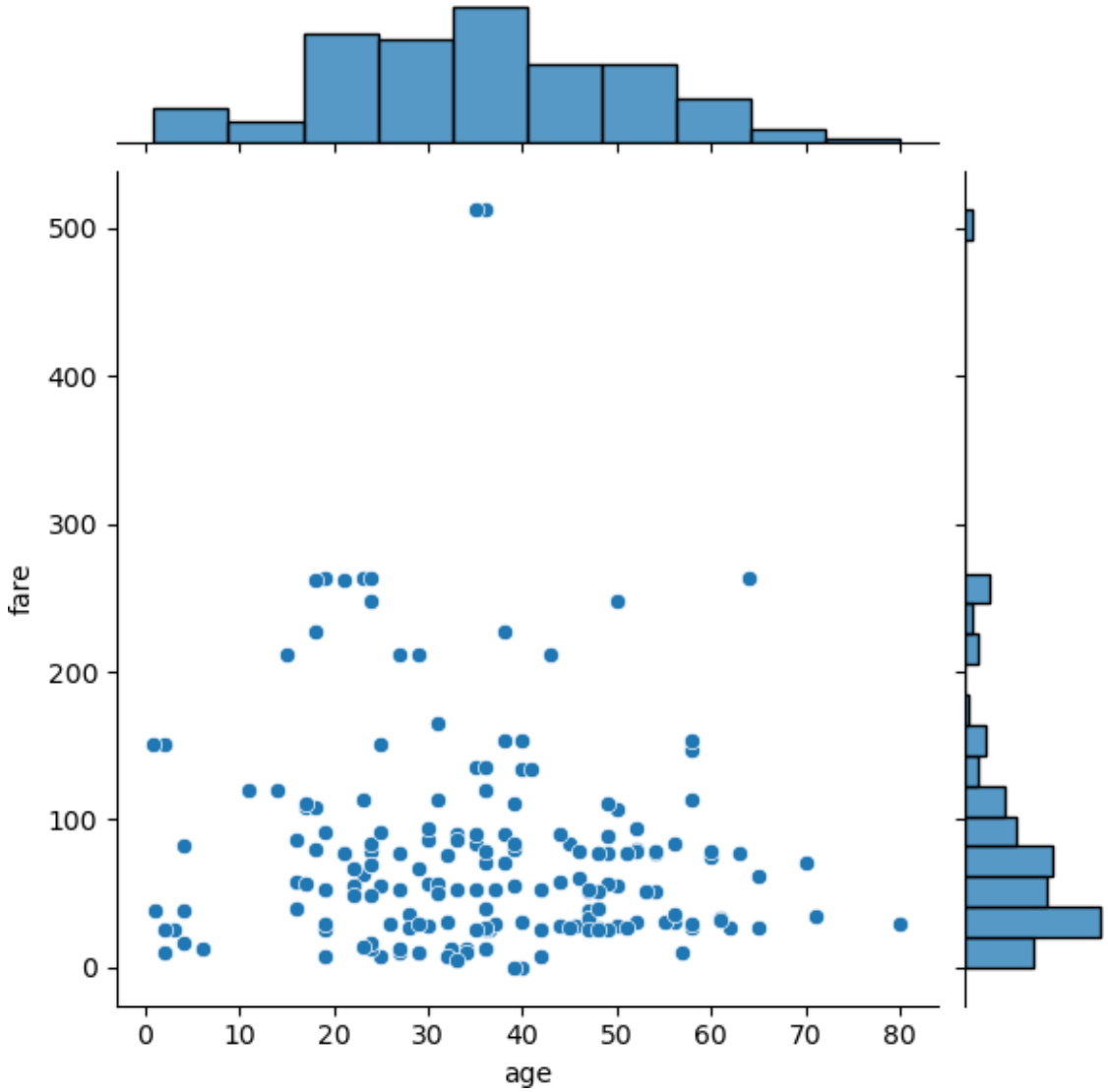
```
dataset = dataset.dropna()
```

```
sns.jointplot(x='age', y='fare', data=dataset)
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
<seaborn.axisgrid.JointGrid at 0x140ed2710>
```

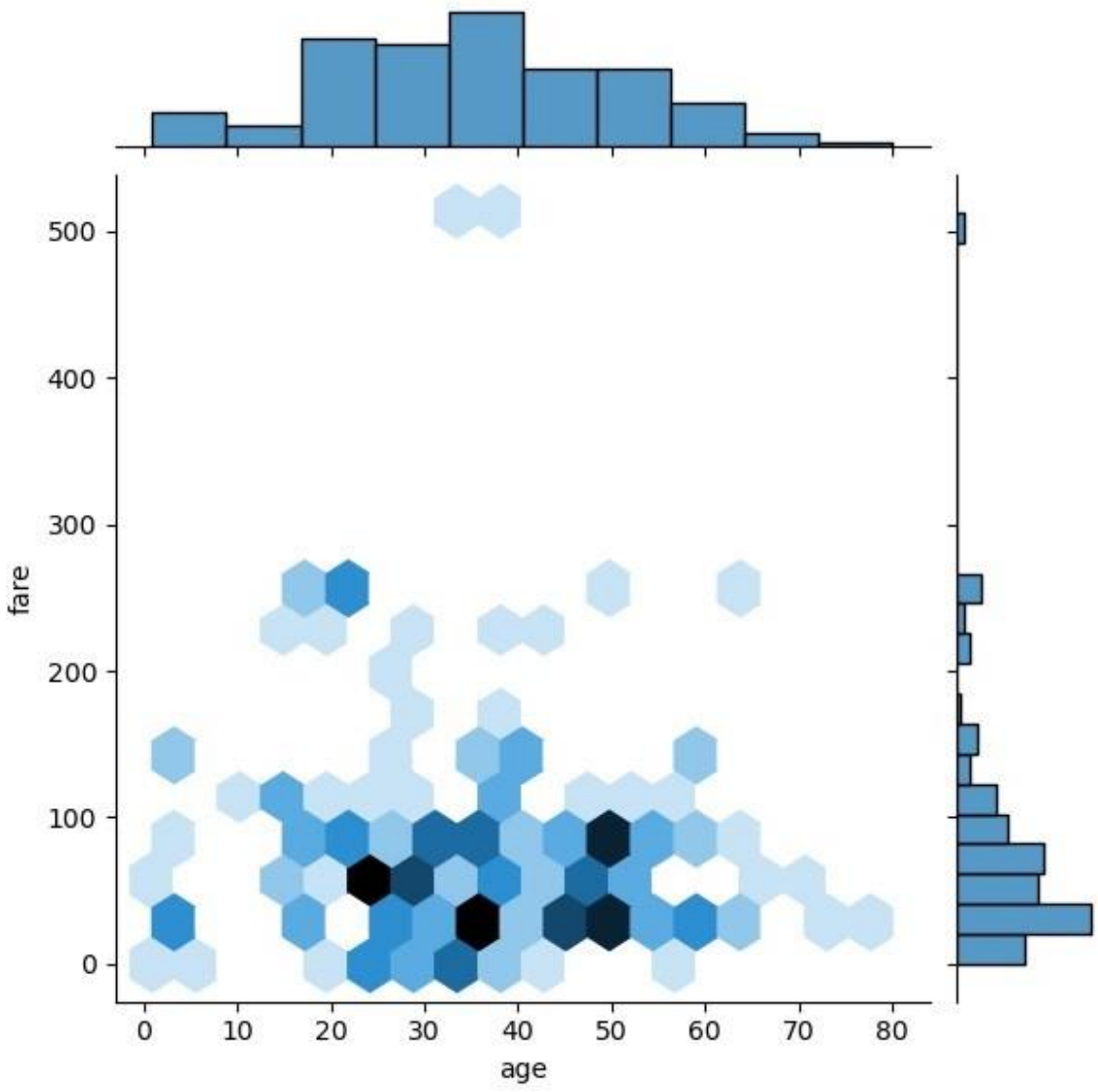



```
sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

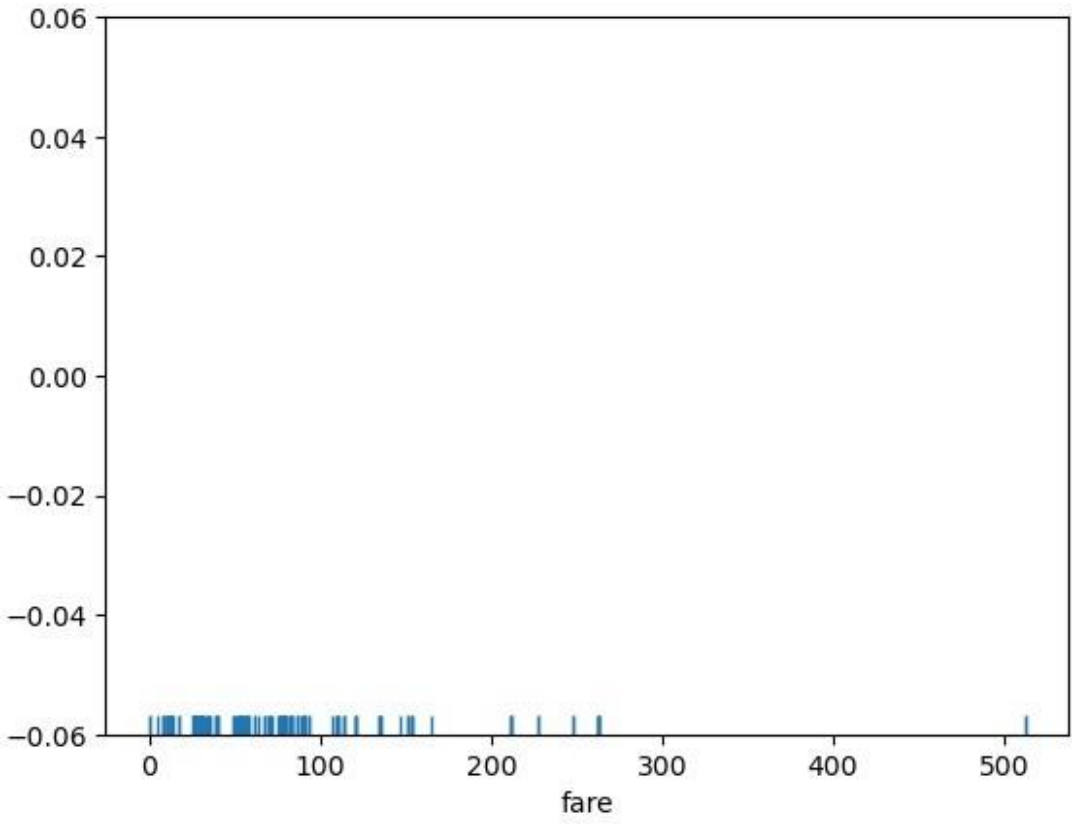
```
<seaborn.axisgrid.JointGrid at 0x140ec2c10>
```



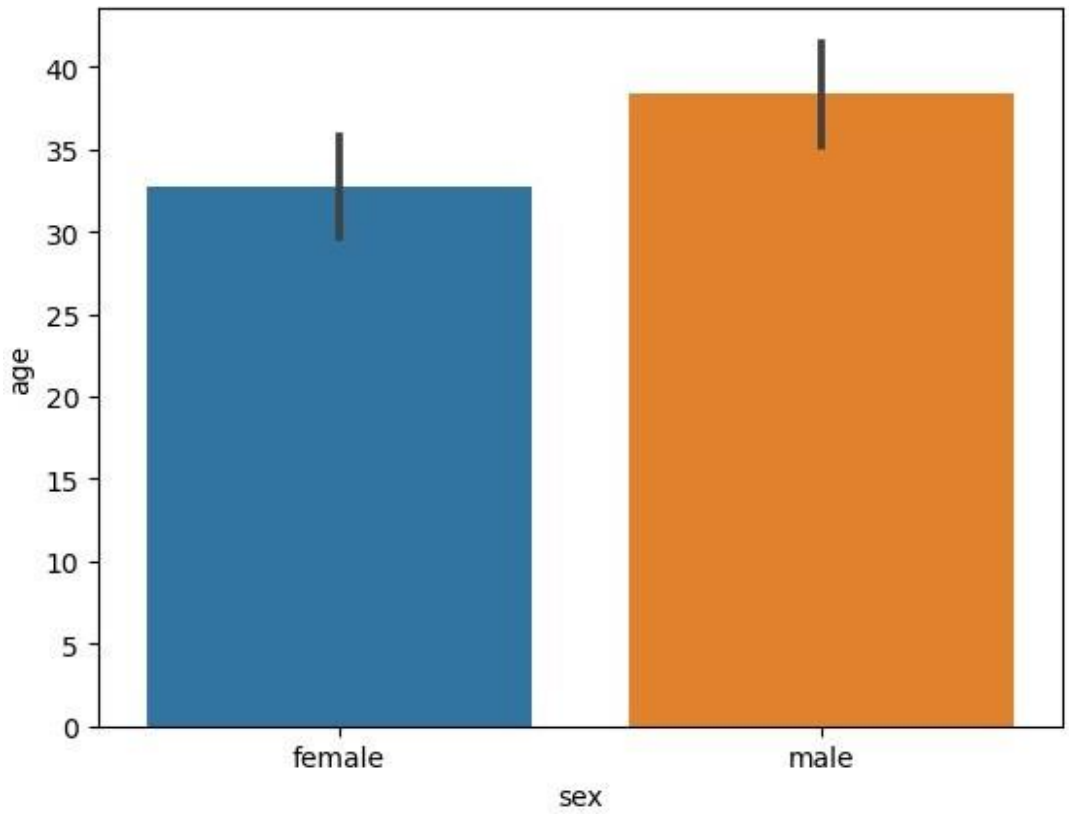

```
sns.rugplot(dataset['fare'])
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
<Axes: xlabel='fare'>
```

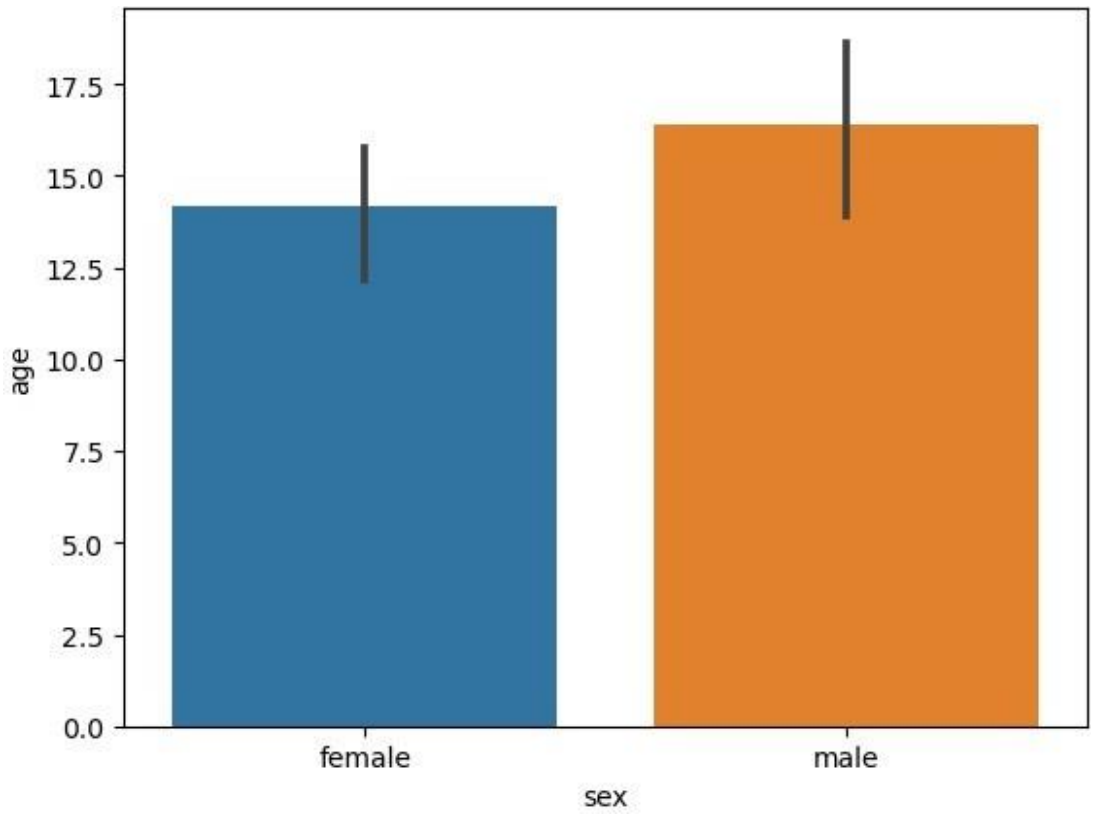



```
sns.barplot(x='sex', y='age', data=dataset)
<Axes: xlabel='sex', ylabel='age'>
```



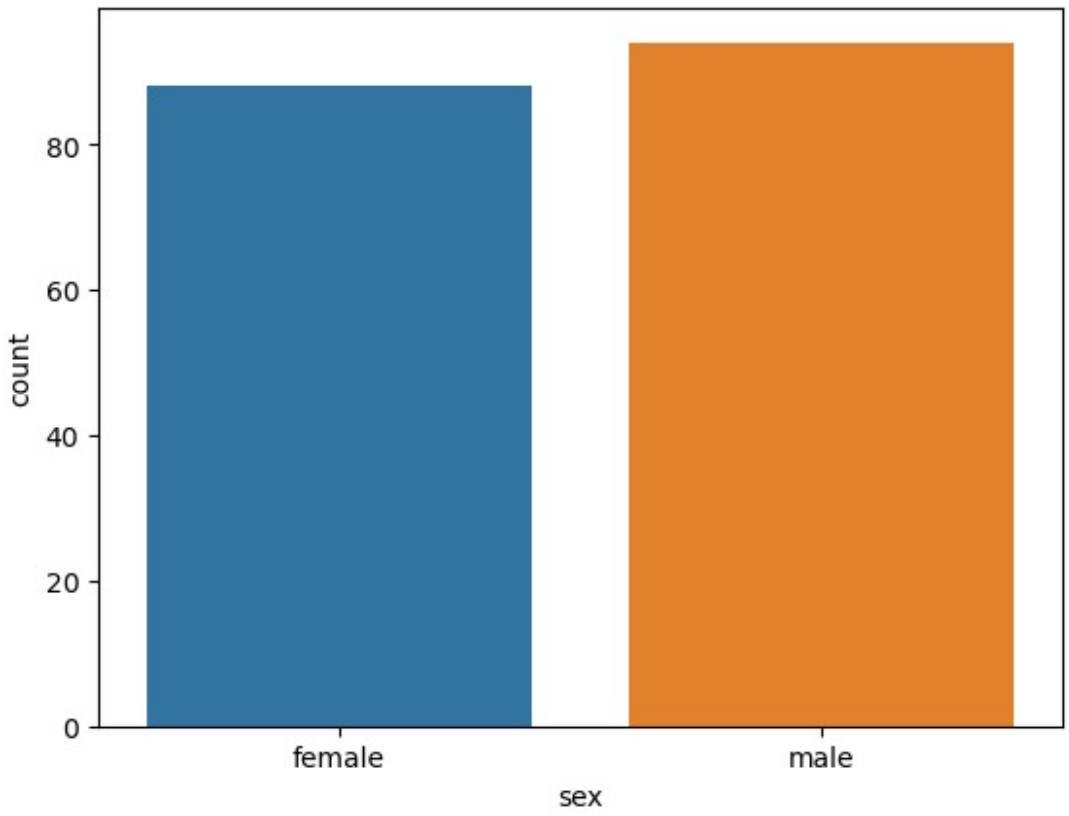

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.barplot(x='sex', y='age', data=dataset, estimator=np.std)
```

```
<Axes: xlabel='sex', ylabel='age'>
```



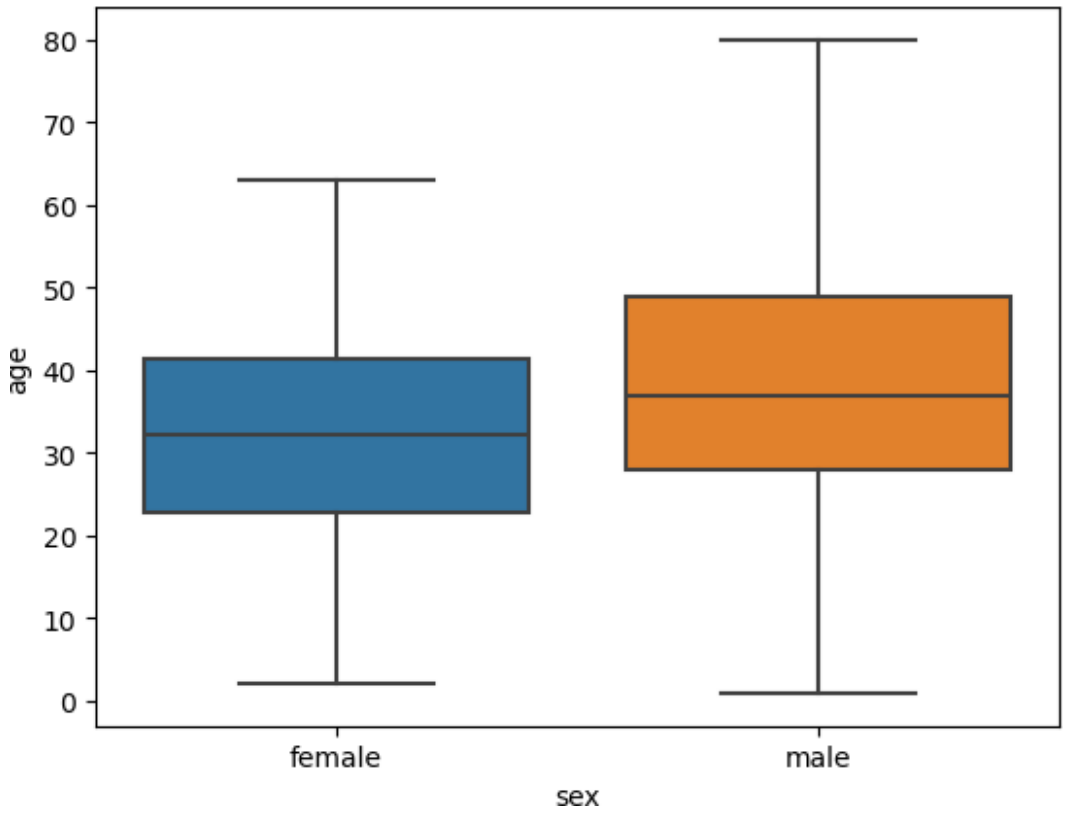

```
sns.countplot(x='sex',  data=dataset)
```

```
<Axes: xlabel='sex', ylabel='count'>
```

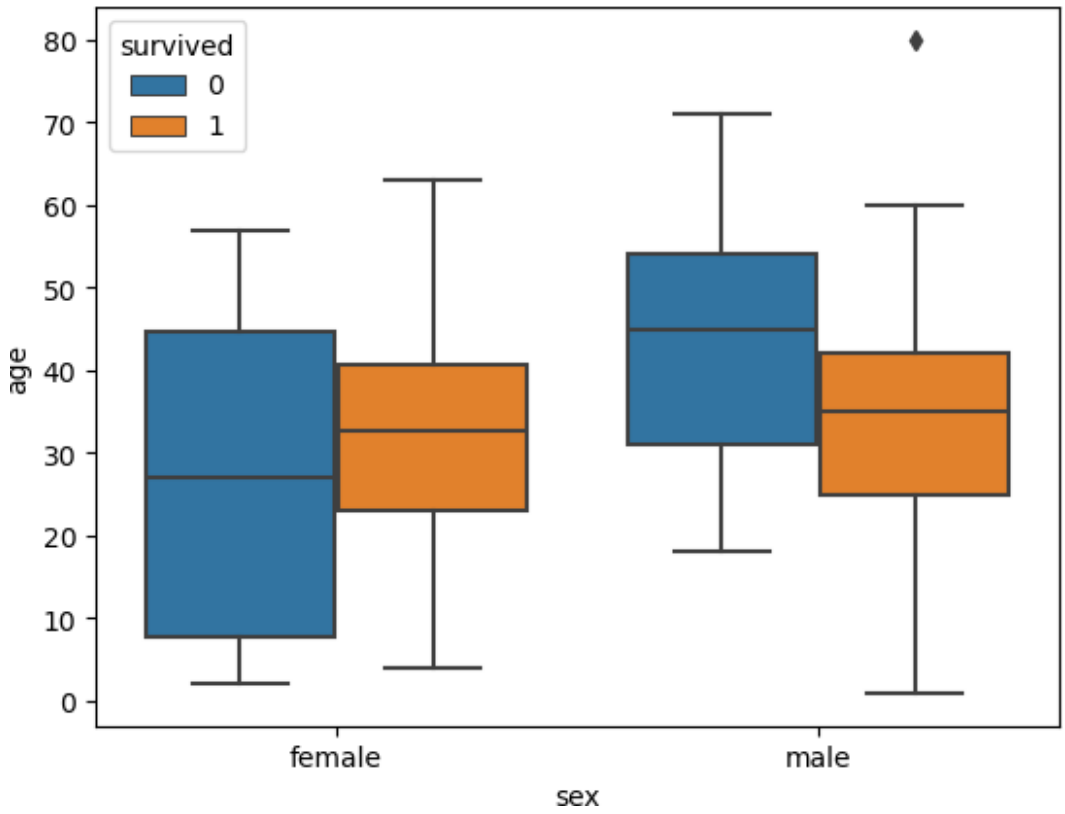



```
sns.boxplot(x='sex', y='age', data=dataset)
```

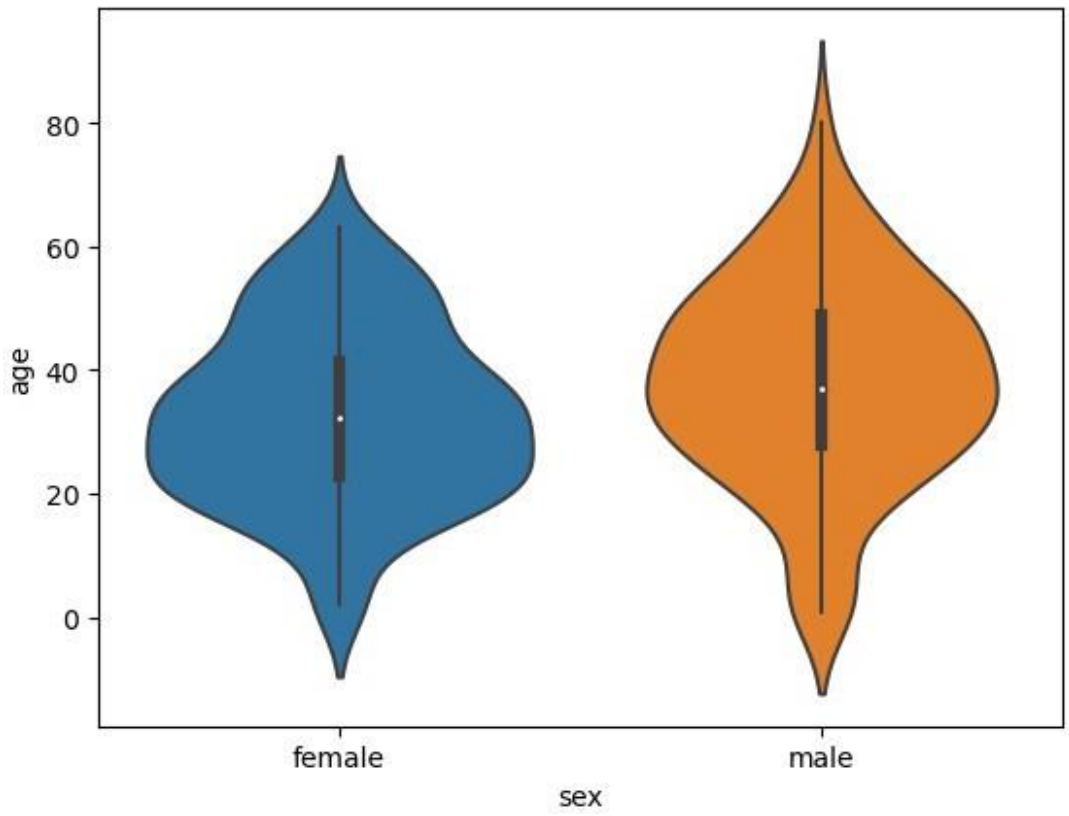
```
<Axes: xlabel='sex', ylabel='age'>
```



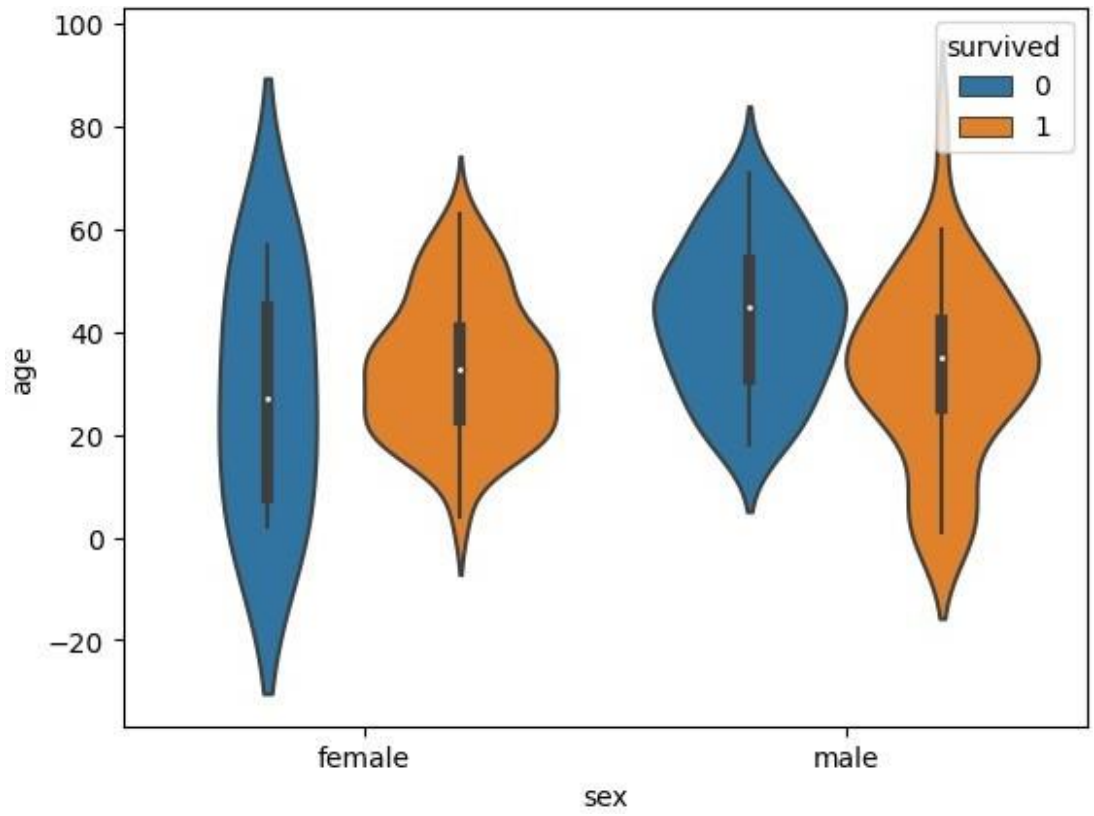

```
sns.boxplot(x='sex', y='age', data=dataset, hue="survived")
<Axes: xlabel='sex', ylabel='age'>
```




```
sns.violinplot(x='sex', y='age', data=dataset)
<Axes: xlabel='sex', ylabel='age'>
```



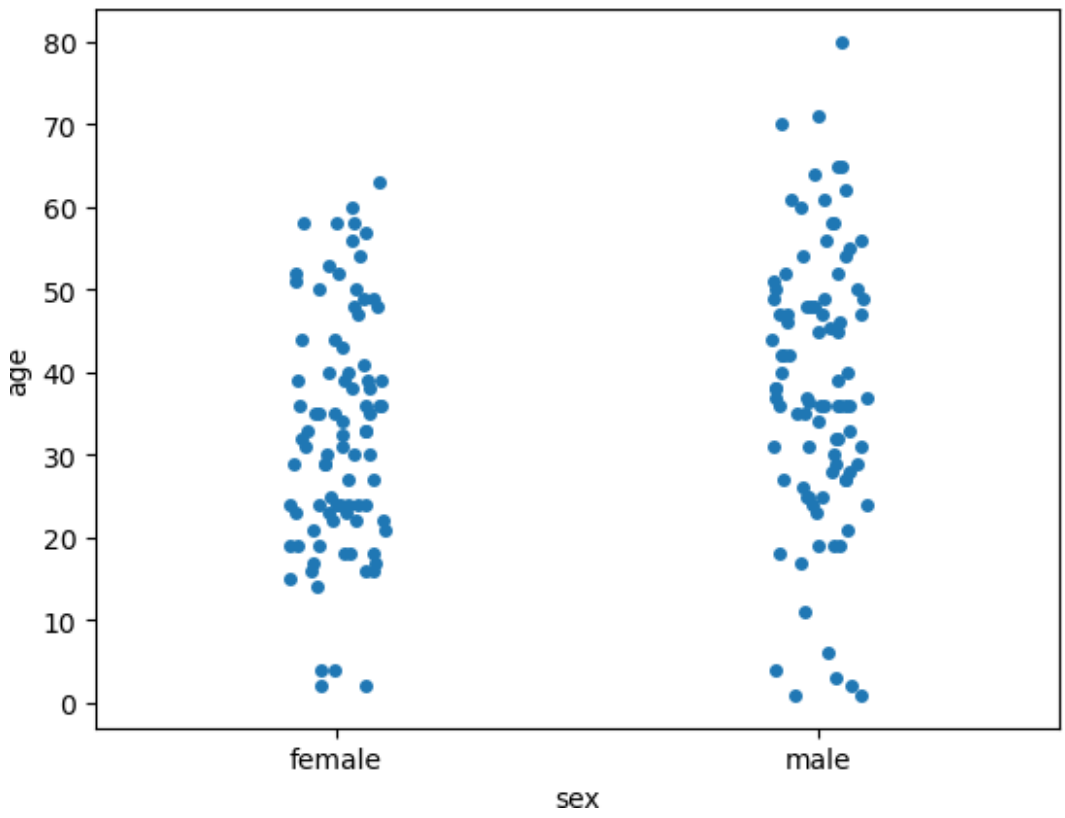

```
sns.violinplot(x='sex', y='age', data=dataset, hue='survived')
<Axes: xlabel='sex', ylabel='age'>
```




```
sns.stripplot(x='sex', y='age', data=dataset)
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.
py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before
operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
<Axes: xlabel='sex', ylabel='age'>
```


80
70
60
50
40
30
20
10
0
age
female
male
sex

Assignment No.10

Rollno: TC021F035

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

Iris = pd.read_csv('./Downloads/IRIS.csv')
Iris
```

```
     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
..            ...          ...           ...          ...
...
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

[150 rows x 5 columns]
```

```
Iris.shape
```

```
(150, 5)
```

```
Iris.describe()
```

```
       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
```

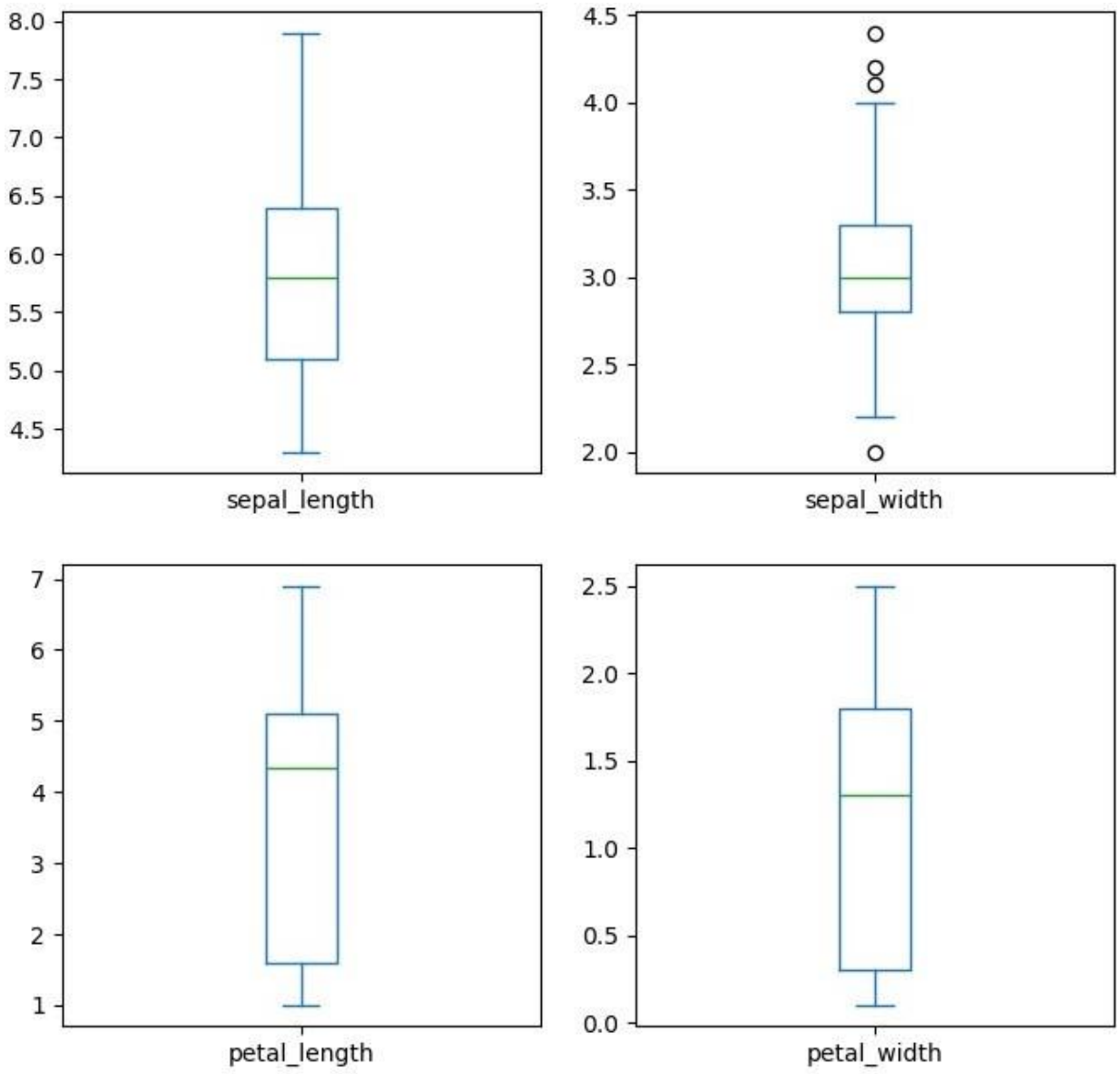
```
Iris.dtypes
```

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

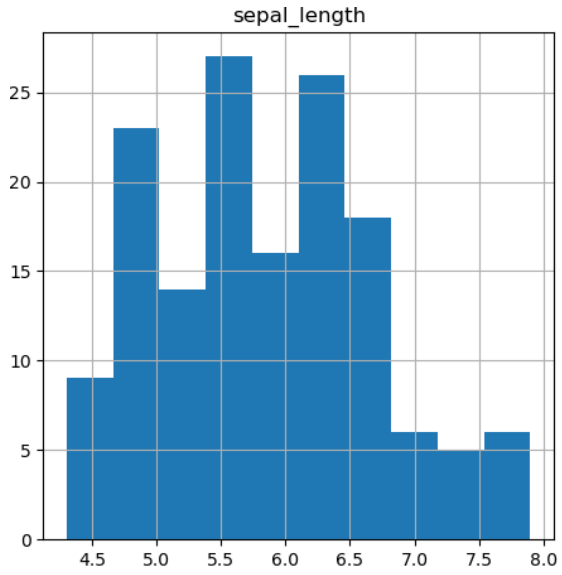
```
Iris.isnull().sum()
```

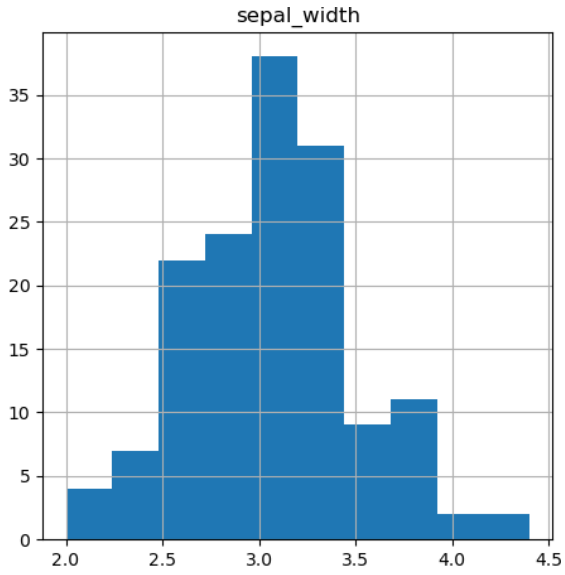
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

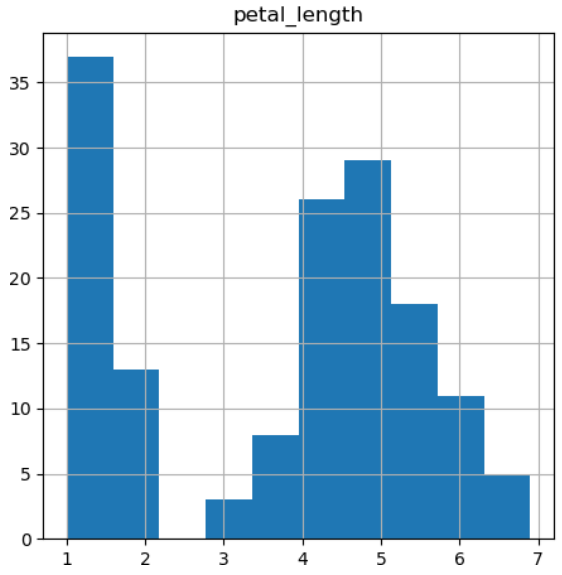
```
Iris.plot(kind='box', subplots=True, layout=(3,2), figsize=(8,12));
```

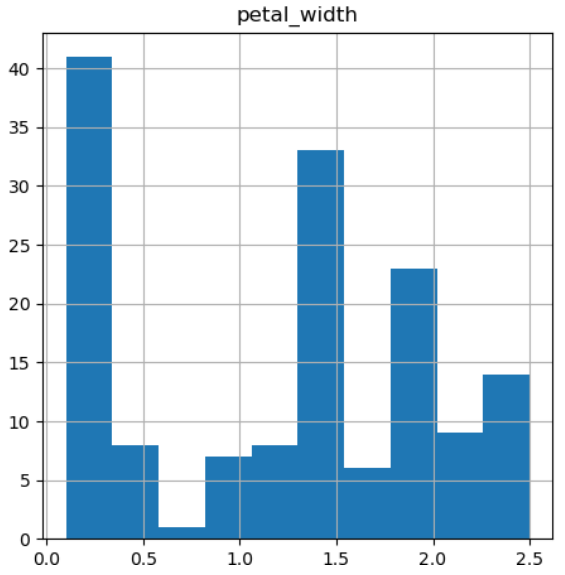



```
Iris.hist(figsize=(12,12))
plt.show()
```





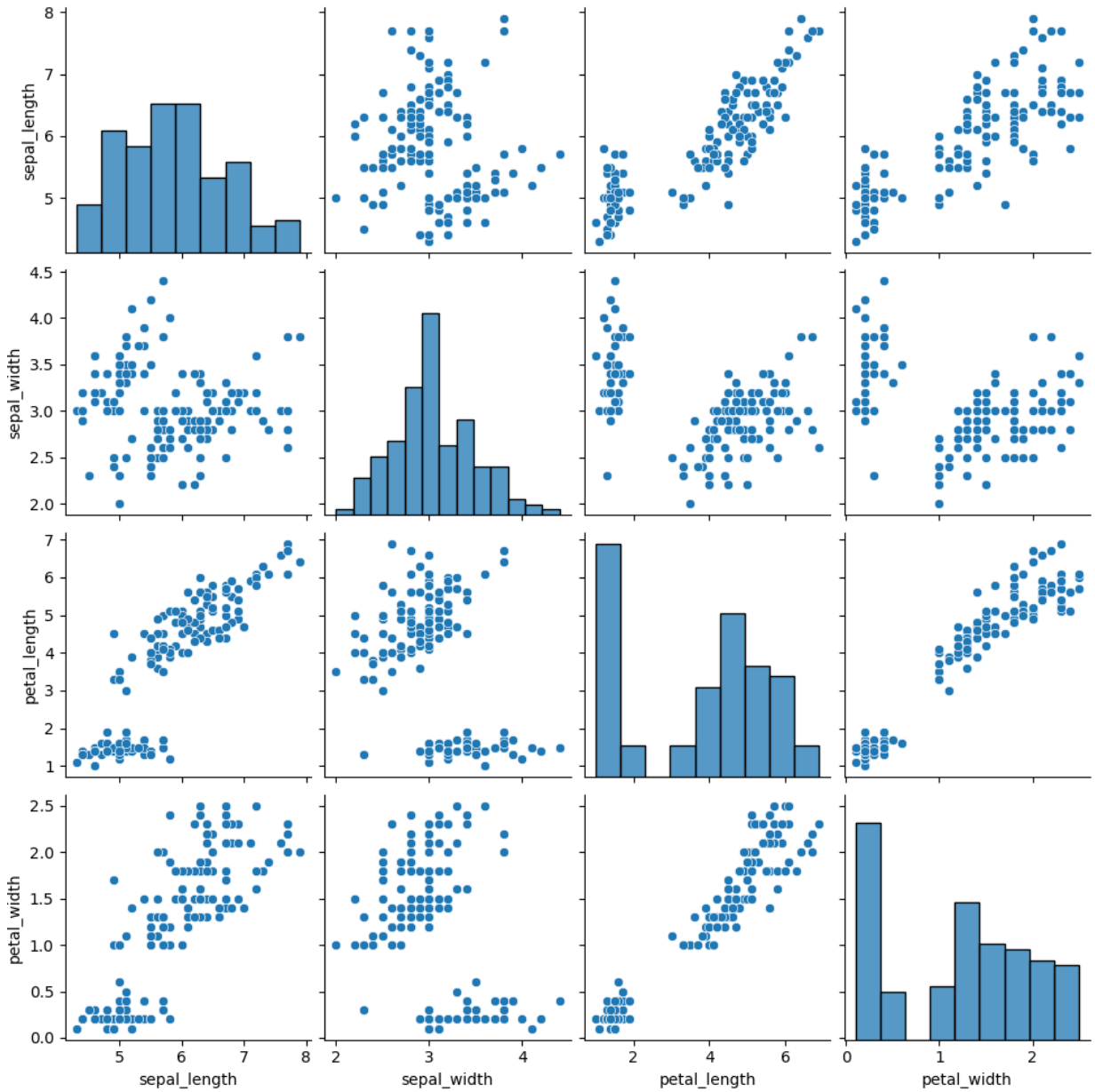





```
sns.pairplot(Iris)
```

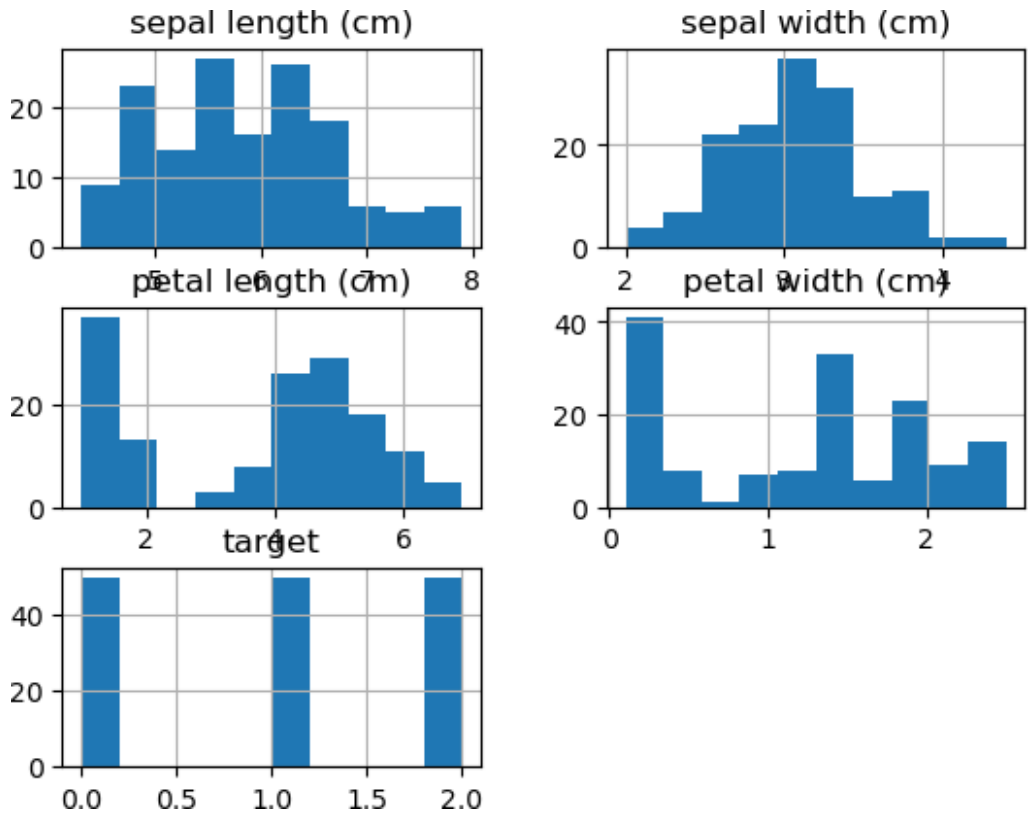
```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/
_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.
py:1119: FutureWarning: use_inf_as_na option is deprecated and will be
removed in a future version. Convert inf values to NaN before
operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

```
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Applications/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
<seaborn.axisgrid.PairGrid at 0x146541390>
```

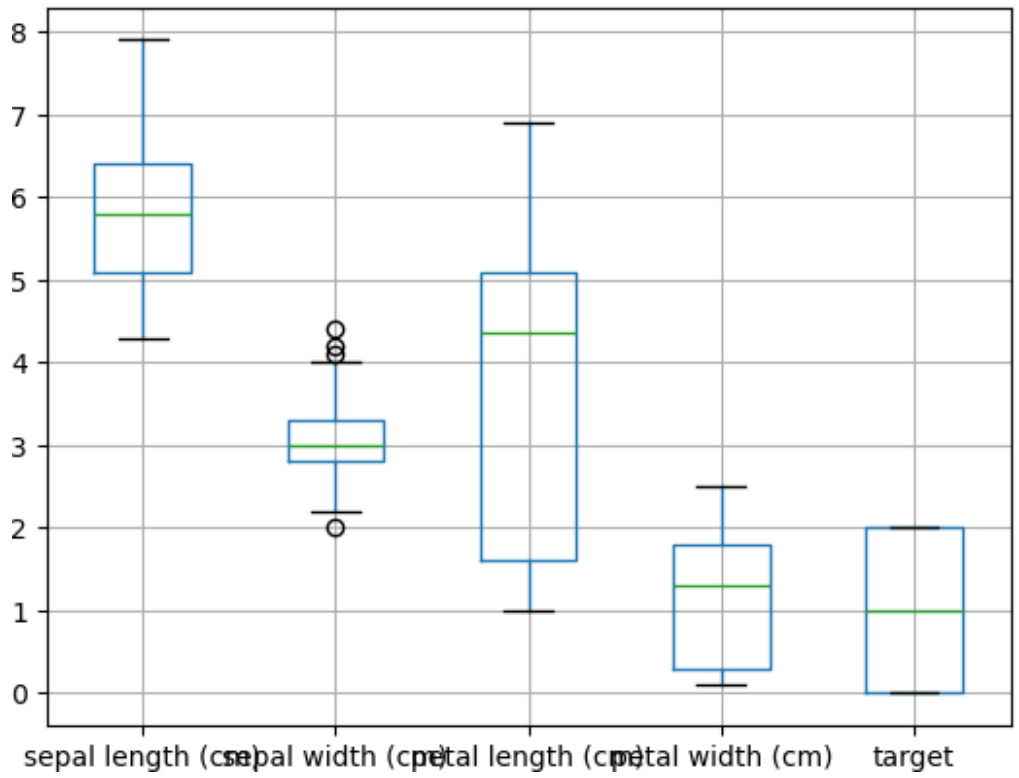


```
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target

 import matplotlib.pyplot as plt
iris_df.hist()
plt.show()
```




```
iris_df.boxplot()
plt.show()
```



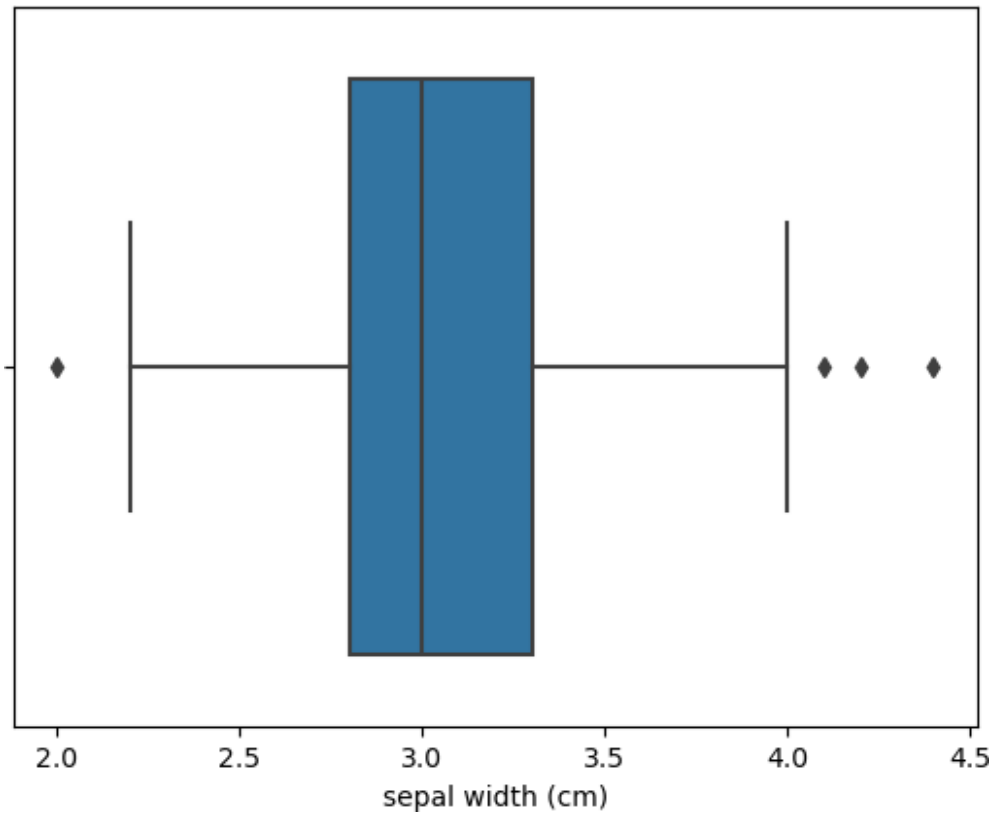

```
iris_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	\
count	150.000000	150.000000	150.000000	
mean	5.843333	3.057333	3.758000	
std	0.828066	0.435866	1.765298	
min	4.300000	2.000000	1.000000	
25%	5.100000	2.800000	1.600000	
50%	5.800000	3.000000	4.350000	
75%	6.400000	3.300000	5.100000	
max	7.900000	4.400000	6.900000	

	petal width (cm)	target
count	150.000000	150.000000
mean	1.199333	1.000000
std	0.762238	0.819232
min	0.100000	0.000000
25%	0.300000	0.000000
50%	1.300000	1.000000
75%	1.800000	2.000000
max	2.500000	2.000000

```
sns.boxplot(x = 'sepal width (cm)', data = iris_df)
```

```
<Axes: xlabel='sepal width (cm)'>
```


2.0
2.5
3.0
3.5
4.0
4.5
sepal width (cm)