```
!pip install linalg
```

```
Requirement already satisfied: linalg in /usr/local/lib/python3.10/dist-packages (1.0.4)
```

```
import numpy as np
import linalg as la
```

1. Create 5 matrices with five different dimensions (1-D,2-D,...5-D)

```
mt1=np.arange(5)
print(mt1)
```

```
[0 1 2 3 4]
```

```
mt2=np.arange(4).reshape(2,2)
mt2
```

```
array([[0, 1],
       [2, 3]])
```

```
mat3=np.arange(1,28)
mt3=mat3.reshape(3,3,3)
mt3
```

```
array([[[ 1,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]],

       [[19, 20, 21],
        [22, 23, 24],
        [25, 26, 27]]])
```

```
mat2=np.arange(7,71)
mt4=mat2.reshape(4,4,4)
mt4
```

```
array([[[ 7,  8,  9, 10],
        [11, 12, 13, 14],
        [15, 16, 17, 18],
        [19, 20, 21, 22]],

       [[23, 24, 25, 26],
        [27, 28, 29, 30],
        [31, 32, 33, 34],
        [35, 36, 37, 38]],

       [[39, 40, 41, 42],
        [43, 44, 45, 46],
        [47, 48, 49, 50],
        [51, 52, 53, 54]],

       [[55, 56, 57, 58],
        [59, 60, 61, 62],
        [63, 64, 65, 66],
        [67, 68, 69, 70]]])
```

```
mat3=np.arange(2,127)
mt5=mat3.reshape(5,5,5)
mt5
```

```
array([[[  2,   3,   4,   5,   6],
        [  7,   8,   9,  10,  11],
        [ 12,  13,  14,  15,  16],
        [ 17,  18,  19,  20,  21],
        [ 22,  23,  24,  25,  26]],

       [[ 27,  28,  29,  30,  31],
        [ 32,  33,  34,  35,  36],
        [ 37,  38,  39,  40,  41],
        [ 42,  43,  44,  45,  46],
        [ 47,  48,  49,  50,  51]],

       [[ 52,  53,  54,  55,  56],
        [ 57,  58,  59,  60,  61],
        [ 62,  63,  64,  65,  66],
```

```
            [ 67,  68,  69,  70,  71],
            [ 72,  73,  74,  75,  76]],

           [[ 77,  78,  79,  80,  81],
            [ 82,  83,  84,  85,  86],
            [ 87,  88,  89,  90,  91],
            [ 92,  93,  94,  95,  96],
            [ 97,  98,  99, 100, 101]],

           [[102, 103, 104, 105, 106],
            [107, 108, 109, 110, 111],
            [112, 113, 114, 115, 116],
            [117, 118, 119, 120, 121],
            [122, 123, 124, 125, 126]]])
```

2. Find determinants of 5 matrices and display your output

```
print(np.linalg.det(mt2))
print(np.linalg.det(mt3))
print(np.linalg.det(mt4))
print(np.linalg.det(mt5))
```

```
    -2.0
    [ 0.00000000e+00  0.00000000e+00 -2.13162821e-14]
    [7.57306469e-29 0.00000000e+00 6.05845175e-28 0.00000000e+00]
    [ 0.00000000e+00  0.00000000e+00 -2.86985925e-41  0.00000000e+00
      0.00000000e+00]
```

3. Find inverse of the above 5 matrices and display your output

```
print(np.linalg.inv(mt2))
```

```
    [[-1.5  0.5]
     [ 1.   0. ]]
```

4. Find the rank, diagonal and trace of the 5 matrices

Rank

```
print(np.linalg.matrix_rank(mt2))
print(np.linalg.matrix_rank(mt3))
print(np.linalg.matrix_rank(mt4))
print(np.linalg.matrix_rank(mt5))
```

```
    2
    [2 2 2]
    [2 2 2 2]
    [2 2 2 2 2]
```

Diagonal

```
print(np.diag(mt2))
```

```
    [0 3]
```

```
dia=[np.diag(i) for i in mt3]
for j in dia:
  print(j)
```

```
    [1 5 9]
    [10 14 18]
    [19 23 27]
```

```
dia=[np.diag(i) for i in mt4]
for j in dia:
  print(j)
```

```
    [ 7 12 17 22]
    [23 28 33 38]
```

```
     [39 44 49 54]
     [55 60 65 70]]
```

```
dia=[np.diag(i) for i in mt5]
for j in dia:
  print(j)
```

```
     [ 2  8 14 20 26]
     [27 33 39 45 51]
     [52 58 64 70 76]
     [ 77  83  89  95 101]
     [102 108 114 120 126]]
```

5. Find Eigen value and eigen vector for 5 matrices

Eigen Values

```
print(np.linalg.eigvals(mt2))
print(np.linalg.eigvals(mt3))
print(np.linalg.eigvals(mt4))
print(np.linalg.eigvals(mt5))
```

```
     [-0.56155281  3.56155281]
     [[ 1.61168440e+01 -1.11684397e+00 -1.30367773e-15]
      [ 4.24242853e+01 -4.24285286e-01 -8.76087811e-16]
      [ 6.92598907e+01 -2.59890679e-01  3.45459719e-15]]
     [[ 5.93479818e+01 -1.34798181e+00  1.93151193e-15 -1.64649453e-15]
      [ 1.22652251e+02 -6.52250567e-01 -3.43772031e-15  1.42014306e-15]
      [ 1.86429118e+02 -4.29117517e-01 -1.06352174e-14 -1.49718129e-15]
      [ 2.50319591e+02 -3.19591445e-01  1.95941062e-15  4.72236028e-15]]
     [[ 7.34057287e+01+0.00000000e+00j -3.40572874e+00+0.00000000e+00j
        3.80184041e-16+0.00000000e+00j -7.52077843e-16+1.36739955e-15j
       -7.52077843e-16-1.36739955e-15j]
      [ 1.96273731e+02+0.00000000e+00j -1.27373133e+00+0.00000000e+00j
       -1.85991187e-14+0.00000000e+00j  1.07476344e-16+0.00000000e+00j
       -1.06339632e-15+0.00000000e+00j]
      [ 3.20779352e+02+0.00000000e+00j -7.79351908e-01+0.00000000e+00j
       -6.24883107e-15+0.00000000e+00j -5.63570073e-16+4.64014788e-15j
       -5.63570073e-16-4.64014788e-15j]
      [ 4.45561090e+02+0.00000000e+00j -5.61090287e-01+0.00000000e+00j
         1.38994623e-14+6.85999716e-15j  1.38994623e-14-6.85999716e-15j
       -2.68952014e-15+0.00000000e+00j]
      [ 5.70438260e+02+0.00000000e+00j -4.38259524e-01+0.00000000e+00j
         4.78531798e-14+0.00000000e+00j  2.13854617e-15+0.00000000e+00j
       -3.99605509e-16+0.00000000e+00j]]
```

Eigen Vector

```
print(np.linalg.eig(mt2))
print(np.linalg.eig(mt3))
print(np.linalg.eig(mt4))
print(np.linalg.eig(mt5))
```

```
  1.08836554e-01+0.j        ],
 [-4.20798187e-01+0.j        ,  3.23284351e-01+0.j        ,
   4.44568778e-01+0.11325759j,  4.44568778e-01-0.11325759j,
  -1.98220428e-01+0.j        ],
 [-4.45812281e-01+0.j        ,  7.09639953e-03+0.j        ,
  -2.41117844e-01+0.13588728j, -2.41117844e-01-0.13588728j,
   4.61361390e-01+0.j        ],
 [-4.70826375e-01+0.j        , -3.09091552e-01+0.j        ,
  -6.54107338e-01+0.j        , -6.54107338e-01-0.j        ,
  -7.63407713e-01+0.j        ],
 [-4.95840470e-01+0.j        , -6.25279503e-01+0.j        ,
   4.99997231e-01-0.09625804j,  4.99997231e-01+0.09625804j,
   3.91430197e-01+0.j        ]],

[[-4.07233043e-01+0.j        , -6.37950114e-01+0.j        ,
  -6.32455532e-01+0.j        , -9.48947507e-02+0.j        ,
  -2.08597837e-02+0.j        ],
 [-4.26795068e-01+0.j        , -3.21746641e-01+0.j        ,
   6.32455532e-01+0.j        , -1.34873945e-01+0.j        ,
   5.48599964e-01+0.j        ],
 [-4.46357093e-01+0.j        , -5.54316735e-03+0.j        ,
   3.16227766e-01+0.j        ,  7.19298140e-01+0.j        ,
  -8.01656852e-01+0.j        ],
 [-4.65919118e-01+0.j        ,  3.10660306e-01+0.j        ,
  -6.55034382e-14+0.j        , -6.54395442e-01+0.j        ,
   4.09529459e-02+0.j        ],
 [-4.85481142e-01+0.j        ,  6.26863780e-01+0.j        ,
  -3.16227766e-01+0.j        ,  1.64865998e-01+0.j        ,
   2.32963725e-01+0.j        ]]]))
```

EDA:

Frame a problem statement, clean, preprocess and visulaize the data and interpret your conclusion

Problem Statement:Breast cancer prediction dataset Visualization

Write a python program to visualize the breast cancer prediction dataset with the help of pandas and matplotlib library and understand the relationship between the parameters to define the tumor is malignant or benign

```
import pandas as pd
from matplotlib import pyplot as plt


df=pd.read_csv("/content/8_BreastCancerPrediction.csv")
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 33 columns

Double-click (or enter) to edit

```
df.columns
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
```

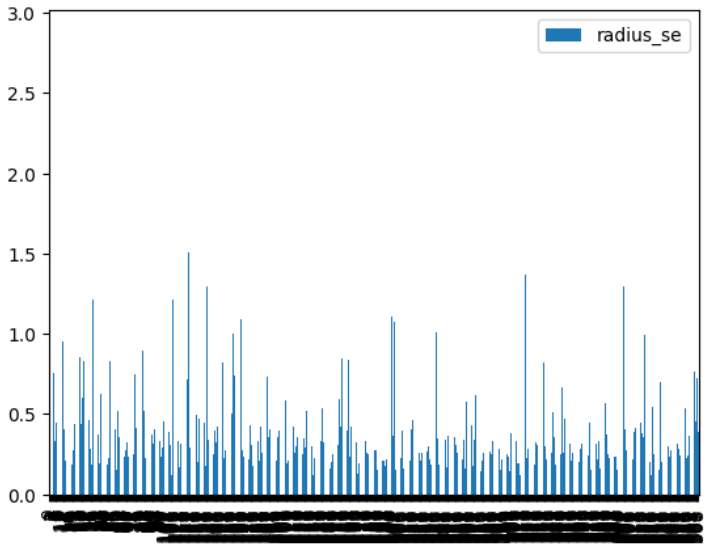
```
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

```
df.isna().sum()
```

```
id                           0
diagnosis                    0
radius_mean                  0
texture_mean                 0
perimeter_mean               0
area_mean                    0
smoothness_mean              0
compactness_mean             0
concavity_mean               0
concave points_mean          0
symmetry_mean                0
fractal_dimension_mean       0
radius_se                    0
texture_se                   0
perimeter_se                 0
area_se                      0
smoothness_se                0
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

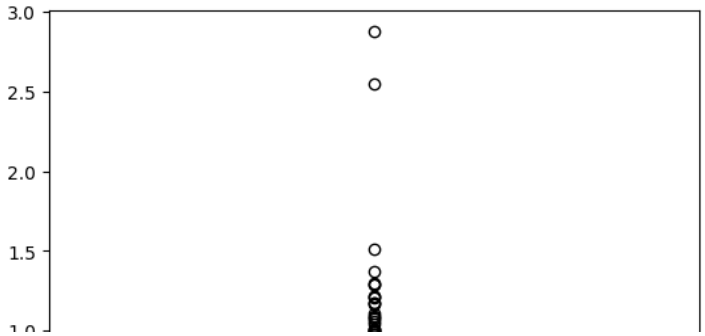
```
dat=df[["diagnosis","radius_se"]]
dat.plot.bar()
```

```
<Axes: >
```



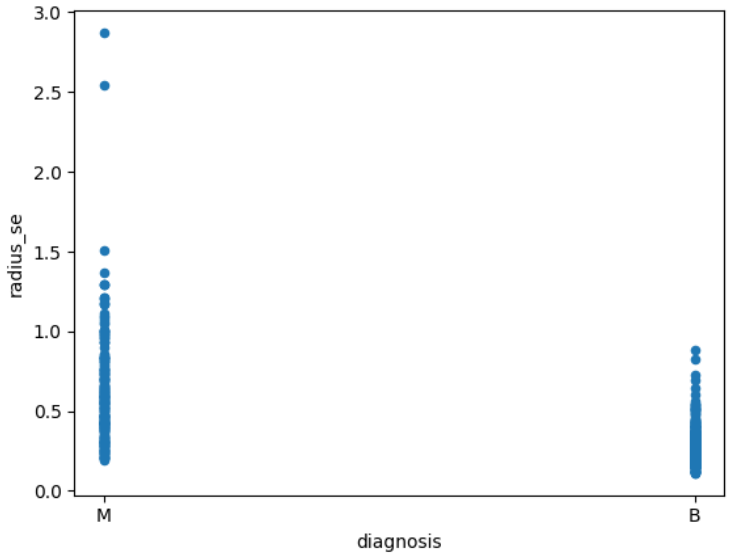

```
dat.plot.box()
```

```
<Axes: >
```




```
dat.plot.scatter("diagnosis","radius_se")
```

```
<Axes: xlabel='diagnosis', ylabel='radius_se'>
```



Conclusion:

The malignant tumor has the highest radius_se values than benign

Bottle Dataset

```
df2=pd.read_csv("/content/9_bottle.csv")
df2
```

```
<ipython-input-30-7f2c6ebef612>:1: DtypeWarning: Columns (47,73) have mixed types.
  df2=pd.read_csv("/content/9_bottle.csv")
```

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STh
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64
...	...	...	...	...	...	...	...	...	
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87

```
df2.isna().sum()
```

```
Cst_Cnt                    0
Btl_Cnt                    0
Sta_ID                     0
Depth_ID                   0
Depthm                     0
                         ...
TA1                   862779
TA2                   864629
pH2                   864853
pH1                   864779
DIC Quality Comment   864808
Length: 74, dtype: int64
```

```
df2.columns
```

```
Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'T_degC',
       'Salnty', 'O2ml_L', 'STheta', 'O2Sat', 'Oxy_µmol/Kg', 'BtlNum',
       'RecInd', 'T_prec', 'T_qual', 'S_prec', 'S_qual', 'P_qual', 'O_qual',
       'SThtaq', 'O2Satq', 'ChlorA', 'Chlqua', 'Phaeop', 'Phaqua', 'PO4uM',
       'PO4q', 'SiO3uM', 'SiO3qu', 'NO2uM', 'NO2q', 'NO3uM', 'NO3q', 'NH3uM',
       'NH3q', 'C14As1', 'C14A1p', 'C14A1q', 'C14As2', 'C14A2p', 'C14A2q',
       'DarkAs', 'DarkAp', 'DarkAq', 'MeanAs', 'MeanAp', 'MeanAq', 'IncTim',
       'LightP', 'R_Depth', 'R_TEMP', 'R_POTEMP', 'R_SALINITY', 'R_SIGMA',
       'R_SVA', 'R_DYNHT', 'R_O2', 'R_O2Sat', 'R_SIO3', 'R_PO4', 'R_NO3',
       'R_NO2', 'R_NH4', 'R_CHLA', 'R_PHAEO', 'R_PRES', 'R_SAMP', 'DIC1',
       'DIC2', 'TA1', 'TA2', 'pH2', 'pH1', 'DIC Quality Comment'],
      dtype='object')
```

```
df2=df2.drop(df2.iloc[:,60:].columns,axis=1)
```

```
per=(df2.isna().sum()/len(df2)*100)
pr=pd.DataFrame(per,columns=['values'])
per1=pr[pr['values']>=80].index
per1
```

```
Index(['BtlNum', 'T_qual', 'S_qual', 'SThtaq', 'NH3uM', 'C14As1', 'C14A1p',
       'C14As2', 'C14A2p', 'DarkAs', 'DarkAp', 'MeanAs', 'MeanAp', 'IncTim',
       'LightP'],
      dtype='object')
```

```
df2=df2.drop(['BtlNum', 'T_qual', 'S_qual', 'SThtaq', 'NH3uM', 'C14As1', 'C14A1p',
       'C14As2', 'C14A2p', 'DarkAs', 'DarkAp', 'MeanAs', 'MeanAp', 'IncTim',
       'LightP'],axis=1)
df2.isna().sum()/len(df2)*100
```

```
Cst_Cnt         0.000000
Btl_Cnt         0.000000
Sta_ID          0.000000
Depth_ID        0.000000
Depthm          0.000000
T_degC          1.267600
Salnty          5.475318
O2ml_L         19.501586
STheta          6.092179
O2Sat          23.540029
Oxy_µmol/Kg    23.540723
RecInd          0.000000
T_prec          1.267600
S_prec          5.475318
P_qual         22.096910
O_qual         78.646791
O2Satq         74.817168
ChlorA         73.952869
Chlqua         26.096272
Phaeop         73.952984
Phaqua         26.095809
PO4uM          52.210119
PO4q           47.762131
SiO3uM         59.058140
SiO3qu         40.930991
NO2uM          60.967691
NO2q           38.779437
NO3uM          60.987694
NO3q           38.726365
NH3q            6.540227
C14A1q          1.879835
C14A2q          1.877754
DarkAq          2.823915
MeanAq          2.824031
R_Depth         0.000000
R_TEMP          1.267600
R_POTEMP        5.324196
R_SALINITY      5.475318
R_SIGMA         6.111488
R_SVA           6.101660
R_DYNHT         5.394727
R_O2           19.501586
R_O2Sat        22.941784
R_SIO3         59.057215
R_PO4          52.209194
dtype: float64
```

```
df3=df2.iloc[:,:11]
df3
```

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STh
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-	0	10.500	33.4400	NaN	25.64

```
print(df3.columns)
df3.isna().sum()/len(df3)*100
```

```
Index(['Cst_Cnt', 'Btl_Cnt', 'Sta_ID', 'Depth_ID', 'Depthm', 'T_degC',
       'Salnty', 'O2ml_L', 'STheta', 'O2Sat', 'Oxy_µmol/Kg'],
      dtype='object')
Cst_Cnt         0.000000
Btl_Cnt         0.000000
Sta_ID          0.000000
Depth_ID        0.000000
Depthm          0.000000
T_degC          1.267600
Salnty          5.475318
O2ml_L         19.501586
STheta          6.092179
O2Sat          23.540029
Oxy_µmol/Kg    23.540723
dtype: float64
```

```
df3=df3.fillna(df3['T_degC'].mean())
df3=df3.fillna(df3['Salnty'].mean())
df3=df3.fillna(df3['O2ml_L'].mean())
df3=df3.fillna(df3['STheta'].mean())
df3=df3.fillna(df3['Oxy_µmol/Kg'].mean())
df3=df3.fillna(df3['O2Sat'].median())

df3.isna().sum()
```

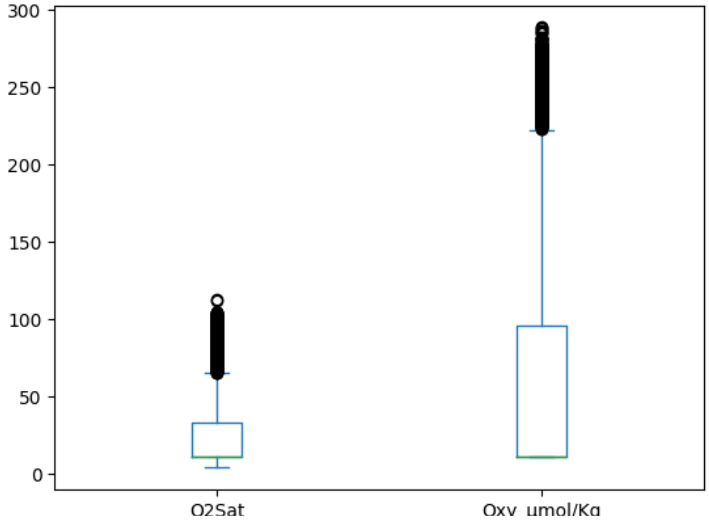
```
Cst_Cnt        0
Btl_Cnt        0
Sta_ID         0
Depth_ID       0
Depthm         0
T_degC         0
Salnty         0
O2ml_L         0
STheta         0
O2Sat          0
Oxy_µmol/Kg    0
dtype: int64
```

```
df3
```

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	S
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	10.799677	25.
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	10.799677	25.
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	10.799677	25.

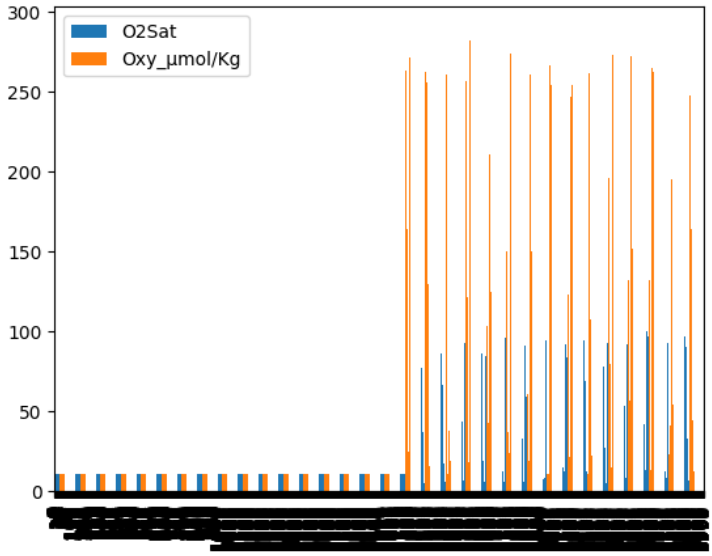
```
ds=df3[["O2Sat","Oxy_µmol/Kg"]]
ds=ds.iloc[:4000]

ds.plot.box()
```

```
<Axes: >
```



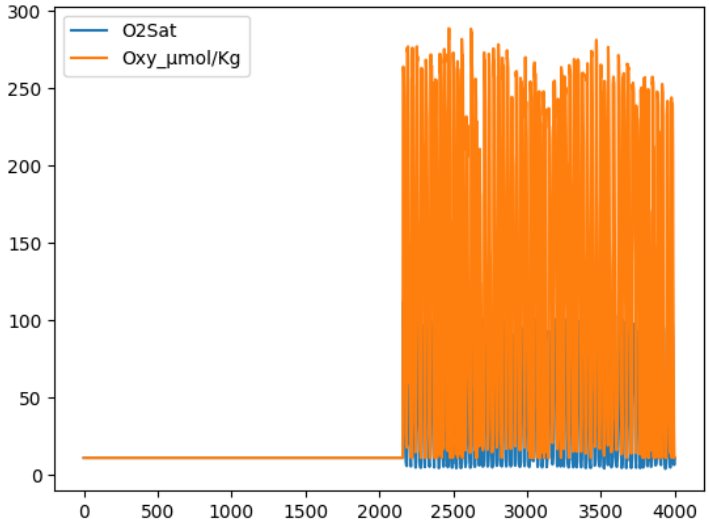

```
ds.plot.bar()
```

```
<Axes: >
```




```
ds.plot.line()
```

```
<Axes: >
```



✓ 1s completed at 7:49 AM