Machine Learning:

Machine Learning Procedure:-

- Dataset Reading and Studying
- Data Cleaning and Analysis
- Data Visualization or EDA (Exploratory Data Analysis)
- Encoding (converting of string columns to integer columns)
- ip/op Creation (separating input data and output/target data)
- Train Test Split (separate the training data and testing data)
- Standard Scaler Transform (standardizing all the input datas)
- Machine Learning Algorithm
- Prediction
- Accuracy

Regression Model:-

Linear Regression:-

In [1]:
```
# Importing the packages:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Reading the Dataset:-
adv = pd.read_csv(r"C:\Users\lab25\Downloads\archive (9)\advertising.csv")
adv
```

Out[2]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [3]:
```
# Data Cleaning:-
# Checking null values:
adv.isnull().sum()
```

Out[3]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [4]:
```
# checking the datatypes:
adv.dtypes
```

Out[4]:
```
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object
```

In [5]:
```
# checking the unique values:
for i in adv.columns:
    print(f"{i}:\n {adv[i].unique()}\n")
```

```
TV:
 [230.1  44.5  17.2 151.5 180.8   8.7  57.5 120.2   8.6 199.8  66.1 214.7
  23.8  97.5 204.1 195.4  67.8 281.4  69.2 147.3 218.4 237.4  13.2 228.3
  62.3 262.9 142.9 240.1 248.8  70.6 292.9 112.9  97.2 265.6  95.7 290.7
 266.9  74.7  43.1 228.  202.5 177.  293.6 206.9  25.1 175.1  89.7 239.9
 227.2  66.9 100.4 216.4 182.6 262.7 198.9   7.3 136.2 210.8 210.7  53.5
 261.3 239.3 102.7 131.1  69.   31.5 139.3 216.8 199.1 109.8  26.8 129.4
 213.4  16.9  27.5 120.5   5.4 116.   76.4 239.8  75.3  68.4 213.5 193.2
  76.3 110.7  88.3 134.3  28.6 217.7 250.9 107.4 163.3 197.6 184.9 289.7
 135.2 222.4 296.4 280.2 187.9 238.2 137.9  25.   90.4  13.1 255.4 225.8
 241.7 175.7 209.6  78.2  75.1 139.2 125.7  19.4 141.3  18.8 224.  123.1
 229.5  87.2   7.8  80.2 220.3  59.6   0.7 265.2   8.4 219.8  36.9  48.3
  25.6 273.7  43.   73.4 193.7 220.5 104.6  96.2 140.3 243.2  38.   44.7
 280.7 121.  171.3 187.8   4.1  93.9 149.8  11.7 131.7 172.5  85.7 188.4
 163.5 117.2 234.5  17.9 206.8 215.4 284.3  50.  164.5  19.6 168.4 276.9
 248.4 170.2 276.7 165.6 156.6 218.5  56.2 287.6 253.8 205.  139.5 191.1
 286.   18.7  39.5  75.5 166.8 149.7  38.2  94.2 283.6 232.1]

Radio:
 [37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6  2.1  2.6  5.8 24.  35.1  7.6
 32.9 47.7 36.6 39.6 20.5 23.9 27.7  5.1 15.9 16.9 12.6  3.5 29.3 16.7
 27.1 16.  28.3 17.4  1.5 20.   1.4  4.1 43.8 49.4 26.7 37.7 22.3 33.4
  8.4 25.7 22.5  9.9 41.5 15.8 11.7  3.1  9.6 41.7 46.2 28.8 28.1 19.2
 49.6 29.5  2.  42.7 15.5 29.6 42.8  9.3 24.6 14.5 27.5 43.9 30.6 14.3
 33.   5.7 43.7  1.6 28.5 29.9  7.7 20.3 44.5 43.  18.4 40.6 25.5 47.8
  4.9 33.5 36.5 14.  31.6 21.  42.3  4.3 36.3 10.1 17.2 34.3 46.4 11.
  0.3  0.4 26.9  8.2 38.  15.4 20.6 46.8 35.   0.8 36.9 26.8 21.7  2.4
 34.6 32.3 11.8 38.9  0.  49.  12.   2.9 27.2 38.6 47.  39.  28.9 25.9
 17.  35.4 33.2 14.8  1.9  7.3 40.3 25.8 13.9 23.3 39.7 21.1 11.6 43.5
  1.3 18.1 35.8 36.8 14.7  3.4 37.6  5.2 23.6 10.6 20.9 20.1  7.1 30.2
  7.8  2.3 10.   5.4 21.3 45.1 28.7 12.1 41.1 42.  35.6  3.7  8.6]

Newspaper:
 [ 69.2  45.1  69.3  58.5  58.4  75.   23.5  11.6   1.   21.2  24.2   4.
  65.9   7.2  46.   52.9 114.   55.8  18.3  19.1  53.4  49.6  26.2  19.5
  12.6  22.9  40.8  43.2  38.6  30.    0.3   7.4   8.5   5.   45.7  35.1
  32.   31.6  38.7   1.8  26.4  43.3  31.5  35.7  18.5  49.9  36.8  34.6
   3.6  39.6  58.7  15.9  60.   41.4  16.6  37.7   9.3  21.4  54.7  27.3
   8.4  28.9   0.9   2.2  10.2  11.   27.2  31.7  19.3  31.3  13.1  89.4
  20.7  14.2   9.4  23.1  22.3  36.9  32.5  35.6  33.8  65.7  16.   63.2
  73.4  51.4  33.   59.   72.3  10.9   5.9  22.   51.2  45.9  49.8 100.9
  17.9   5.3  29.7  23.2  25.6   5.5  56.5   2.4  10.7  34.5  52.7  14.8
  79.2  46.2  50.4  15.6  12.4  74.2  25.9  50.6   9.2   3.2  43.1   8.7
  43.    2.1  65.6  59.7  20.5   1.7  12.9  75.6  37.9  34.4  38.9   9.
  44.3  11.9  20.6  37.   48.7   9.5   5.7  50.5  24.3  45.2  30.7  49.3
   5.4  84.8  21.6  19.4  57.6   6.4  18.4  47.4  17.   12.8  41.8  20.3
  35.2  23.7  17.6   8.3  27.4  71.8  19.6  26.6  18.2   3.7  23.4   5.8
   6.   13.8   8.1  66.2]

Sales:
 [22.1 10.4 12.  16.5 17.9  7.2 11.8 13.2  4.8 15.6 12.6 17.4  9.2 13.7
 19.  22.4 12.5 24.4 11.3 14.6 18.  17.5  5.6 20.5  9.7 17.  15.  20.9
 18.9 10.5 21.4 11.9 17.8 25.4 14.7 10.1 21.5 16.6 17.1 20.7  8.5 16.1
 10.6 23.2 19.8 16.4 10.7 22.6 21.2 20.2 23.7  5.5 23.8 18.4  8.1 24.2
 14.  16.  11.  13.4 22.3 18.3 12.4  8.8  8.7  6.9 14.2  5.3 17.3 13.6
 21.7 12.9 16.7  7.3 19.4 22.2 11.5 16.9 17.2 19.7 21.8 12.2  9.4 15.9
```

```
  6.6 15.5  7.  15.2 24.7  1.6 17.7  5.7 19.6 10.8 11.6  9.5 20.8  9.6
 10.9 19.2 20.1 12.3 10.3 18.2 20.6  3.2 15.3 13.3 19.9  8.  20.   8.4
  7.6 27.  16.8 17.6 26.2  6.7  5.9 14.8 25.5]
```

In [6]:
```
# to check the information of the dataset:
adv.info()
```

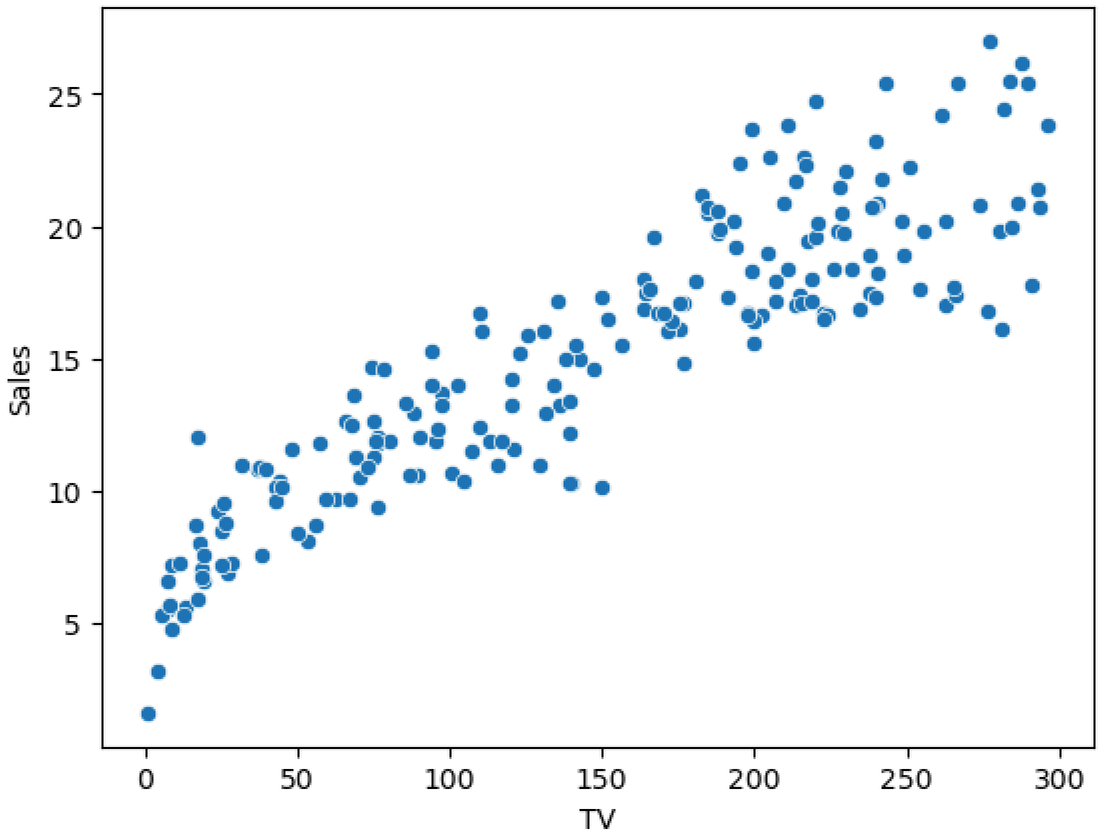
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [7]:
```
# to check the statistical value of all columns:
adv.describe()
```

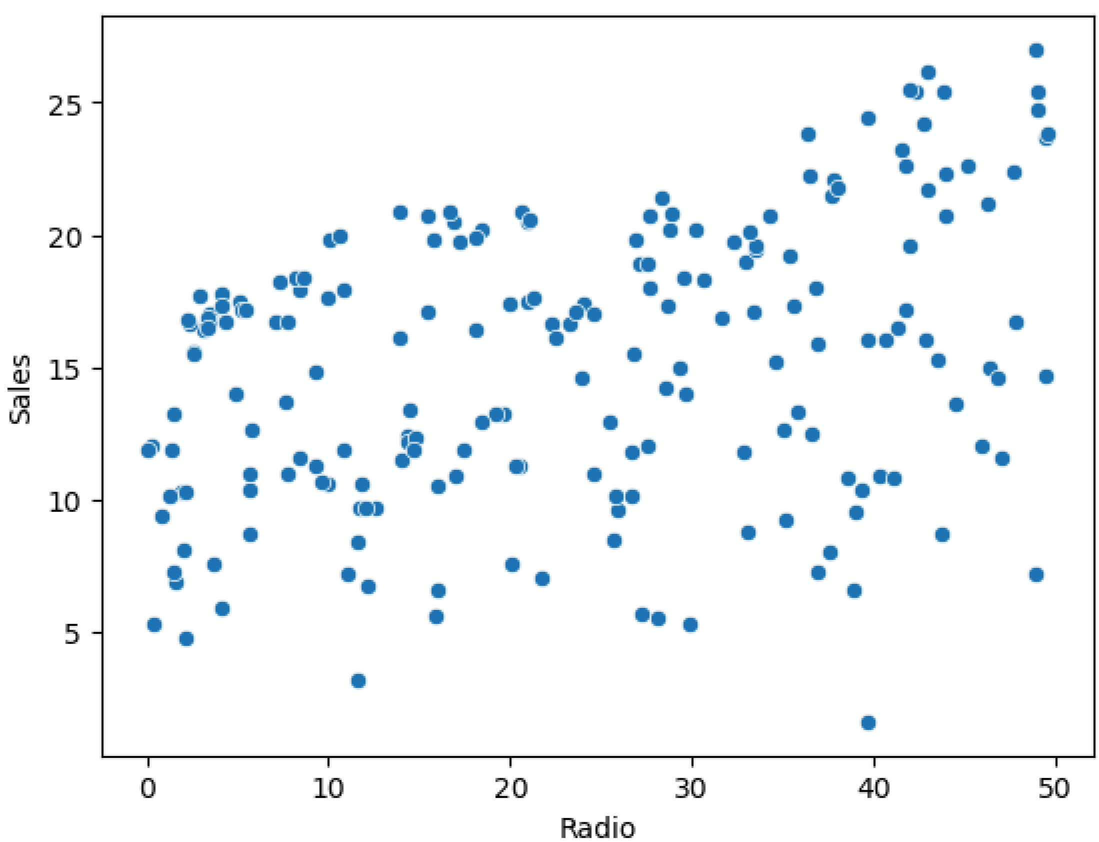
Out[7]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

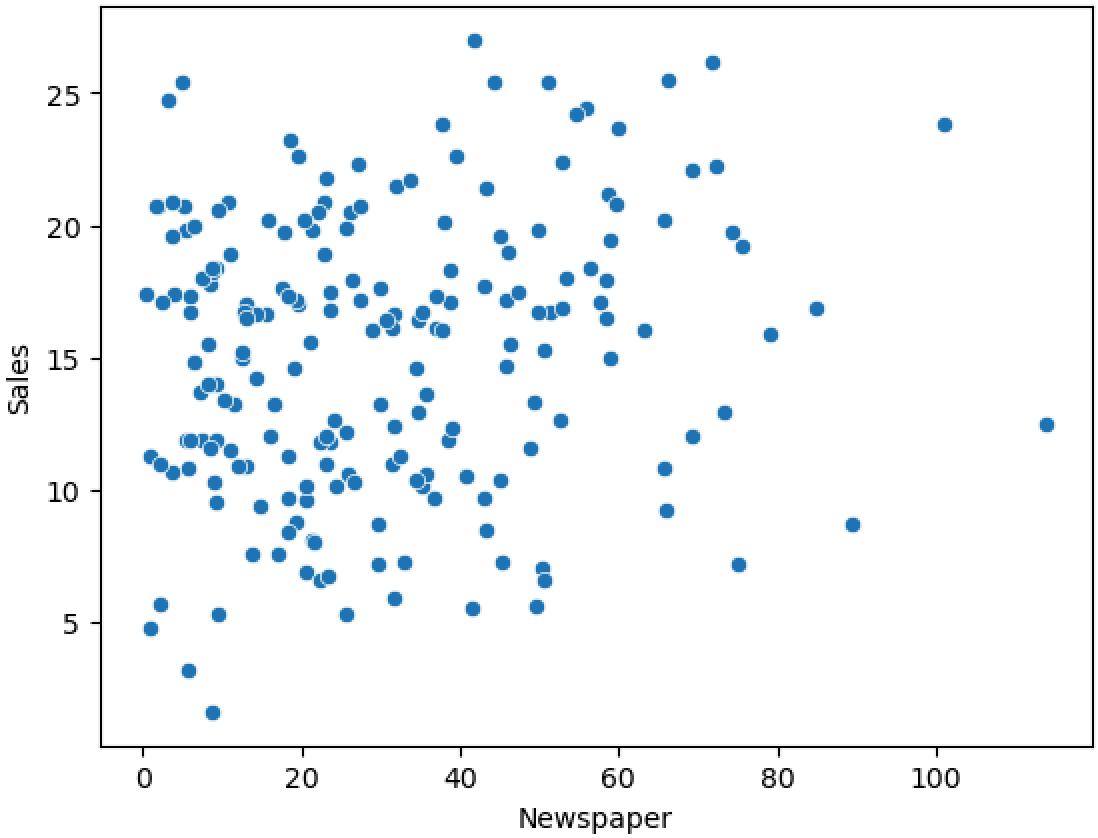
In [8]:
```
# Data Visualization:
sns.scatterplot(x=adv.TV, y=adv.Sales)
plt.show()
```



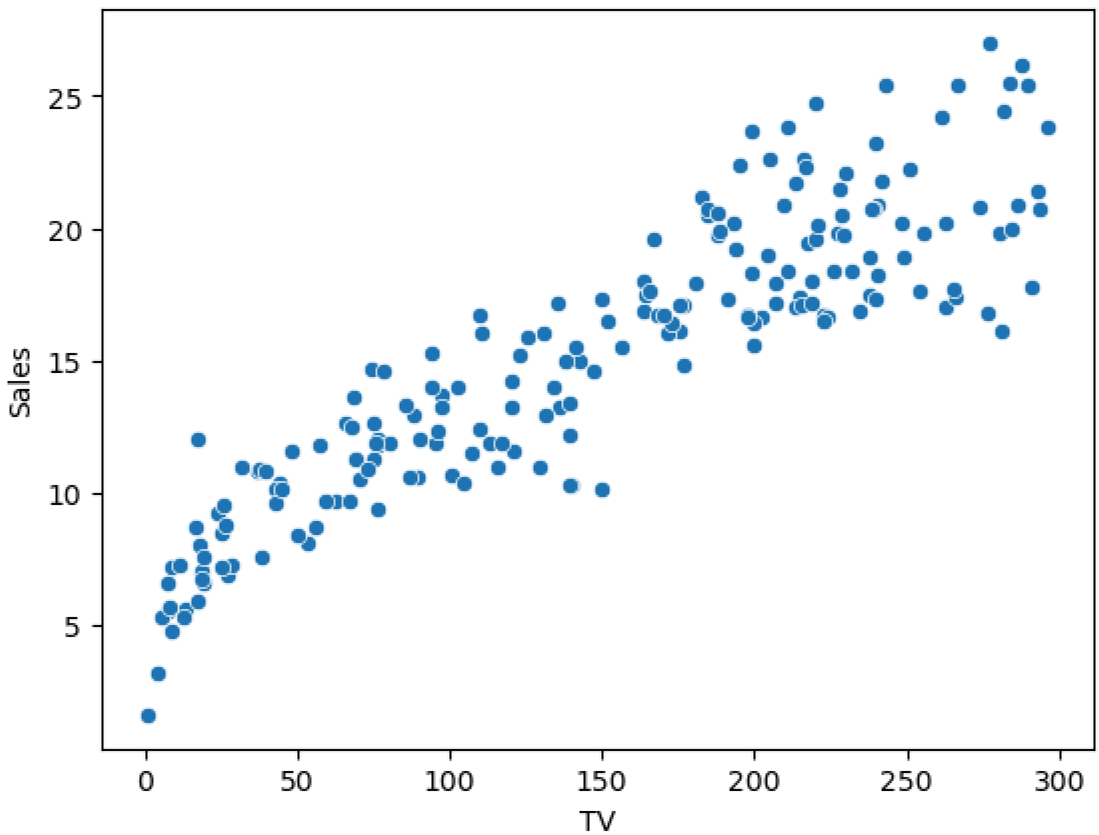
In [9]:
```
sns.scatterplot(x=adv.Radio, y=adv.Sales)
plt.show()
```

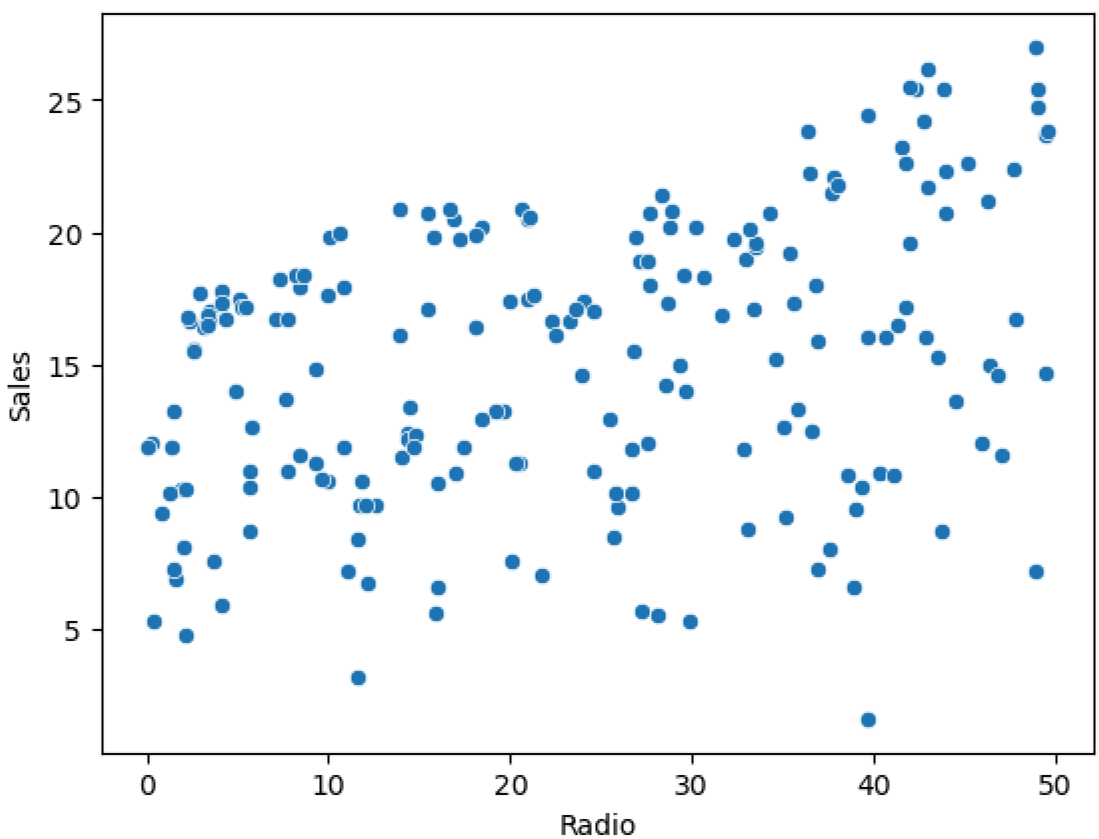


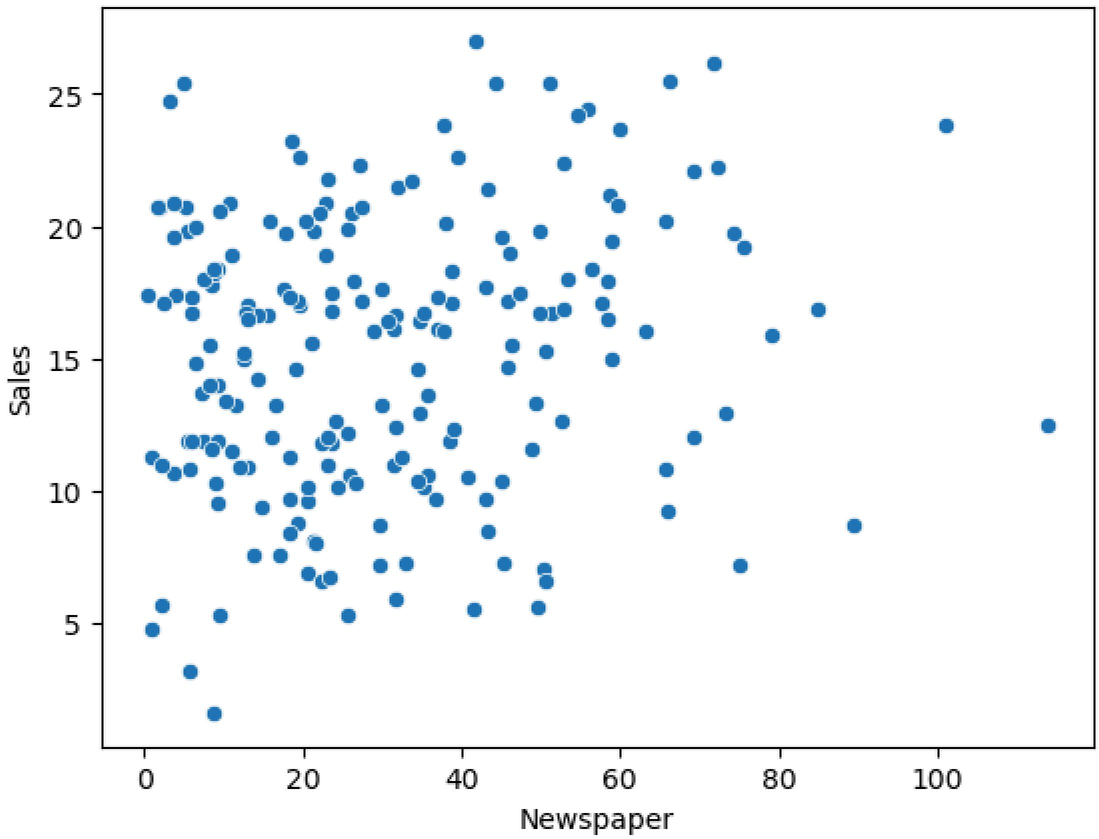
In [10]:
```
sns.scatterplot(x=adv.Newspaper, y=adv.Sales)
plt.show()
```



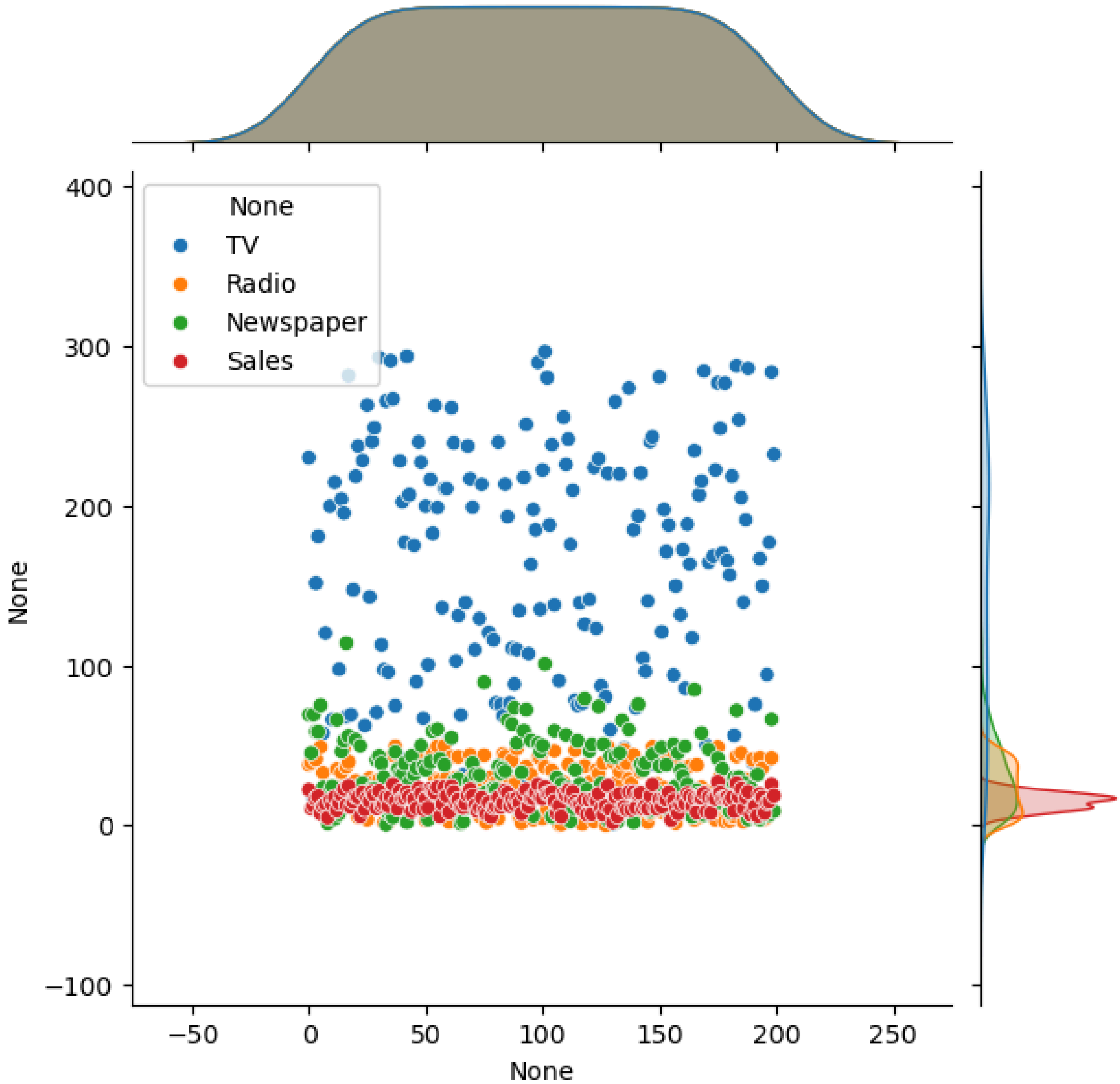
In [11]:
```
# Plotting the scatterplot for all the continuous columns at a time.
for i in ['TV','Radio','Newspaper']:
    sns.scatterplot(x=adv[i], y=adv.Sales)
    plt.show()
```


Sales
TV
0 50 100 150 200 250 300
5 10 15 20 25

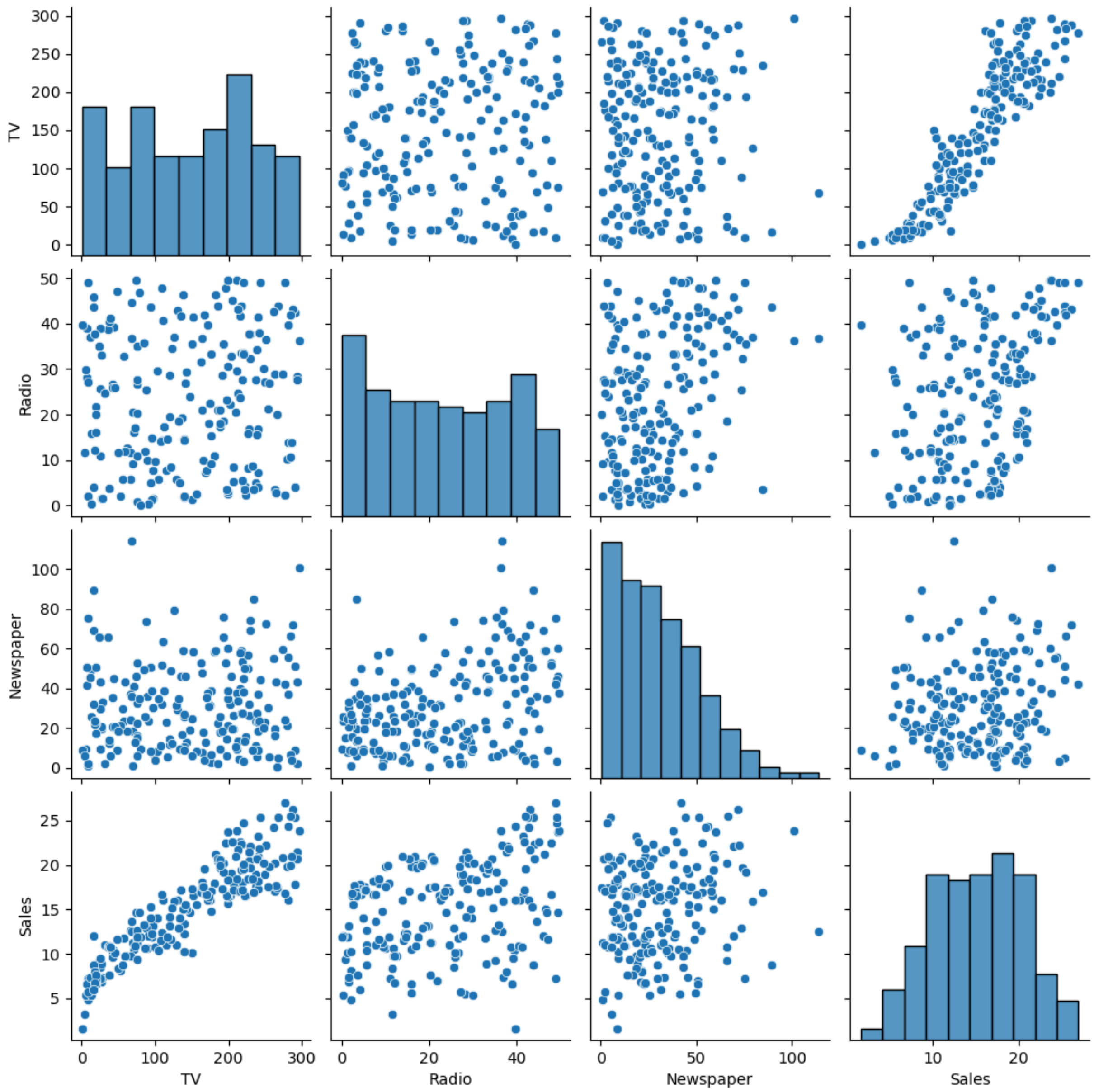

Sales
Radio
0 10 20 30 40 50
5 10 15 20 25



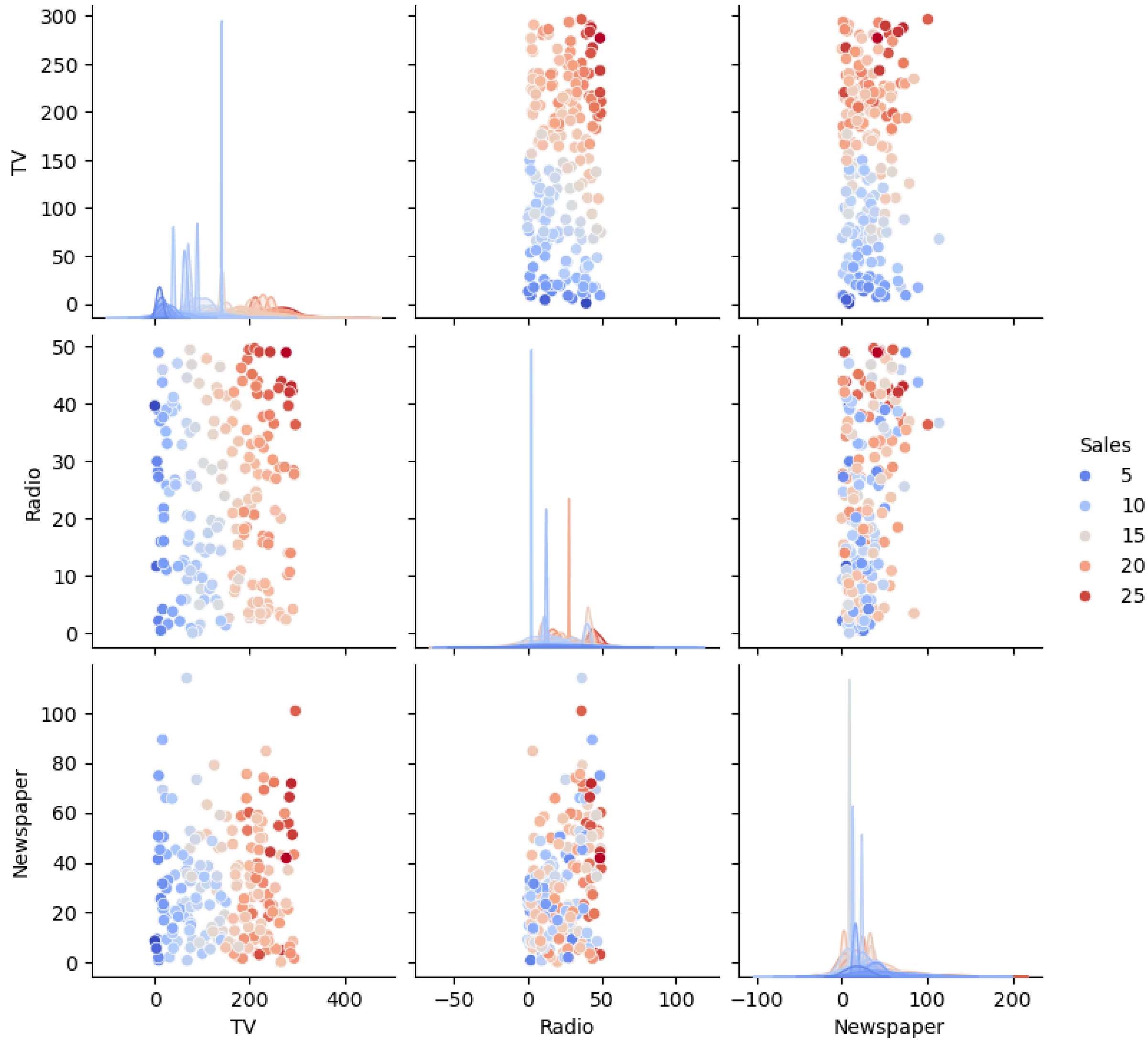
In [12]:
```
sns.jointplot(data=adv)
plt.show()
```



In [13]:
```
sns.pairplot(adv)
plt.show()
```



In [14]:
```
sns.pairplot(adv,hue='Sales',palette='coolwarm')
plt.show()
```

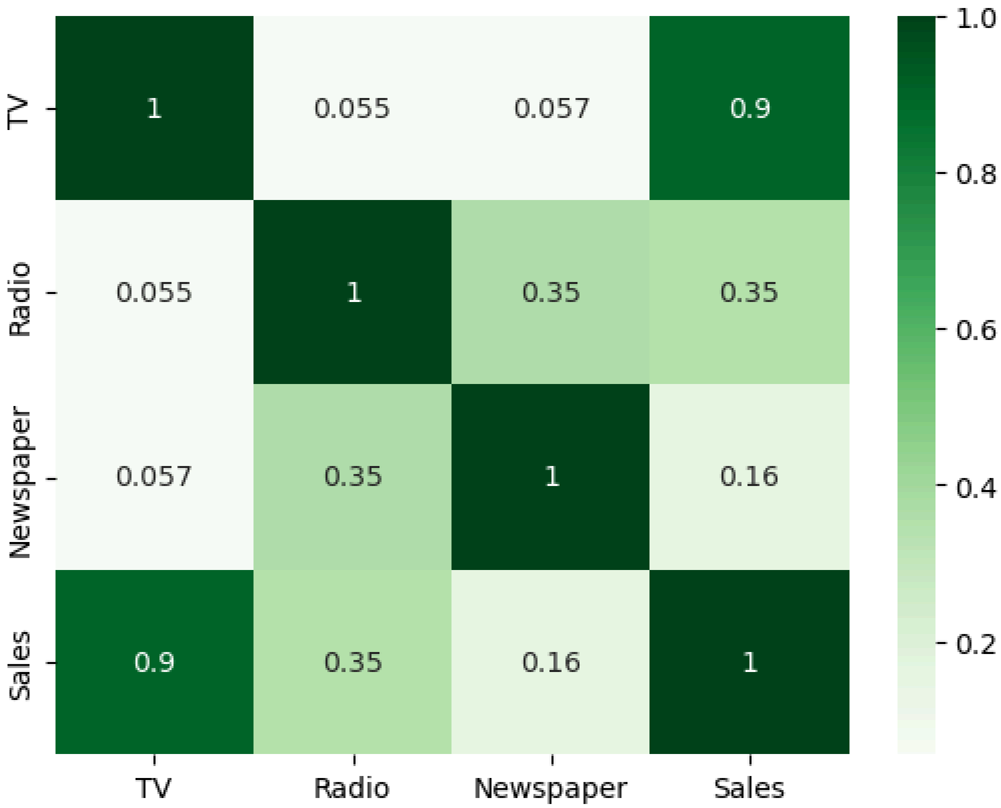


In [15]:
```
# Correlation: finds the statistical relationship between 2 varibales.
c = adv.corr()
c
```

Out[15]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.901208
Radio	0.054809	1.000000	0.354104	0.349631
Newspaper	0.056648	0.354104	1.000000	0.157960
Sales	0.901208	0.349631	0.157960	1.000000

In [16]:
```
# plotting the correlation using heatmap:
sns.heatmap(c,annot=True,cmap='Greens')
plt.show()
```



In [17]:

```
# Encoding:- As the dataset doesnot contain any string columns therefore this step
```

In [18]:

```
# Creation of ip/op:- separating the input columns from output columns
adv
```

Out[18]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [19]:
```
# ip will store all the input columns except the target one.
ip = adv.drop('Sales',axis=1)
ip.head()
```

Out[19]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

In [20]:
```
op = adv.Sales
op.head()
```

Out[20]:
```
0    22.1
1    10.4
2    12.0
3    16.5
4    17.9
Name: Sales, dtype: float64
```

In [21]:
```
# Train Test Split: splitting of the 100% datas into training and testing datas.
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(ip,op,test_size=0.15,random_state = 5)
```

In [22]:
```
xtrain.head()
```

Out[22]:

	TV	Radio	Newspaper
25	262.9	3.5	19.5
156	93.9	43.5	50.5
42	293.6	27.7	1.8
141	193.7	35.4	75.6
50	199.8	3.1	34.6

In [23]:
```
xtest.head()
```

Out[23]:

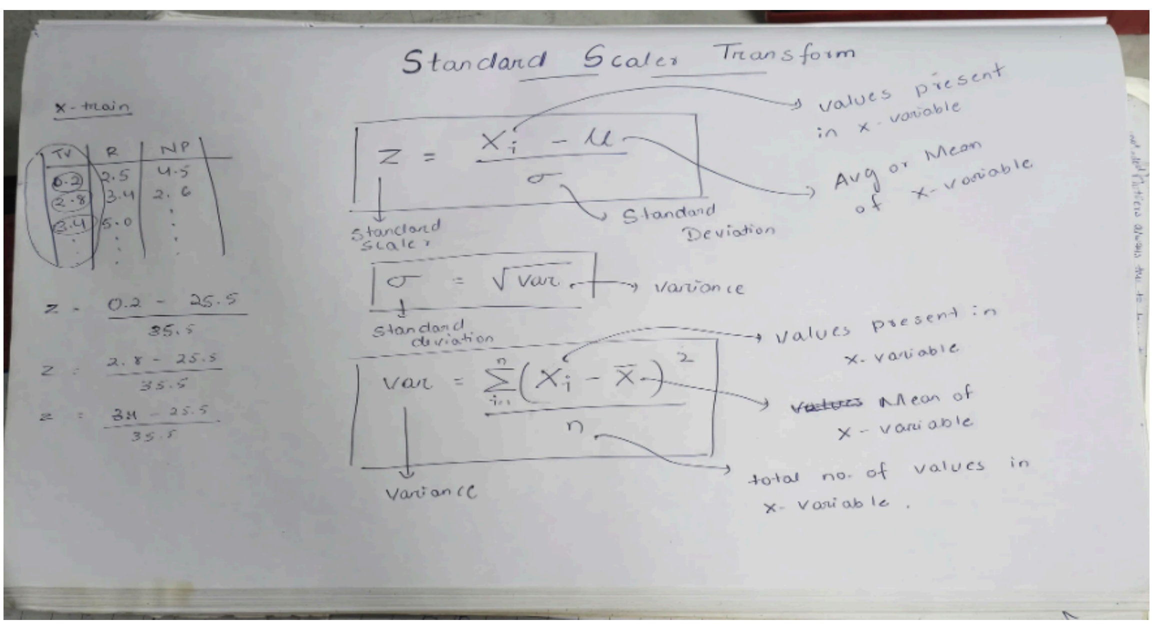
	TV	Radio	Newspaper
119	19.4	16.0	22.3
77	120.5	28.5	14.2
148	38.0	40.3	11.9
149	44.7	25.8	20.6
154	187.8	21.1	9.5

In [24]:
```
ytrain.head()
```

Out[24]:
```
25      17.0
156     15.3
42      20.7
141     19.2
50      16.4
Name: Sales, dtype: float64
```

In [25]:
```
ytest.head()
```

Out[25]:
```
119      6.6
77      14.2
148     10.9
149     10.1
154     20.6
Name: Sales, dtype: float64
```



In [26]:
```
# Stadardizing the datas using Standard Scaler Transform:
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [27]:
```
xtrain = sc.fit_transform(xtrain)
xtest = sc.fit_transform(xtest)
```

In [28]:
```
xtrain
```

```
Out[28]: array([[ 1.38344542, -1.32993638, -0.50913682],
                [-0.58787665,  1.38145277,  0.92126676],
                [ 1.74154949,  0.31045406, -1.32585112],
                [ 0.57625319,  0.83239647,  2.07943224],
                [ 0.64740742, -1.35705027,  0.18760815],
                [-0.51205657, -0.91644953, -1.24279543],
                [-1.23759641, -1.31637943, -0.77214651],
                [-0.05946902, -0.59786131, -0.22767031],
                [ 0.88886521,  0.68327007,  0.33987692],
                [-0.95647888, -0.71309535, -0.56450728],
                [-0.14695372, -0.31994392,  0.18760815],
                [ 1.10816021, -0.51651963, -0.14922882],
                [-1.47438835,  0.98152287, -0.41223851],
                [ 0.35929112, -0.04202653,  0.04456779],
                [ 0.9798493 , -0.42162101, -0.19998508],
                [-1.16410925,  1.09675691,  0.67209968],
                [ 0.51443066, -0.34027934, -0.22767031],
                [ 0.96701821, -0.49618421,  0.89358153],
                [ 0.32896308, -0.34027934,  0.00765415],
                [-1.59220108,  1.06964302,  0.92588096],
                [-0.24726952,  0.77816869, -0.83674538],
                [-0.62870284, -1.54684751, -0.33841123],
                [ 0.73022627, -0.99779121, -0.19075667],
                [ 0.62174524,  0.01220125, -0.75368969],
                [-0.7710113 ,  1.60514238,  0.18299395],
                [-0.63686808, -0.89611411,  0.23836441],
                [-1.45455848, -0.20470988, -0.62449194],
                [ 1.58524348, -0.88255717, -0.42146692],
                [ 1.36478202,  1.32722499,  1.11506337],
                [-0.80717165,  0.80528258,  1.02277927],
                [-1.0999538 , -0.78088007, -0.55989307],
                [-1.02763311, -1.18080997, -0.0384879 ],
                [ 0.77455186,  0.43246657, -0.97978574],
                [ 0.70806348,  1.48990834, -0.50452261],
                [-1.40556704,  0.81206105,  1.63185434],
                [ 0.80604635,  0.1003214 , -0.80444594],
                [-0.0349733 ,  0.2494478 ,  0.72285594],
                [ 1.21430832,  0.47991588, -0.47222318],
                [ 0.03501447,  0.05287209, -0.52759364],
                [-0.0944629 , -0.26571614, -0.64294877],
                [-1.31574941,  0.1003214 , -1.3073943 ],
                [-1.18160619,  0.18844155, -0.46299477],
                [-0.48522792,  0.43924504, -1.02131358],
                [-1.38457071,  1.07642149, -0.97978574],
                [ 1.77421044,  0.89340272,  3.24682612],
                [-0.40240907,  1.6729271 ,  0.96279461],
                [ 0.59608306,  1.66614863,  1.03200768],
                [ 0.30213444, -1.03846204,  0.21529338],
                [-0.91215329, -1.1740315 , -0.29226918],
                [ 0.62174524, -1.32993638, -1.13666871],
                [ 1.02417489, -0.98423426, -1.00747097],
                [ 0.28113811, -1.08591135, -0.81828856],
                [ 1.43010393,  1.40178819, -1.17819656],
                [ 0.22164851,  0.5748145 ,  1.03200768],
                [ 0.42577949, -0.83510786,  1.28578896],
                [ 1.69605744,  1.3001111 ,  0.95356619],
```

```
[-0.3919109 ,  1.18487706,  1.50727081],
[-0.88532465,  1.4492375 ,  0.2337502 ],
[ 0.82937561,  0.03253667,  1.24887532],
[-0.8025058 , -0.83510786, -1.1320545 ],
[ 1.11399252, -1.28926554,  0.29373487],
[ 1.13615531,  1.00863676, -0.33841123],
[-1.48255359, -1.28926554,  0.049182  ],
[ 1.54441728, -1.41127805, -0.3153402 ],
[-0.54588399, -1.05201899, -1.07668404],
[ 1.59107579, -0.6249752 ,  0.29834907],
[-1.67501994,  1.11709233, -1.00747097],
[-1.37057316,  0.66971312, -0.51836523],
[-0.90282159, -0.7741016 ,  0.28912066],
[ 0.86436949,  0.31045406,  1.05507871],
[ 1.70772207, -1.28926554, -1.01669938],
[ 0.63924218,  0.50702977,  0.37679056],
[-0.33008838, -1.04524052, -0.34302543],
[ 0.85620425,  0.70360548,  1.31347419],
[ 0.47360447, -0.14370362, -0.39378169],
[-0.65319856,  0.16132765,  1.97791972],
[-0.85966247, -0.48262727,  0.47368887],
[-0.04663793, -1.43839194, -0.99362835],
[-1.0591276 , -1.43161347, -0.42146692],
[-0.74768205, -1.56718293, -0.98439994],
[ 0.84570609,  1.40856666, -0.15384303],
[ 0.84104024,  1.25944026,  0.4183184 ],
[-1.5292121 , -0.48940574,  0.87973891],
[ 0.06417604, -1.47906278, -0.28765497],
[ 1.63306845, -0.8486648 , -1.11359768],
[-1.48605297,  1.39500972,  2.71619254],
[ 0.88653228,  1.75426878, -1.26125225],
[-0.8118375 ,  1.78138267,  0.69978492],
[ 1.67156172,  1.34756041,  1.90409244],
[-0.11662569, -1.23503776, -0.97978574],
[ 0.06300957,  0.84595341, -1.1320545 ],
[-0.1539525 ,  1.33400346, -0.07540154],
[-0.82700151, -0.41484254, -0.81367435],
[ 0.24847715, -0.88933564, -0.59680671],
[ 1.62490321,  1.27977568,  1.64569696],
[ 0.23564606, -0.1504821 ,  0.7782264 ],
[ 0.72905981, -1.21470234, -0.51375102],
[-0.80483872, -0.19115293,  0.09070984],
[ 0.47360447,  1.40856666, -1.33046532],
[-1.36240792, -1.45872736, -0.45376636],
[-0.54938338, -1.46550583, -0.02464528],
[-0.05596963, -1.424835  , -0.18152826],
[ 0.38145391, -0.93678495, -1.11359768],
[ 0.88069997,  0.70360548,  0.67209968],
[ 1.11515898,  1.24588331, -0.55527887],
[ 0.26247471,  1.27977568, -1.24279543],
[ 0.64740742, -1.39094263, -0.43069533],
[-0.68352659,  0.85951036,  0.8658963 ],
[ 0.69756532,  0.66293465,  0.71362753],
[-0.43040417, -0.61819673, -0.90595846],
[-0.89232343,  0.91373814,  3.85128699],
[ 0.31496553,  1.1238708 ,  0.33064851],
```

```
[ 0.57042087, -0.31994392,  1.62262593],
[ 1.41493991, -0.21148835, -1.39506419],
[-1.62019619,  0.45958046, -0.97517153],
[-1.34957683, -1.46550583,  0.11378087],
[ 1.41027406, -1.37060721,  0.57520138],
[ 0.99384686,  0.62226381,  2.01483336],
[-0.05830255, -0.58430436, -0.93825789],
[ 0.82121037,  0.05965056, -1.22433861],
[ 1.29596071,  0.25622627, -1.15512553],
[-1.53037856, -1.54006904, -0.22767031],
[ 0.911028  , -1.27570859,  0.88896732],
[-1.11978367,  1.61869932, -1.01669938],
[ 0.54592516,  0.37823879, -0.56912148],
[-0.87599295, -0.17759599, -0.56450728],
[ 1.15365226,  1.75426878,  0.63518604],
[ 0.22398143,  0.92729509, -1.06745563],
[-1.18043973,  0.24266933,  0.21067918],
[ 1.08599742,  0.29689711, -0.90134425],
[-1.48255359,  1.54413612,  1.78873732],
[-0.36624872, -0.38772865,  0.37217635],
[ 1.27729731, -0.12336821, -0.02464528],
[-0.07463303,  1.57802848,  1.31347419],
[-0.31609082, -0.57074742, -1.15973974],
[ 0.95068774, -1.01134815,  1.19811906],
[-1.58520231,  0.27656169, -1.3120085 ],
[ 1.00084563,  0.99507982,  1.78412311],
[ 0.77571832,  1.79493962,  0.33064851],
[-1.58170292,  1.74749031,  2.05174701],
[-0.58437726, -1.23503776, -1.0351562 ],
[ 0.50859835, -0.40128559, -0.5829641 ],
[-0.98797337, -0.75376618,  0.57981558],
[ 0.38145391,  0.69682701,  0.37679056],
[-1.39156949, -0.82155091, -0.0384879 ],
[ 1.5094234 ,  0.39179573,  1.34577363],
[-1.46389018, -0.09625431,  0.91665255],
[ 1.11749191, -1.07235441, -1.00747097],
[-0.66602965, -0.76732313, -0.21382769],
[-0.79317409,  0.29689711, -0.670634  ],
[ 1.73338425,  0.35112489,  0.58442979],
[ 1.54675021,  1.74749031,  0.51983092],
[-0.87832587, -0.93678495, -1.36737896],
[-1.39040303,  0.1748846 ,  0.58904399],
[ 1.11749191, -0.43517796, -0.35225384],
[-0.79200763,  0.24266933, -0.37993908],
[ 0.86553595, -1.20114539, -0.14461462],
[ 0.76172077, -0.17081752, -0.91518687],
[-0.46306513, -1.18080997,  0.17837974],
[-0.28109694, -0.23860225, -0.87365902],
[-1.54670904,  0.93407356,  0.67671389],
[ 0.36628989, -0.52329811, -1.29816589],
[-1.63536021, -0.78088007, -1.14589712],
[-1.22243239,  1.21876942, -1.14128291],
[-1.58286938, -1.424835  , -1.36276476],
[-0.17378237, -1.18080997,  0.03533938],
[-0.561048  , -0.56396894,  0.38601897],
[-0.21694149,  0.93407356,  2.24554362],
```

```
        [-1.46505664, -0.74698771, -0.32918282],
        [-0.10612753,  1.25944026,  0.70901333]])
```

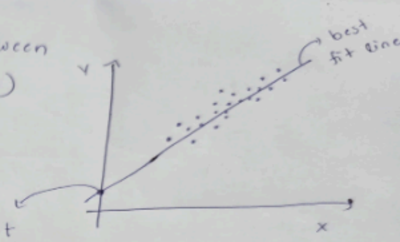
In [29]:
```
xtest
```

Out[29]:
```
array([[-1.71612085, -0.53499267, -0.38003424],
       [-0.50447354,  0.29265688, -0.74795583],
       [-1.49320651,  1.07395805, -0.8524274 ],
       [-1.41290941,  0.11388458, -0.45725235],
       [ 0.30209288, -0.19731165, -0.9614412 ],
       [-0.49848121, -1.03820359,  0.81911762],
       [ 0.73593692, -1.43547537, -0.68436445],
       [-1.25950599,  0.57736832, -0.32552734],
       [ 1.03315605,  0.19996013, -0.35278079],
       [-0.63270921, -0.64755301,  0.0469365 ],
       [ 1.47898473, -0.67403779, -1.22489124],
       [-0.80169266, -1.50168734, -1.05682828],
       [ 0.89653113, -1.25670307, -0.32552734],
       [ 0.47826712, -0.1178573 ,  0.04239426],
       [ 0.43512241,  1.67648692,  1.33239096],
       [ 0.90611884,  0.67668627, -1.15221536],
       [-0.13294963,  1.14017002,  1.26425733],
       [ 0.78387549,  0.90180695,  0.06056323],
       [-1.03299625, -1.54141452, -0.72070238],
       [-1.50638962,  0.96139771,  1.58675651],
       [-0.07182796, -1.42223298, -1.01594811],
       [-0.23601755,  0.34562645, -0.82063171],
       [ 1.19974257,  0.31252047, -0.67073772],
       [ 0.86177567, -1.36926341,  2.45886695],
       [ 0.23977274,  1.46460864,  1.27334182],
       [ 1.0583238 ,  0.82235259,  1.89108672],
       [ 0.7167615 , -1.36926341, -0.7979205 ],
       [ 1.42385538,  1.02760968,  1.1416168 ],
       [-1.86113502,  0.26617209,  0.48753396],
       [ 0.61009818,  1.25273035,  0.14232358]])
```

Linear Regression

(It finds the linear relationship between dependent and independent variables)

Here, we use the formula of Straight line equation



slope of the line

$$\boxed{y = mx + c}$$

y → dependent variable (op)

x → independent variable (ip)

c → y-intercept (the point at which the straight line intersects the y-axis)

$$\boxed{m = \frac{y_2 - y_1}{x_2 - x_1}}$$

m → Slope of the line

$$\boxed{y = m_1 . x_1 + m_2 . x_2 + \dots + m_n . x_n}$$

Multiple Linear Regression

In [30]:
```python
# Ml Algorithm:
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

In [31]:
```python
lr.fit(xtrain,ytrain)
```

Out[31]:
```
▾ LinearRegression
LinearRegression()
```

In [32]:
```python
# Prediction using testing datas:
pred = lr.predict(xtest)
pred
```

Out[32]:
```
array([ 6.0977766 , 13.03920365,  9.6222318 ,  8.51971231, 16.04781597,
       11.04273374, 16.16537643,  9.95863291, 20.09777389, 11.00159697,
       20.80805065,  8.86112365, 17.20223308, 17.01916139, 19.62867087,
       20.2229928 , 16.13869179, 20.02913473,  7.72565543,  9.44477367,
       12.3993731 , 14.37531347, 21.04380067, 16.93216054, 18.38514281,
       21.23382504, 16.17554387, 23.24373732,  6.68168273, 19.76196046])
```

To find the accuracy of the model we need to find the MSE (Mwean Squared Error) and r2Score.

In [33]:
```python
# to find the MSE:
from sklearn.metrics import mean_squared_error
error = mean_squared_error(pred,ytest)
error
```

Out[33]:
```
3.1048626876695975
```

In [34]:
```python
# r2Score:- finds the accuracy of the model.
from sklearn.metrics import r2_score
acc = r2_score(pred,ytest)
acc
```

Out[34]:
```
0.8737580848329649
```

To plot the Best fit line:

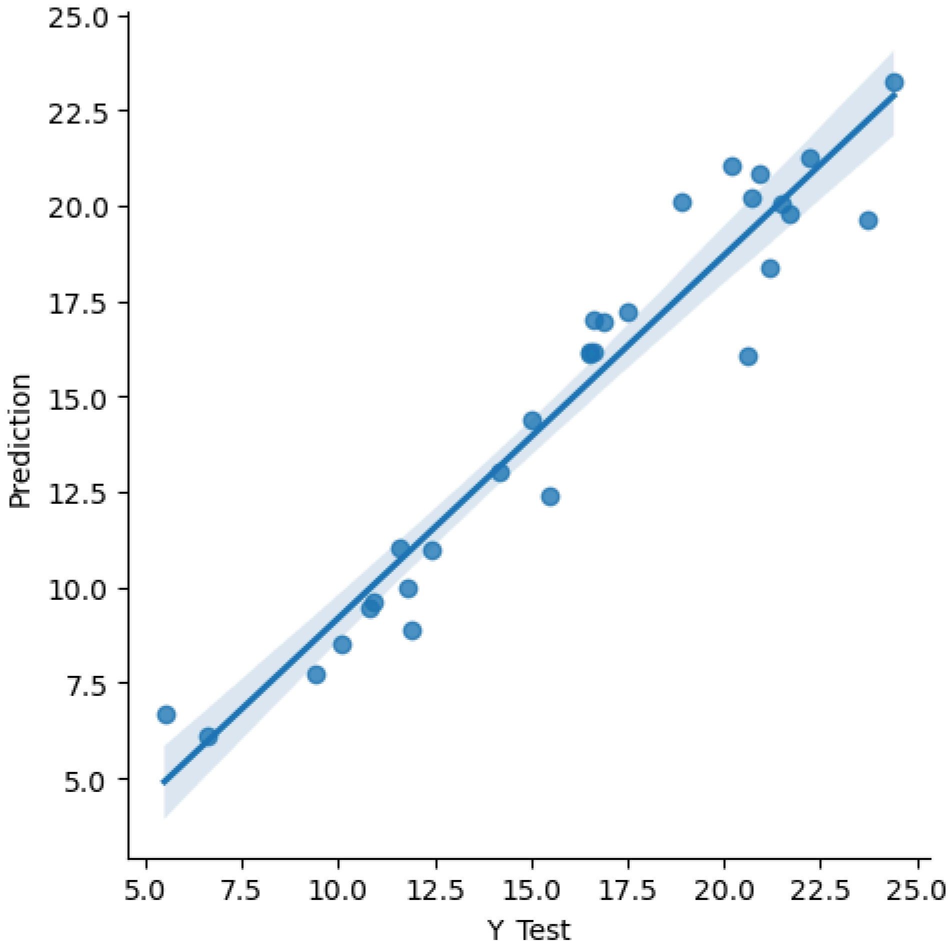
```
lmplot: stands for linear model plot
```

In [35]:
```python
df = pd.DataFrame({'Y_Test':list(ytest),
                   'Prediction':list(pred)})
df
```

Out[35]:

	Y_Test	Prediction
0	6.6	6.097777
1	14.2	13.039204
2	10.9	9.622232
3	10.1	8.519712
4	20.6	16.047816
5	11.6	11.042734
6	16.6	16.165376
7	11.8	9.958633
8	18.9	20.097774
9	12.4	11.001597
10	20.9	20.808051
11	11.9	8.861124
12	17.5	17.202233
13	16.6	17.019161
14	23.7	19.628671
15	20.7	20.222993
16	16.5	16.138692
17	21.5	20.029135
18	9.4	7.725655
19	10.8	9.444774
20	15.5	12.399373
21	15.0	14.375313
22	20.2	21.043801
23	16.9	16.932161
24	21.2	18.385143
25	22.2	21.233825
26	16.5	16.175544
27	24.4	23.243737
28	5.5	6.681683
29	21.7	19.761960

In [36]:
```
# to plot the best fit line:
sns.lmplot(x='Y_Test',y='Prediction',data=df)
plt.show()
```



Classification Model:-

```
The nature of the output column/dependent column should be
categorical.
Eg: (0/1), (True/False), (Male/Female) etc.
```

In [37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [38]:
```
# Data Reading:-
main = pd.read_csv(r"C:\Users\lab25\Downloads\maintenance_data (1).csv")
main
```

Out[38]:

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider
0	56	0	92.178854	104.230204	96.517159	TeamA	Provider4
1	81	1	72.075938	103.065701	87.271062	TeamC	Provider4
2	60	0	96.272254	77.801376	112.196170	TeamA	Provider1
3	86	1	94.406461	108.493608	72.025374	TeamC	Provider2
4	34	0	97.752899	99.413492	103.756271	TeamB	Provider1
...	...	...	...	...	...	...	...
995	88	1	88.589759	112.167556	99.861456	TeamB	Provider4
996	88	1	116.727075	110.871332	95.075631	TeamA	Provider4
997	22	0	104.026778	88.212873	83.221220	TeamB	Provider1
998	78	0	104.911649	104.257296	83.421491	TeamA	Provider4
999	63	0	116.901354	99.998694	47.641493	TeamB	Provider1

1000 rows × 7 columns

In [39]:
```
# Data Cleaning:
main.isnull().sum()
```

Out[39]:
```
lifetime          0
broken            0
pressureInd       0
moistureInd       0
temperatureInd    0
team              0
provider          0
dtype: int64
```

In [40]:
```
main.dtypes
```

Out[40]:
```
lifetime            int64
broken              int64
pressureInd       float64
moistureInd       float64
temperatureInd    float64
team               object
provider           object
dtype: object
```

In [41]:
```
for i in main.columns:
    print(i,':','\n',main[i].unique(),'\n')
```

```
lifetime :
 [56 81 60 86 34 30 68 65 23 38 29 82 80 48 92 88 74 61 35 26 63 79 53 73
 13 36 31 25 58 19 84 12 15 43  1 20 16  3 18  7 47 39 57  4 24 28 49 76
 52  8 40 46  5 41 93 77 62 85 55 33 17 45  9 72 50 42 44 54 64 27 22 59
 66 83 14 51 71 21 78  6 69 89  2 67 87 11 10 32 37 90]

broken :
 [0 1]

pressureInd :
 [ 92.17885406  72.07593772  96.27225443  94.40646126  97.75289859
  87.67880097  94.61417404  96.48330289 105.486158    99.17823531
  97.81784409  67.81225145  86.36611059  76.14465414 103.1072633
  88.41407945  84.35504868  79.66925455  86.22910861  84.17942039
 100.0059233  115.6075596   97.69718903 101.4156229  118.9786971
 102.1127749  129.1243378  109.0330362  107.2980695  127.2639544
 138.1911205   72.55408417  95.21464918  91.24713174  81.55537698
  69.13459519 132.8574784   54.73533145  84.89864628  94.15228726
  84.34406647  95.75352436 123.9284399  101.5053508  120.9455502
  83.06857115 113.3491219   75.64669299 113.4358321   81.77978353
  68.85065142 150.665421    72.42312878  97.13286233 113.965257
  57.46321326 144.3445174   80.69223153  93.91332598  73.32115289
  83.95831157  56.17177135 105.9000372  115.698401   119.7459957
 105.8937636   87.39284133  82.88180029 122.287991    85.07220883
 119.204381   107.7859931   90.67123432  66.63788383 107.8373588
 113.2726053   85.33137798  82.52821911  90.63682861 112.60533
  78.66593081  70.10171881  99.12691885 129.8155876  129.3586686
  86.43068027 127.6432644   95.24539906  83.49423659  96.21961865
  79.1113931  106.8671887  111.2558114   96.12034618  86.10751429
 108.4133588  100.2518501  117.9043996  121.6736792   60.23306632
  94.23091985  74.48747426  92.453609   112.047415   125.0326918
  96.00878453 109.6711377  154.9245853  105.7582145   93.35727589
  99.96158153  95.92226826 111.7239334   97.9282765  128.7277348
 133.1077066  126.7058619  150.6956895   80.01967528  74.57754143
 140.9858744   35.76360199  91.60822135 103.7075508  111.8094367
 124.8601041  112.7364301  102.194994   128.3172975   94.01171684
  91.39931729 102.7097196   77.4666105  143.8647421   93.91140451
 122.8634385  120.0742901   65.11291145  75.99730079  90.59983598
  85.97976393 111.6690617   86.99664346  93.62245547 143.8612427
  64.28565742  97.53345121  77.83610485  95.78219396  80.16096302
 114.6298749   82.67022466 101.1211088   85.4702501  129.5853061
 113.0908777  113.4148533   84.11160975  84.24156074  96.84048235
 112.7450016   92.35071864  73.14513861 121.1085311   82.61100175
 125.8449723   99.43119894  96.06869558  89.14378006  82.39429705
 122.5191039  114.6734454  104.1060598  104.1285145   86.49978354
 102.5436645  114.7006664   89.62264913 101.1584808   51.69950796
  81.50310722 134.9526042   89.32006408  80.94962803  81.82940534
 107.4348672  116.6937881   85.44405211  80.00815226 110.6439173
  90.84614958  99.49925475  78.77184657  71.10556819  70.35776433
  93.24543008  88.68783476  65.55163161 129.4538603  119.2571561
 122.5029001   89.3451342  105.1548812   96.43645598  98.14785492
  92.10080195  87.09276089 102.3336394   91.63775968 104.6585302
 116.8149535   87.96763611  96.14067129  53.45018161  90.38576175
  67.89336141 111.5160364  121.3890098  147.4138919  109.7446432
  79.82891144 104.6414108   79.9061587  121.6248825  110.7350681
 101.7492024   71.81758969 108.1568179  105.0382945   90.48431219
```

```
  96.57340584  93.48628393  79.0391301  124.7428685  110.0800055
 109.7472599   72.25553465  94.39762265  99.1729528  123.5105012
 102.185641    87.22145536  95.8067128  123.0263818   88.10845571
  88.67721939 110.6072377   38.78754767  89.33143548  75.81598114
  85.226111    83.58406008 108.7205931   96.61713227 108.7769039
 101.3580689  129.0020995   97.10854231 116.6944294  115.1857482
  89.40512082  88.77023945 114.142562    79.25673701 103.7368214
  96.10586879 119.514682    88.15204937  76.76859083 116.3087415
  81.94855191 114.3549146   48.59034661  99.52789867  81.37314584
  72.29605807 105.7066065   85.29060065  95.72174212  81.92521008
 127.2580478   89.75231566  90.944893    79.63958383 130.1991676
 112.5857118   86.02485068  98.94246341 107.9141561  100.5258397
 100.1524759   74.66116758 101.9277194  122.3290021   72.34009193
  84.52197479 108.0678329  102.8312375   77.30203512  92.27626092
  93.21014475  81.95766678 118.0336224  104.1600552   90.20638283
 101.3526393   84.41393902  90.44987724 113.0237965   79.53109778
 104.3627063  112.0633881  108.8375233  118.101249   127.3713508
  73.20541659  89.6800915  173.2825408   83.57488346 133.0639889
 113.2849762  103.8908585  116.246305    93.69823433 124.6300872
 114.5831739   86.90220321  71.48203946  95.3103348  111.6225804
 105.9714478   96.46384171  99.00365273 146.48261    109.8195707
  94.03852545  97.30113122  86.41755545 132.093877   114.2781179
  70.04576881 122.9730061   88.99176552 104.7316543   67.46314505
  86.65688722 130.6205854  136.3466953  117.6863649   86.81929694
  77.4245435  136.5477996   75.46380429  91.11156321 126.4345257
  93.31462565  76.06914566 106.8312633  104.7029198   64.97934021
  90.99176758  76.23814115 112.2521075  105.5142856  138.9968485
  80.7822454   58.28240383 126.2074958  111.737241    57.43642184
  84.13029987  89.42263505  88.40910573 100.777048   132.8250743
  93.80107698 116.0347608   79.21047345  63.47598728 105.4103217
 123.7075284  108.4494949   85.78027201 129.7160319   65.7483249
  83.3672664  111.9591844  133.8946635   95.31746825 100.3562126
 104.6556904  107.3121062  129.4026635   88.04750683 129.540845
  92.1617229   81.05687592  96.68465852  77.00011108 127.821078
 105.6359631   86.3692016  109.1177255   96.97853587 120.1842062
 108.5361485   79.39533761  96.65174149  83.85966038  85.57802723
 129.8158207   93.67574478  94.01292355  98.56889316  85.54546864
  97.52706923 102.1167092  118.5137949   88.37081726 110.4460737
 112.4511083   85.56228351  36.95721852 123.3028407   86.67802992
  71.49098287  69.53731478 100.2848608  103.1069062   99.16949664
  89.03305936  96.00625928 104.4622671  144.8317169   95.11565143
  66.93302741  71.22337836 103.4524605  118.4184127   57.19314091
 102.6766575  133.7792866   99.21774611 116.814456   120.2859938
 112.9778763  116.7404122   94.38858829 116.9840757   73.70460011
 121.6911516  107.2713872  109.2796161   73.82682304 100.7690515
 112.2059189   97.92286888  81.73325676 101.0718413  106.5604683
 103.842346    89.63557821 121.2374588   99.29371474 141.0425952
  62.11775037  86.45206246 114.2375319   82.87927937 111.0734869
 105.6715844   81.96399134  88.59556354  93.66493104 125.2429312
 122.4142712  116.5570977  124.919901    91.48387459  99.20956467
 105.3733927   70.95902236 100.8647969   62.33579998  98.19445357
  94.55845182 101.7235342   94.60239889 110.9349035   68.43805127
 124.060321    98.01613128 146.3333418  143.8070826   94.60611059
  83.2948559   92.65273389 115.0194282  127.2677998   95.91575356
  95.07575508  87.50831657 129.9664351  143.5872596  121.3920162
  98.2261406   67.47327555 105.9076314   93.09981147 119.1369483
```

```
  88.18621311  86.00046399  79.74372147  91.05138614 101.6767349
 109.5579662  110.0071726   69.02317525 118.4128433   86.0706438
 152.2472591  124.3806396  130.7760286   97.00377888 103.2224299
  76.31401714  71.32208515 137.2128229  126.8191693   88.01620789
  77.528434    76.19664829  97.86140584  86.14392774  98.56261935
 113.2269285   77.90980281  73.43908733  79.31455063  82.89800458
  83.31611894  87.51321781  69.93989757  80.40071612  71.98632777
  79.52418296  87.0741334  108.9292165   95.06305508  98.41174596
 105.383124    75.17163243 111.2730416   89.10571119 136.363399
  98.4683561   64.10778851 101.1146141   87.66982255 133.4835583
  92.26027172  96.70897012 109.6262398  101.6326601  102.3409431
  95.03146636  97.56399942  98.32962285  98.85788463  94.68950331
  96.87798631  91.64573951 112.4033275  137.3669077   87.71379376
 110.083131    94.00602087  68.66148375  95.32397567 116.6319208
  86.82007041  83.41775869  83.86989415 123.046048    94.60705053
 112.8939709  109.2597706  127.2669194   99.64954629  51.46222395
  67.46672736 124.5444495   70.23154327  93.62504245  80.8034882
  90.0703862   99.94713693  82.48807497 111.7661724   80.53923807
 119.1471724   75.24853056  97.50182643  56.91870478  96.10524366
  69.04580099  83.12436689  96.0259551   62.21239864 113.0339432
  90.24452612 115.250734   110.8812294   73.64748388 131.5024077
 130.0894769  107.3409195   89.88222274  84.25104439  99.55228351
 105.7059853   63.68829486 104.171491   110.8689295   85.20434836
 132.6130061   90.19991521 105.8495106  105.5088984   84.75126749
 107.641318    85.58348492  93.28289225  87.59504619  92.15915621
  89.13264319 137.1074186  103.6970951   85.33211838 114.6718716
  88.71833465 103.9198822  116.2051523  112.5047783  112.2564357
  78.91818984 100.221632    68.83180412 100.1303832   92.97667487
  81.60561713 126.1653198  106.0579014   83.84387513 126.8592528
  89.9994371   96.53224252  99.23940261 115.3062038  104.6215697
 111.3151668   99.1679382  105.0034653  130.5173334  106.1000779
  79.25573205 114.6843238  146.4982724   95.42828461  58.57504538
  96.3452806  114.0235311  117.6266596   58.16388876 138.2638844
  96.80196127  89.23363155 136.5810149  117.9904551  121.6938075
  83.50915384  60.1987941  120.4946348   78.35441224 105.6467623
  99.73112801 100.6223605  116.7030362   87.63602833 102.5176749
  92.94336584  62.39240693  92.50877794  95.12069014  90.18068383
  90.08072329  90.03014039  85.12634681 141.3049796   69.29337314
 114.5215235   66.05914598 101.2279631  110.0090321  102.1307254
 102.5645544  113.4407661   46.60128721 111.5163715   99.46358828
  93.7971681   81.67473275 116.6123569  105.3398198  102.5285553
 115.5279271   82.86997595 123.1083489  113.7242812   87.72727391
  92.5804527   72.87854742  84.53736236 121.7053459   49.60538555
  92.48346899  91.58886359 104.5512445  113.7301442  116.5487897
  87.73855425 121.8713508   70.36458943 134.4874366   77.20681327
  89.50688382 101.7687594   92.17880634 126.1254999  113.1199934
 113.6090937   88.04593898  80.44794663  98.59001085 127.5206011
  93.7078384   81.28916813  92.540717   135.587121    81.75718946
  65.64282279  94.41582296 106.4488937   82.17819319 114.798646
 111.6453989   95.61878045 119.0728916  114.6897687   97.61987153
  33.48191737  82.42322758  72.01005056  95.83270361 112.0956784
  82.51900511  96.48678974 100.1031836   87.81222539 113.5169831
 110.076247    81.36962424  90.14957518  91.06149269 119.4140619
 104.7586955  118.3935992   92.68039619  94.15601843  93.24211823
  61.94270337  92.43671384  75.0604639   83.32483885  87.64751401
  87.09320002 122.7898163  106.3317345   70.9206612   83.03044607
```

```
 113.068106   113.601145    81.07606688  89.22954094  92.01036137
  57.60764935 131.8385011  138.9963047  107.1796832  113.5985097
 124.1530955  108.3393848  117.7823809   89.35061975  81.41411179
  75.80609176  89.06199119  96.70986038 113.211629    81.94738031
  85.56227794 104.6820796  115.8488571  114.2240915   96.96522323
 114.2631231  105.0747055  129.9217437  119.5429652   90.20118571
 105.5712867   85.21625561  56.90766195 107.2584217  102.1557371
 112.1474165  101.8051589   83.48699368 126.3497155  124.3589233
 114.8598046   77.70504868 117.083811   111.4722536  122.2064503
 111.0257141   76.81390329  87.41407071  95.4696821  106.1130127
  86.07886827  94.46946956  95.21735068  85.80748956 138.729936
  97.32105012  88.46874409  83.45815627 132.8335446   64.84102158
 130.4633019   89.30661009  84.6746399  112.8676307   99.55729989
  62.86186865  96.4221917  120.1676238  103.3218849   90.16256001
  94.24723246 104.2805506  104.4344592   39.78317591  97.87080501
 101.5846744  117.0221942   79.97226263  87.42199024  62.64646201
  92.77751204  65.25918125 139.5743417   86.69146311 107.5901149
 113.4352002  106.6969636   96.17271983 103.8805889   87.91652826
 106.9841215   77.26951694  92.32067629  80.60779421  81.13096087
 138.3759366  106.7294506   93.78066682 120.3512884   90.57959798
  42.44635676 138.4454389  127.6631691  100.4601827  109.5958521
  96.98795347 166.7858975   91.16611817 128.9411083   98.83673629
  80.42128326  84.46620776  89.95337574 121.1658905   77.12083796
  81.71752422  89.74268264 101.6701795   86.88757075  95.59981157
 103.0135453   98.68886709 106.6486289  101.637611   128.2207485
 101.8850304   88.55083112 122.1087757   93.45679795 104.5401682
  74.82551372  74.97672381 100.7887618  133.4265573   75.4853782
  91.06771541 113.8749271   88.40955072  69.80779035 129.1863512
 112.9292352   79.59452653  90.78568329  52.49765001 110.3339493
  64.03563589 121.9064549   95.28030654 119.0011817  103.8813764
  84.29478616  70.96539336  71.92183381 112.4122561   94.57927475
  98.14966037  98.37929005 120.6151502  118.0070529  110.1976712
  80.19902714 100.0915524  105.5421334   70.16567395  75.6867736
 150.1404798  107.5213703   85.73088376  99.65564682  56.72643095
  81.71098194 102.0595059  103.2401042   84.58244428  82.61501033
 104.4124785   70.76412566  99.68203559  74.03205981 143.5503382
  77.69900009  79.74938628  90.78063751 117.7069331  119.6780052
  70.16802889  83.27692515 111.9914415  102.7767557   87.946098
  92.57385892  88.20535721 140.3729067   95.30752801  76.76990774
 119.0339468  113.7665009  109.7687115  125.7018715   93.88255716
  86.98757582  87.85255835 115.7507865   94.68396594 110.2297466
  88.58975909 116.7270751  104.0267784  104.9116487  116.9013541 ]

moistureInd :
 [104.2302045  103.0657014   77.80137602 108.4936078   99.413492
 115.7122623   85.70223564  93.04679747 118.2919966   99.13871701
 111.0741675   96.10784622  92.56197159  93.97345443 103.6731971
  97.36275458 103.1749187   95.92690751  86.11189974 114.0740657
 103.5028764  103.0135727  101.8849607   89.75374604 105.298916
 106.2506946  103.3447202   92.7541199   85.05529129 103.5572918
  91.65626896 111.3764192   93.07170934  93.03728223  90.69889367
  81.98012234  82.0372826   91.32443271  84.86790095  88.44665868
 103.3770521   92.31580754 102.3402566   77.65982051 101.9858519
 104.1607096  114.4731859  114.1988574   91.82061858  91.96565385
 106.2142358   89.13903219 120.4841552  117.2449063  127.3026074
 104.9790894   95.12186217 102.9254586   91.26211515 123.8980003
```

101.7508802	88.91703002	104.2681687	105.8508457	119.1046335
104.39546	106.1662339	91.02517864	93.84835213	104.1094647
111.7721561	117.3903359	89.5638833	92.20583056	99.83817218
114.696552	97.04522638	109.6384237	101.8016105	100.2743409
110.5184077	92.55906673	105.455789	104.0702758	95.48779959
104.6939431	92.64479758	115.9503864	93.40046999	90.66213673
73.6071607	118.9483574	97.35998621	105.5761964	93.68407938
112.8680361	93.34673498	89.84703399	100.0076838	107.5271185
107.9583453	99.10009387	92.49535193	89.84959363	101.1634821
84.58755361	104.6003316	77.67724907	112.4089301	89.29033639
105.9447877	102.5904676	89.93333376	96.30706049	96.5898045
104.051493	92.44197037	111.9887615	120.844063	113.001844
87.25006628	106.3355289	114.8617036	92.41019194	75.48027478
118.1497781	92.37349843	93.38983164	106.729157	91.93279642
99.91949583	102.8467778	89.20179928	95.87773954	106.0364132
115.9176028	102.9920732	105.453561	96.3718599	98.07438142
104.2711849	93.27223301	91.03846392	94.01478639	108.4817025
114.037572	96.30481564	112.8325443	112.7934006	109.9425243
102.294442	106.2353533	87.70386034	111.5486787	93.69210818
109.3698093	108.0811633	118.7811884	101.8366491	117.8289515
91.61766973	106.1939232	95.24348535	95.09554947	113.1563812
94.1146668	96.43331087	105.2206332	106.3110111	96.82672815
100.3323366	91.86074592	105.9628557	91.5325882	86.63641816
94.82559513	109.0935529	102.9938823	89.23415453	58.54730111
104.1141034	88.4801008	95.97596256	106.3025588	104.9825534
90.82303655	104.5255342	107.3373551	92.94532758	98.14071808
107.2455027	103.3831611	93.16231384	108.6931765	93.16992001
89.39662465	86.69133425	89.23078421	108.6197469	100.6706946
87.08359987	102.957647	95.19828835	102.8929828	110.3888521
89.99461085	98.38474842	96.42298434	116.7340858	107.5376658
85.58815181	91.20086693	79.33497737	79.64982916	104.5337845
99.51692219	99.11730398	83.71084563	87.44533599	89.67105476
101.0689842	111.4188395	77.43338581	98.69430428	113.2917759
92.01452778	100.2809694	102.231344	93.95323864	106.8332232
108.5498769	92.9159865	106.4118914	97.85352326	96.84071815
108.0530389	117.6716536	95.02644317	105.2015316	105.9690406
89.06691257	87.38629913	92.91202985	101.2923412	99.41904421
97.52182754	103.8373069	116.4918169	85.15218999	106.1237704
114.9718442	79.72451035	100.8918579	110.2104906	87.49910161
81.41299882	93.13240058	103.1132546	106.4850719	101.1496633
89.46824597	110.7154578	97.90547112	97.42450677	97.68297405
83.88845914	102.3775797	106.6021446	98.69889549	97.67504843
109.6314486	91.16757756	107.0189746	94.57180405	91.70991941
98.65100391	103.4861975	73.29862984	104.6947288	87.59759596
97.65391673	106.4519667	92.3535606	115.057383	97.95620176
101.6142635	80.53200774	90.31914773	107.5154579	108.7645407
105.3768446	116.0870869	98.61359172	99.430185	104.698884
93.23293592	106.343999	93.54105464	95.24526849	83.69236521
74.96813943	112.2018487	98.52804145	87.36072497	101.7319388
98.46945124	110.8229509	113.957099	104.8803362	100.4875962
88.17677958	112.2294842	94.49523742	109.5193655	103.5734976
87.27730432	102.2533692	111.6626543	106.4894726	84.46012536
102.6527809	108.4477704	80.81829191	106.250103	94.23652593
91.77533546	98.91228582	89.46945152	93.46411836	93.89606386
87.36748742	120.2509566	97.32850275	98.84825154	90.71151167
96.57974053	90.87000417	103.1555992	96.41462829	95.16679302

104.9802014	94.18413115	94.08955846	102.8613655	100.3128302
110.3557443	117.5362464	96.60797615	105.2636914	85.25841804
89.95804395	97.29373333	113.00593	103.1377918	105.3148247
97.26276037	97.0736997	101.2135811	101.7302246	102.1024831
93.95345499	106.7215997	102.8398951	99.48156441	86.86899972
94.55384281	99.04385992	108.3094338	96.47043186	113.5861283
97.39921807	117.8713739	96.96752761	86.18997764	103.2915252
95.22043506	113.029899	91.62352994	101.2841864	81.93303627
95.05824196	107.2553883	92.84483672	101.1895895	94.92631364
112.1436343	117.9394084	105.3395675	80.63088947	103.8248007
93.5158375	99.27197297	105.4037907	108.4000402	92.12162741
111.7654021	97.77739652	91.47319167	98.53532535	95.99036544
92.74172011	80.8678127	92.77764496	80.58066955	82.46557766
107.4292572	94.7009006	107.9781424	86.36248475	100.6729505
96.95590895	101.9202298	107.9943727	106.0739247	103.5603805
101.249552	102.3093935	96.81729627	88.25204726	99.59673272
94.6896782	102.4185949	106.369383	103.7623283	97.62710347
93.2324069	82.61246511	102.4336216	100.2276629	109.0407797
102.111698	91.51747606	100.0798604	117.9077792	84.65580465
95.69960941	91.64444738	99.76755611	98.54147055	102.1576218
88.96320258	103.1399645	85.04469833	96.56036437	103.4541368
98.3782401	104.2045991	117.5545718	85.83568366	95.0772386
96.44820999	89.43047465	102.9387744	82.78028935	111.5098444
101.9365276	103.6839047	95.6129819	101.4597725	92.93064673
88.89829459	93.41569831	88.68012128	98.01543531	113.3588218
95.0504184	85.44861354	104.4609854	99.50867566	90.44967458
114.5880459	82.55990559	92.01723212	114.6482201	96.11064846
104.5029136	91.861367	110.8317553	103.915682	102.7426686
87.22617684	91.05610686	108.0251614	92.24498992	94.19122048
97.76062025	87.62102969	99.90417874	93.21089513	87.65995975
106.6025178	97.53770278	98.23750591	106.3239601	113.9721223
107.5046306	88.29525666	113.8623509	84.26949324	95.06581029
104.6095164	90.08488794	95.76767249	87.52940523	116.1570602
102.2590175	107.3156409	116.0320561	103.4661907	101.3228819
116.0388643	100.1340867	88.19704009	97.52466441	104.4654632
115.1305817	110.6769931	103.6400105	98.19892544	104.9150315
104.4650216	101.5432185	107.1000442	104.6199843	88.18026489
96.53014316	101.3429664	108.4570164	83.31109493	101.2847496
102.3763566	102.6919007	108.5911603	94.86906416	96.52450221
128.5950383	81.71882305	104.0279567	97.40118033	89.93110271
102.6538603	115.461282	86.29238721	116.0775585	91.96023876
89.02241135	123.2676935	86.16872259	105.9859448	101.0983454
106.4173177	99.06887516	84.02775136	94.89606948	89.90204761
100.2308122	97.40847326	101.4435887	122.3591251	96.44604138
88.16059474	111.7631163	99.70414132	82.84151939	90.80665259
99.04378211	97.82876698	100.5938109	94.59619158	108.2822108
98.32454997	103.8590033	107.7383667	98.26301894	82.1579423
112.9812551	102.1411703	106.1703546	103.2903791	113.1316562
107.2141766	91.62927168	122.3492056	103.7544081	102.8531259
104.7427573	92.25017384	93.92932564	104.9092518	100.5511169
111.2280979	94.49411538	118.9048733	111.7793293	101.63745
86.51678718	88.07971633	94.06325121	97.22459227	105.454307
88.69783307	106.8743245	92.22803866	74.74741543	116.4932538
104.2146142	103.9545199	101.7665745	97.98055543	111.219653
109.5234175	112.158815	90.74898805	108.8783622	106.5461545
96.36774655	89.53799142	103.8470332	101.3459037	99.26548504

```
 93.9014404   87.04011662 106.3856823   98.5840232  108.8334494
 94.92467534  87.72408077 120.822109   117.3216561  121.8106661
113.5152512  106.1123531  112.06361     90.70366187 102.9923461
108.2812434   89.75347336 119.0872405   92.97577656 106.0197614
 79.23709873 104.4671814   95.81761984  95.39800589  85.24114778
 80.50030303 101.3705708  100.0075896   94.58529922  98.64137136
 88.6523958   93.49884742  80.75135411  96.46360873  92.55529087
104.9729116   96.26037495 105.9369748  105.9267071  113.2484212
106.4346117   99.38605273 113.9079658   94.24235329  99.84788693
112.6724587  112.8651383  101.0899078  104.9358887  107.644111
 91.29465417 111.6055839   95.72529243  83.21343527  91.0466142
100.8711342   96.99857983  92.82247469  97.06000359  92.38315302
 80.19900228 112.2903308  120.355508   115.1030776   99.5059735
 78.91376847  98.09015003  99.4532466  108.1085324  109.4756503
 91.9105934   89.08275675 106.551167    83.91188494 104.2512406
114.4258633   95.62513027 112.5891621  100.565399   109.4179077
110.5097425   91.08115049  95.88523063 111.6681873  117.0454427
115.5031125   84.56255085 114.4996363   98.31505149 109.1463596
 92.23087548  81.85356258  99.96354325 106.0510666  109.6696888
102.9124784  108.3881143  109.9657864   97.6195071  117.5135353
116.1966992   99.81103132 106.0971892   96.53701627  94.56565418
102.6948173  111.1621117  103.1169775  102.2540164   93.29411388
100.0435914  107.835907   104.6835089   84.04373237  95.44711572
 94.37580023 107.2465113  101.0721874  102.2623872  101.1092512
 83.46267061  94.6499297   87.76108377 104.655602    75.91283991
114.396699   105.8568185   88.18427275  97.88356459  99.35744751
 76.75852253  95.0731171  108.4005876   88.83202799  93.48883157
 87.89884994 103.3933774  111.1276651   92.65291582  95.60150259
102.3103093   81.03770521 109.5603947  102.60874     95.34112341
 99.43773299  96.47253257 105.2449218  103.4070849   76.99989789
 97.36370029 109.0622125  108.5000483  104.8846811  112.1013459
126.3110635   94.3830959  103.8757566  123.0806296  108.7066627
 81.57661877 103.4157814   91.73103628  97.97364021 112.6685491
100.2632976   81.32268697 107.4080993   98.72240236 107.6538483
 94.94439656  92.24315126  92.81724306 114.8303256   95.8537947
100.5543245  107.5304932   97.67299844  84.26251393  87.03884506
 97.44047556 102.1765619   94.5356663   95.22812545  77.84237127
 80.41630602  99.90487827  89.49637124 102.3871274   90.12448188
111.9108143  106.7360984   89.61652905  92.57438525 102.8360181
 86.83758483  98.55536282 104.5983261   94.43931888  89.42175577
101.7670565   88.57300613  70.92881491 108.2827754  103.4316575
104.1635614  116.5310626  100.5588038  119.0122866   97.19907991
110.4739002   83.97763667 101.9919728   85.84924473  94.67305419
 93.40602817  88.48760552  94.77453536  75.45927052 102.3813186
 96.4323304  116.7277592  106.1197597   93.02524417  84.25043595
 93.09866928 104.7434586   83.76541926 106.0307272   87.63241888
113.0289302   98.71708976  87.14461217 119.1295103   75.06279953
112.9621698   98.22750893  88.73972756  77.65152004  91.43521609
107.260203    97.61164939  99.80887167 102.6579809  104.0255241
 93.19685586  91.2739045  116.1120162  110.9415742  101.5569311
101.0364492   98.8748043  100.5087382   96.49318685  87.91663016
 94.69792313  97.21304372  78.30528722 112.0110167   73.43455423
 91.1878978   82.99944993  99.02341879 102.9776209  104.561665
110.5161483  101.1249579   95.35328972  93.77154151  96.70531603
 92.87398652  99.62344687  95.73106664  98.82130347  90.89258319
 96.98695666  97.24421296  95.08496803  97.25785131 104.9088072
```

```
 97.10285708 106.5032417   88.1673482  102.8413903  103.721079
101.3424149   85.25810757  82.07504486 106.6893263   97.55150136
 86.58222924 125.3812758   95.08878787  97.60178301 110.4003294
 80.20631364  95.97581186  98.82456948 104.029526   108.4201565
 86.26659608  83.63147022  98.59732672 107.4336744   92.61947424
 99.41313697  91.0343136  111.8542572   83.67069847  90.99237878
 97.33062442 100.1106956   94.05537051  90.09954474 113.9085168
 99.82619873  78.64690295 105.6102415  108.5672605   93.1331722
 85.65029662  96.23301107 106.8901076  109.3399773  100.776327
 84.2348114   96.96697723  86.09934693  84.5964103  109.9128786
108.4494863   98.56383772  97.28968823 101.9694427   97.27128211
 94.52318723  98.97375537 104.2604324   95.30300146  92.31284639
 99.14965123  83.97148642 106.61584     98.42001087 102.9596453
104.8934097   88.90264367 115.8417289   81.48540424  97.50586632
109.9499969   78.281154    99.8653775  114.818523    99.67938347
 93.12458318 109.7798078  106.2575243  111.4517798   96.13748874
103.6563739  110.3469315  104.5390305   97.49013516 110.1044455
120.1119323   88.42367648  93.37811705 101.9967219  103.1290149
123.6561241   94.28668724  98.76222896 108.9614215  101.4455226
112.1675561  110.8713319   88.21287267 104.2572957   99.99869359]

temperatureInd :
 [ 96.51715873  87.27106218 112.1961703   72.02537441 103.7562706
  89.79210466 142.8270014   98.31619045  96.02882218  95.49296461
  94.94244328 122.3718087   96.66794972 108.9442729   79.50453248
  89.31981296 102.3997151   99.73563529 111.8425483   83.62744989
  96.87603499 101.6168426  130.135676   114.909434   115.247085
  92.31519517  92.84635902 124.7604446  109.8410029  133.8596693
 101.9259278   93.54141963  75.84542114  94.45839707  97.02690353
  81.03392506 106.7192651   91.42479473  96.5560518  135.7447785
 113.2754593   95.32702839 110.9507609  107.6560938  101.8841017
 124.3629153  101.5657772   86.81793729  95.36508541  75.14654634
 115.7119699  102.6799026  116.4004334  101.5042043  107.2647361
 100.5280146  127.8402262   74.67365277  91.48661787 129.4396901
 117.8981463   96.14338728  54.57362995  89.34379968 112.3412826
 118.3853166   84.52171258  94.62007382  91.06499024 107.7140369
  84.24208522 113.3773084   86.38418467  99.06197865 111.0848528
  79.69982029  91.4131354   96.59799612 130.0559851   87.98363112
 111.0784536  131.734206   113.4644369   99.8617326  104.7582439
  86.39617998  98.05514152 100.4178602  121.2977598  125.8385221
  74.8746231   97.42421086 122.0342734  124.0671196   83.15696087
  94.82251917  88.15086763  73.17482049 130.3035019   62.65714424
 110.6577848   79.52161712 140.1600911   81.10095313 126.0240766
  95.28725334 102.8819567  112.0488069  112.2429165   99.55773664
 114.9973601   89.99548863 112.7274067  109.4866982  135.2507754
 120.9586286  104.9071869   85.86354729 116.3064452  114.5189329
 125.4733721   94.18852214 104.1536206   91.100293   112.8468411
 113.8879107   85.48980065  70.11393139  94.7417842  111.4627321
  80.8328907  108.3242991   84.53910754 117.1610923  112.5418604
  72.54074697  91.29538601  97.05480756  98.93408655  96.2636136
 121.4452996  119.0101687   58.58520759  70.85057767  93.23788677
 123.755117    76.90808405 121.7863711   98.6350135  113.9839003
  51.1069448  115.3757052  117.6781559   84.48907863  84.91572472
  95.60612507 110.0561783  106.9423282   80.45487363  73.47330364
 100.9545667  107.5371792  120.7810241  104.7811532  130.5451632
  88.68996986  55.91119683 120.2784907   76.40645579 125.4315376
```

```
 93.44364743  88.2958658  113.6162542  110.0802737   92.64598661
 67.16846487 108.6968184  142.0284953  111.1869777   60.51820437
 73.27573124 108.6773664   98.23716736  96.25748233 102.2891495
138.4133405  107.3384773   86.72571576 104.9994585   86.39503297
102.9288987  119.7507572   85.02431101  81.73936165 149.8722976
107.1840475  128.3799916  148.0145809   97.54660485  65.04109861
 73.00784694  97.16886592 116.9706253  114.8533917  138.4908163
 97.29228958  79.95705583  80.75166079 125.5531059  126.3743168
 95.15687347 113.2602184  139.3520024  103.6298501  111.3231862
111.8488205   94.17953507 101.8065298  100.0408353  138.5473255
 99.36824612  96.51154379  71.16947247  74.75057567  54.53286892
103.7654068  102.2693672   83.4708124  120.4900002   55.01166325
112.9562364   76.24447939 102.7117398   98.34049821 100.8559716
 96.14239799  89.68002082  87.03925358  56.1165184   92.65344596
 93.84971699  86.81281172  87.67209364 113.8098361   89.83880804
 98.97230007 121.1524524  125.0025263  108.5998878   96.25360918
 85.41919146 124.6367074  100.8783625   92.23282144  89.65043059
105.7679599   84.84584223 121.280986   107.7119541   91.47168833
 96.83404154  79.61013172 165.1699551  113.8027764  104.781466
 76.37176899  97.89313354 113.0762836   90.75000591 107.035628
 89.60336669 117.1695675   82.67233951 110.7814975  131.4217169
 89.75361583  59.64195681 108.4834019  111.7138916  110.9066888
111.9159592  141.0827592  140.6030213   87.22372543 126.0767394
 80.83568474 172.5441398   82.09243986 124.397205   111.9841191
112.6853212  129.4902715   97.90275087 113.2887586  124.9275537
 87.91034665  87.00380701 105.1493577  102.4544526   78.03294027
120.0701073  116.4980729  131.5920746   99.85630851 115.1979378
 73.08420955 101.1741394   72.40684563  86.11663572  53.46082223
 80.27367989 147.0523134  120.4209254  107.3707073  111.2251287
 94.81097722 108.4548245  124.7089102  107.7732534  108.8989324
109.5798115   99.58076829  92.60063967 100.5615017   99.52129916
 85.70228144  88.99686622 132.7477118   78.12938193 125.2775595
106.5517152  107.8049309   72.77187235 106.9790328   66.85327645
127.2913514   80.6965672  137.1040771  123.0306319   88.99069701
 70.66169669  95.63839735  89.41488442 109.6280935   88.61622406
105.599702    72.85384667  99.96954104  69.38300103 115.9266443
112.2726752  105.3899691  122.0091321   99.68259133 107.896672
 90.12881011  83.24985338  79.91117353  68.59001585  82.67734265
 81.09924955 123.1181318  103.1035794   62.08682761  89.35933861
 87.47371346  59.1837539  115.5916939   85.47370017  88.2030026
 92.50226444 106.1169784  102.8770082   93.77437822 132.9173555
114.1946583  118.9841134   98.48603325 139.5592476  110.8174363
114.3150001  125.340291    90.2360567   99.92397973 102.3436485
115.0931378   98.52943386  90.33291737  92.21326954  86.08794063
107.7381429  102.0307612   87.55605965  90.87088496  72.37703226
112.2239802   68.74486465  92.59096814  97.16121249 104.1203675
 98.90734463  52.58127457  98.68664627 121.4755021  132.2585991
 73.02532901 115.1788376  116.7485215  110.8471409  118.5266303
111.1647232  117.4509664   86.82420605  62.08097086  95.46044467
113.4265545  100.7171218  127.2798115   88.85615429 100.2521234
 62.71681205  97.14373123  87.53799466  49.0321527  107.5590712
111.7422413  120.2912622  106.1875516   70.15612853 105.1147708
120.457516    95.30945357  78.32082733 108.9362241  122.7390448
115.3685257   99.06341525 128.0759255  112.487125   106.4484335
102.9798574  104.1215429   71.46352862 122.2757262   90.56827189
125.7614657  129.3847289  137.6645123  104.9869855  105.5116164
```

```
108.489184    60.30010334 113.5224963  121.2022877   97.47350299
116.0347362   95.77434809 134.6336561  100.4683095  121.0338278
 93.43482374 119.5139037   86.83974385 122.8656317   91.31055355
 84.07694229  86.04541606 139.9407679   84.82289168  89.94547053
120.043249    69.81572205  57.07911388 104.0389086  122.7510846
 73.54281829  86.41816173  81.48699807 105.9797527   76.15820848
 72.06671706  83.54544289  86.01073964  93.70366556  75.88332481
 81.06775078 110.2688197   76.53834937 155.3838857   91.91921815
100.1367328   99.20396556 103.2092373  122.020558    68.8084124
 94.22380183  95.71042159  86.31523612  93.03027489  84.59359472
 68.87349656  74.33315664 110.7243672   95.59879603 131.3832723
 71.95119998  83.05283881 119.3019094   71.49777571  82.66317043
103.5252337  111.4212249   96.94715708  78.61742661  78.75495455
126.3921548  149.4467203   49.81990264  73.37333135 137.6061546
 68.16686514 102.6114378   94.5256947  120.7035003   99.3788547
 77.80657861 117.5391579  133.6240548  117.6669629   93.07884665
107.6887313   95.97696993  99.86760889 128.1063994   49.84846939
 96.48873927 116.2701676   94.90692673  78.50646147  77.27328413
106.3237978  104.0242329   91.86198316  94.01098349 109.951768
 90.30989822 101.0064163  107.6951275  108.0082866  100.7630124
133.6504699  105.6360835  122.1968428   93.73461694  77.3760987
117.3528031  133.7448317  102.2526685   89.43135826  86.11773347
125.6750718  102.4023769  112.4182785   97.3135437  109.2338791
133.4637814   79.98840239 115.5741047  114.2934056   89.05813298
111.1276741   92.34526693 111.4792895   87.22526474 121.869236
 96.62748214  55.62637359  71.75791101 109.4110162   83.1886714
 95.79943674 148.1284018   62.54817493  95.82592896 100.2720256
107.3606306   70.1168102   73.3061197   93.43342614 106.5114031
 89.08602618 130.8823582  103.6496815  126.9995764  116.4650161
129.7877454  100.8153492  118.0945448  116.8030671  117.1148201
 44.05861937  53.86889154  80.80015925  94.92339697  97.14318758
 83.07660209  92.11820297 129.3959183  112.045723   107.9998787
123.3471888  102.0829817  105.0559928  120.1119781  109.8030349
103.6083438   76.75715125  90.89449635 137.7316507   91.46616916
114.6999018  124.5943298  104.0160769  121.5605208  119.1640431
 86.81560948 148.7948773  102.8293459   92.40268117 102.97026
 90.9112721   92.08460808  95.21773293  71.41243434  85.15456641
114.635357    83.12919201 107.2202127  139.8641702   90.69467731
123.5756211  114.6860223  114.0438889   86.09168244 106.784924
 87.98640894 112.342235    77.62546519 120.325407   101.3321156
 80.05832135  87.98386127 111.4629184   92.79820374 109.7438203
129.9878939   76.08557046 100.6230519   75.19061894 110.2672376
 77.21030463  93.00411663  92.92996339  77.07077762  93.34575008
124.736851    76.55626608 136.030098   128.7880801   75.9997621
 84.81261253 108.8943008  107.4600703   84.04405964  73.29702688
127.4278908  124.484758    87.67851893 128.812916   112.167476
135.1693172  101.8822342  103.595199   120.4686983   73.96276076
103.067457   126.8485419   80.72306687  85.16860071  95.60684471
118.6429881   88.76889673 106.5479672  121.0385472   90.37342086
 86.140575   121.6596974  104.2853153   90.687831   111.6442027
107.1263062  132.8960253  112.0732419  108.5662      80.10887306
 91.46874009  98.27463019  71.80056052 123.3052926  119.9772055
117.8775732   83.42249496 137.4741755  105.3091613  125.6032844
 92.37923722 105.1152794  119.8540501  102.4971134   42.27959777
105.8270944   99.98644583 141.8914187  108.3030495   81.18539743
 91.39246184 102.8762405   96.76267759  92.79302585 100.4592065
```

```
 104.753569   114.6001775   93.4643859  120.353069   111.5351793
  99.81819568  99.63780661  67.25648332  90.11035352  75.30231632
 104.1224911   90.7504789  112.6088848  127.0598802  119.861342
 115.0832449   80.75742109 115.3353489   92.80935236  98.51566602
 113.5163748  107.6967466   93.35842227 102.7250535   54.85463932
  79.45524105 109.7863541  116.1649707  111.15933     88.37180867
  97.35961631  83.03736305 132.4949068  102.7298925   91.10385148
  95.52186729 134.0993917  100.0956146  111.9665528   77.19401385
  89.47804982  82.35935334  82.37295228  60.61123187  85.28897211
  99.32586687 110.9381936  121.4210197  105.3645153  110.1829006
 127.5978295   84.35551244  57.34534891  65.21580585  77.59314255
  96.76811894  87.05655228 107.4768874   76.54239502  92.02624326
 106.5518926  109.8760155  123.0878047  117.8925563  114.1714076
  73.99386306  93.02570296 104.9696808  111.5631001  112.4952072
 101.92534    121.3259062   91.72599463  99.08231665  76.29286351
 101.5804487   73.71605989  91.67263447  81.90237741 114.5670649
  95.9716024  102.6007936  138.311966    81.39597051 104.7856021
  86.63390737 130.5570767   65.18346688 114.9243374  119.5940377
  63.18449738  70.92465014 110.7122015   93.91395318  87.24826931
  82.13596405 106.7305167  128.3033183   80.55511582  46.85704695
  91.61400596  64.88013098  96.95931027 128.3332476  112.4178876
  80.99028054  97.91723838  97.78208935 120.3658475   93.40801931
 149.924611    80.76109253  88.43730614  96.3568589   73.31904291
 114.8743613   86.83447267 108.4264351   84.6399811   84.74365498
 109.8976734   99.76389665 124.5358699  119.151983   101.1478423
 117.1686465   95.26037259  71.32737145 124.6895599  110.7019087
  77.81619344 106.867088    92.74548648 108.621829    78.4914549
  53.68552772  99.18399217 127.3901657  129.6915102  115.4800775
  99.01180073  86.52544178  69.38011409  72.22892252 122.1541802
  95.78505053  93.85869715 119.595801   133.4597497  135.8513241
  98.22838553 125.4883602  113.188661   103.6789277   93.48403916
 112.9856511  100.1692002   82.56648328 114.2516554  136.1321316
 105.6019068   93.4746867  133.6236876   76.443343   143.3595386
  62.95617977  89.58568737 101.3501698   88.69819983 114.3900269
  91.440331    96.46031059 116.5434247   93.0974028   78.25048078
  87.017921    44.33108441 102.3655284   95.55979966 129.203039
  69.53376531 123.3805805  110.8264747  118.0431521   85.37598469
 111.2401286  118.0588863   96.96869193 103.4074918  104.4062563
 109.5085553   89.18130235  66.33906123  95.73557981 108.0423921
 106.6359452  114.3053581   92.50320389  82.23645826  91.12806915
  96.27540641  98.43332131  84.73797429  84.82670487 101.7005933
 106.3987993  101.4990011  104.2267675   90.21907271  92.27922542
  93.81460506 123.2548138   66.868012   132.6489933  111.9414802
  99.79866988  76.26973887  89.08310201 109.636018    96.71791287
  96.28547323 127.8021082  112.9373963  114.6269398  112.4831253
 128.2809537  110.1142502   81.1547673   60.70238138 117.2177177
  93.16406456  86.24532972  90.70206847  88.00175655 124.0469975
 117.2735726  107.2609915  104.0767926  122.9497116  121.1105841
 143.9519908   87.93739852  99.6011896  104.29978    114.9546663
  86.05930834 101.1799657  110.3341415  112.9557703  107.1117844
 103.262856    99.49648475  82.25351215 102.3134457   76.71751602
  70.32511371 110.0930537  122.1599585   58.30095817 110.2320489
 106.6508842  114.1031517  101.2066501  118.0273944   75.63057702
  99.86145565  95.07563134  83.22122036  83.42149109  47.64149344]
```

team :

```
 ['TeamA' 'TeamC' 'TeamB']

provider :
 ['Provider4' 'Provider1' 'Provider2' 'Provider3']
```

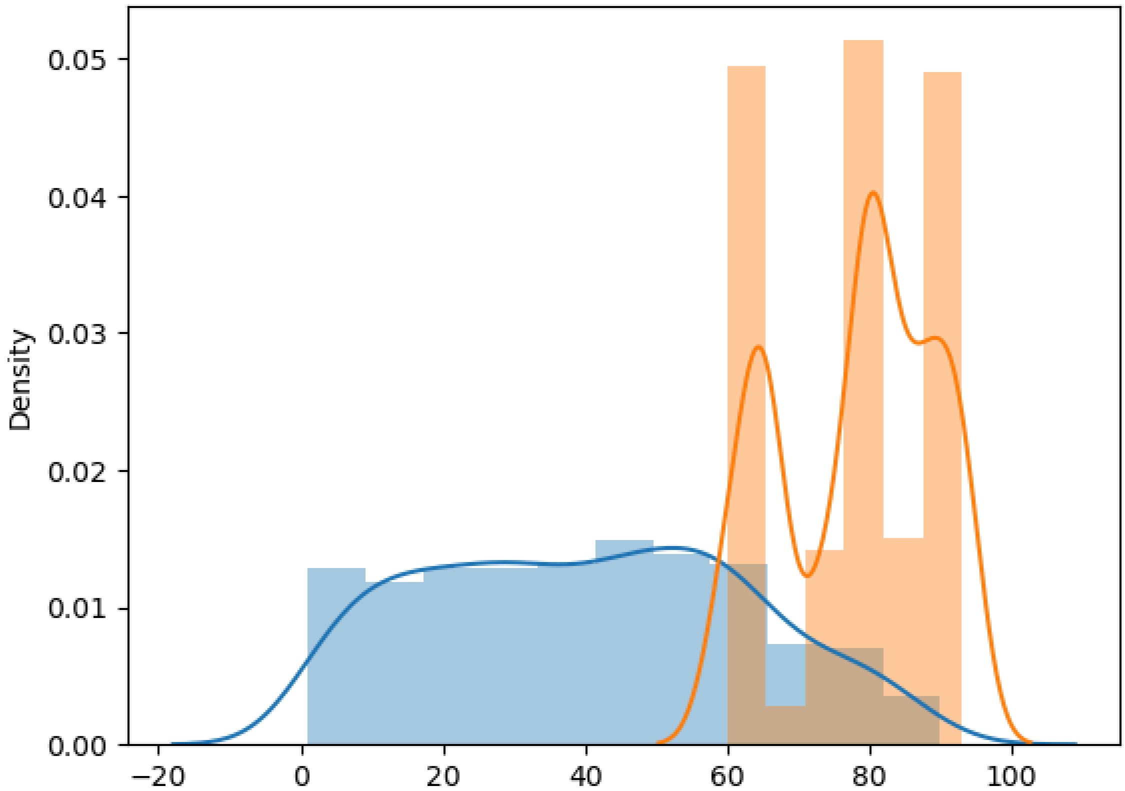
In [42]:
```python
main.describe(include='all')
```

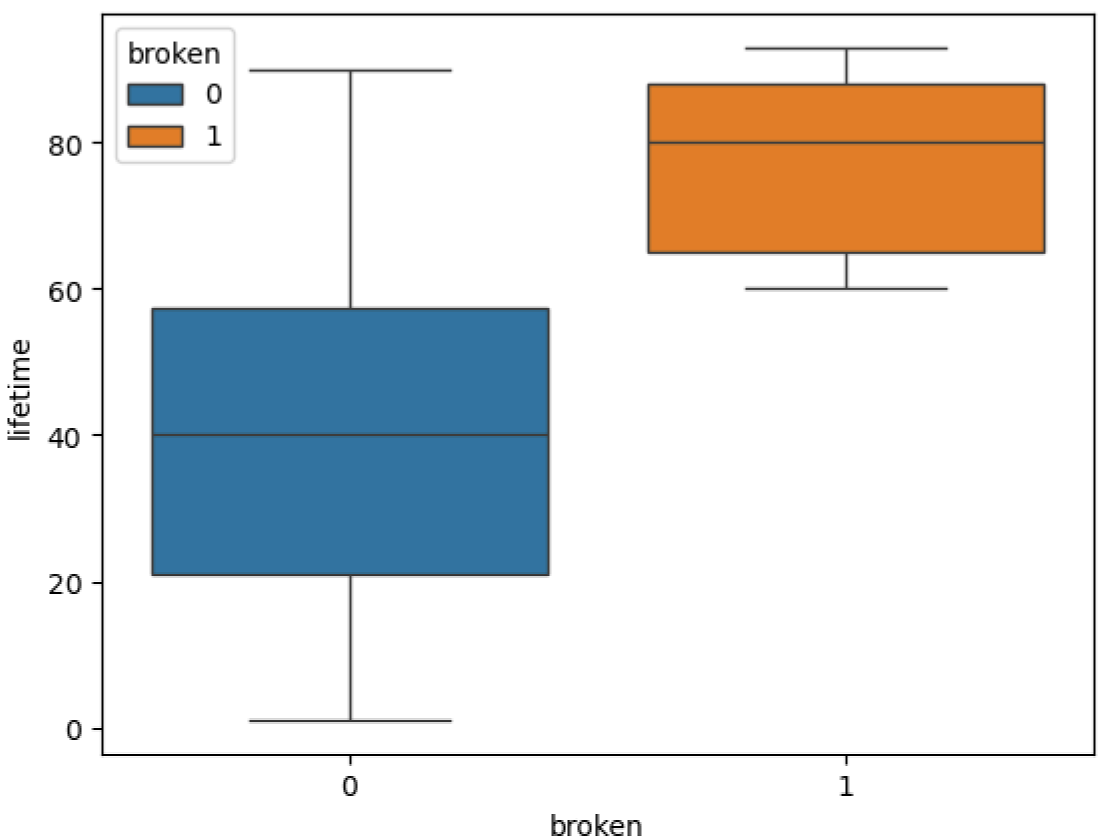
Out[42]:

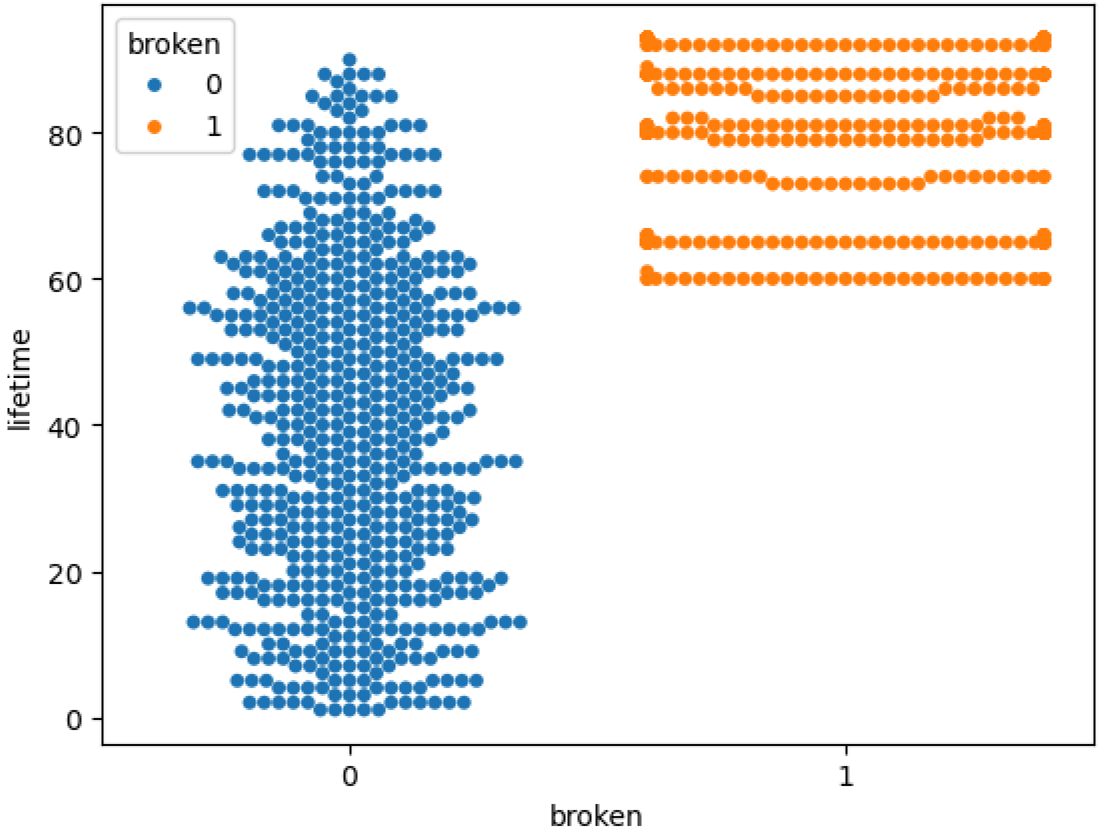
	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	prov
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000	1
unique	NaN	NaN	NaN	NaN	NaN	3	
top	NaN	NaN	NaN	NaN	NaN	TeamB	Provi
freq	NaN	NaN	NaN	NaN	NaN	356	
mean	55.195000	0.397000	98.599338	99.376723	100.628541	NaN	I
std	26.472737	0.489521	19.964052	9.988726	19.633060	NaN	I
min	1.000000	0.000000	33.481917	58.547301	42.279598	NaN	I
25%	34.000000	0.000000	85.558076	92.771764	87.676913	NaN	I
50%	60.000000	0.000000	97.216997	99.433959	100.592277	NaN	I
75%	80.000000	1.000000	112.253190	106.120762	113.662885	NaN	I
max	93.000000	1.000000	173.282541	128.595038	172.544140	NaN	I

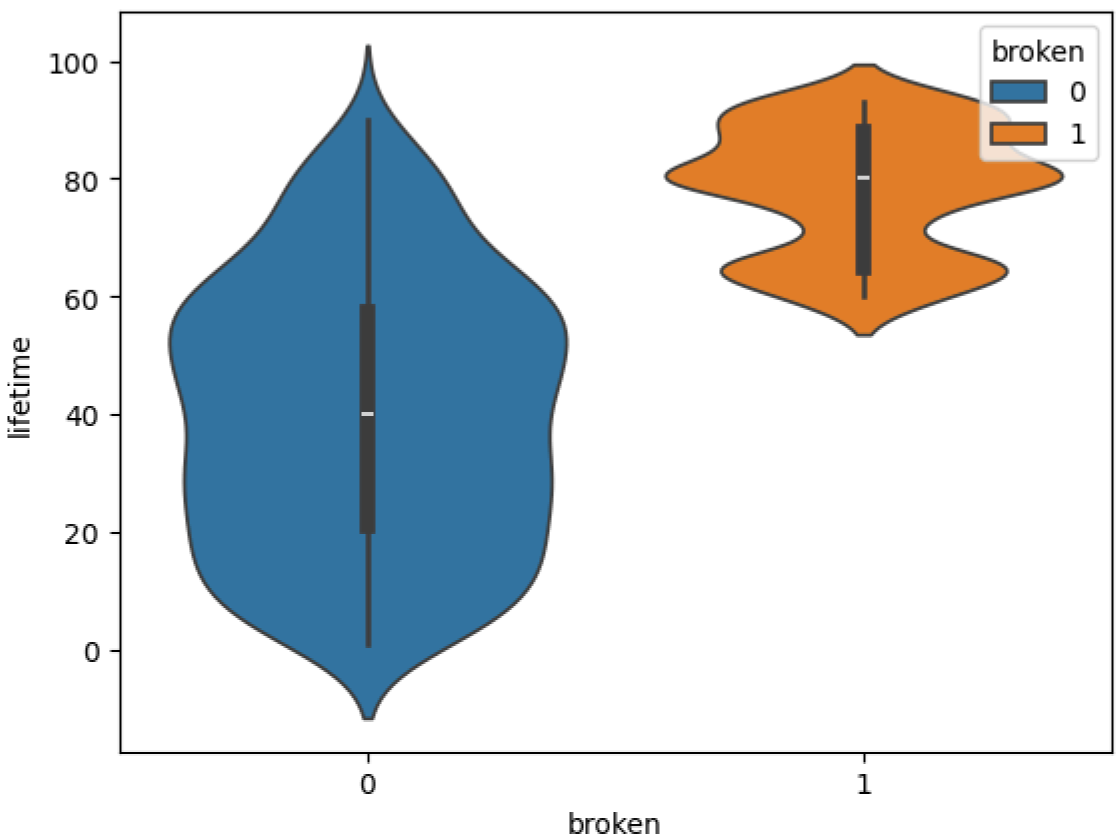
In [43]:
```python
#Data Visualization or EDA
import warnings
warnings.filterwarnings('ignore')
```

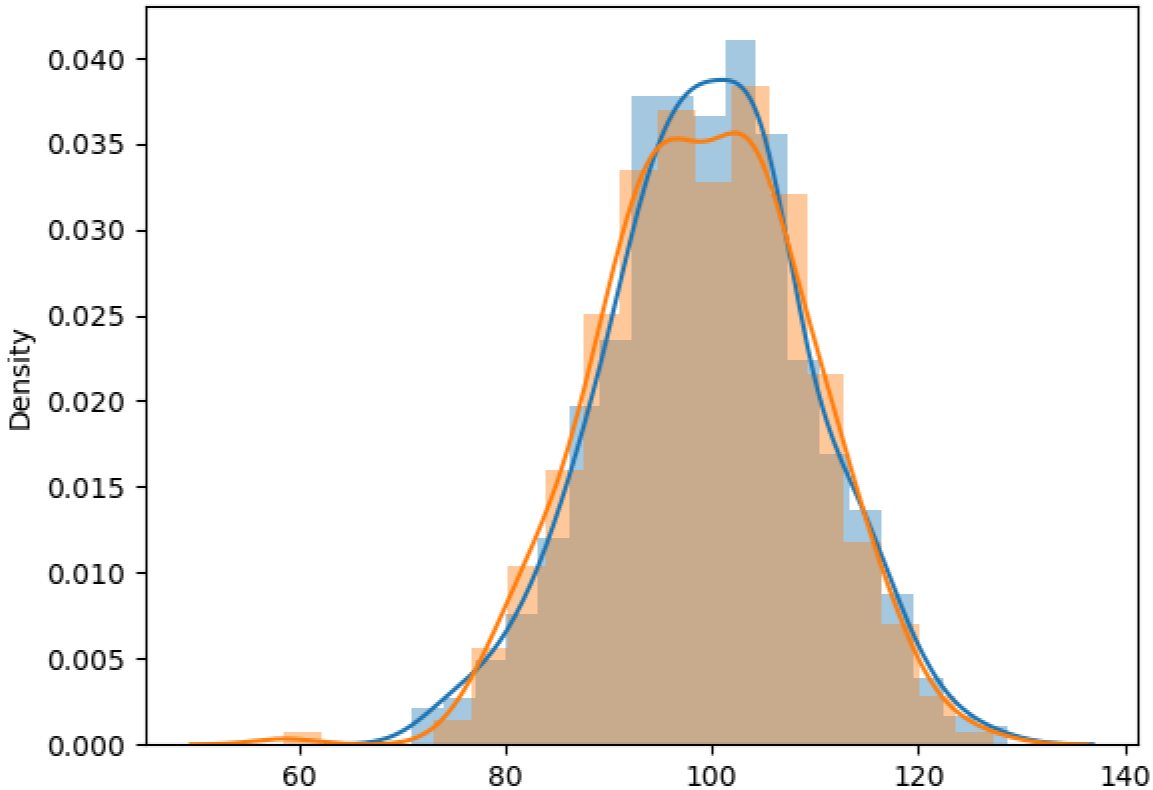
In [44]:
```python
for i in ['lifetime','moistureInd','pressureInd','temperatureInd']:
    #distplot
    sns.distplot(x=main[i][main.broken==0])
    sns.distplot(x=main[i][main.broken==1])
    plt.show()

    #boxplot
    sns.boxplot(x=main.broken,y=main[i],hue=main.broken)
    plt.show()

    #swarmplot
    sns.swarmplot(x=main.broken,y=main[i],hue=main.broken)
    plt.show()

    #violinplot
    sns.violinplot(x=main.broken,y=main[i],hue=main.broken)
    plt.show()
```


Density
0.05
0.04
0.03
0.02
0.01
0.00
−20
0
20
40
60
80
100

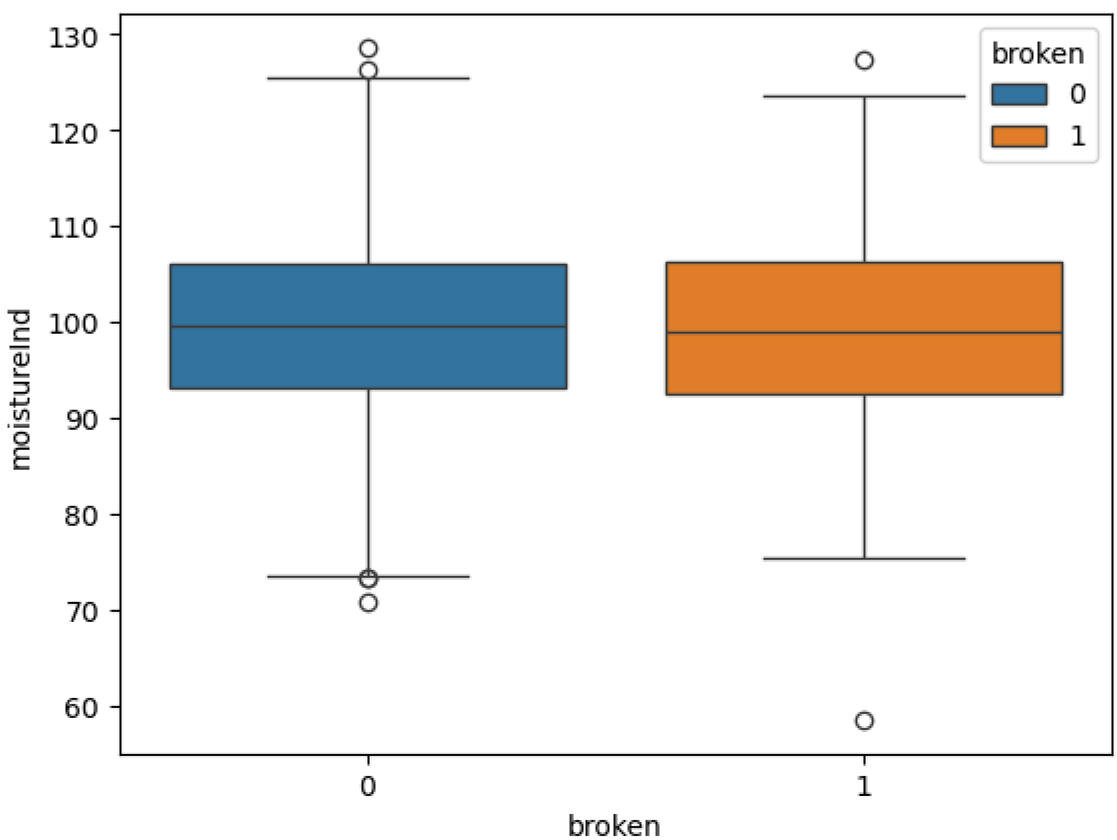

broken
0
1
lifetime
80
60
40
20
0
0
1
broken


broken
0
1
lifetime
0
20
40
60
80
0
1
broken

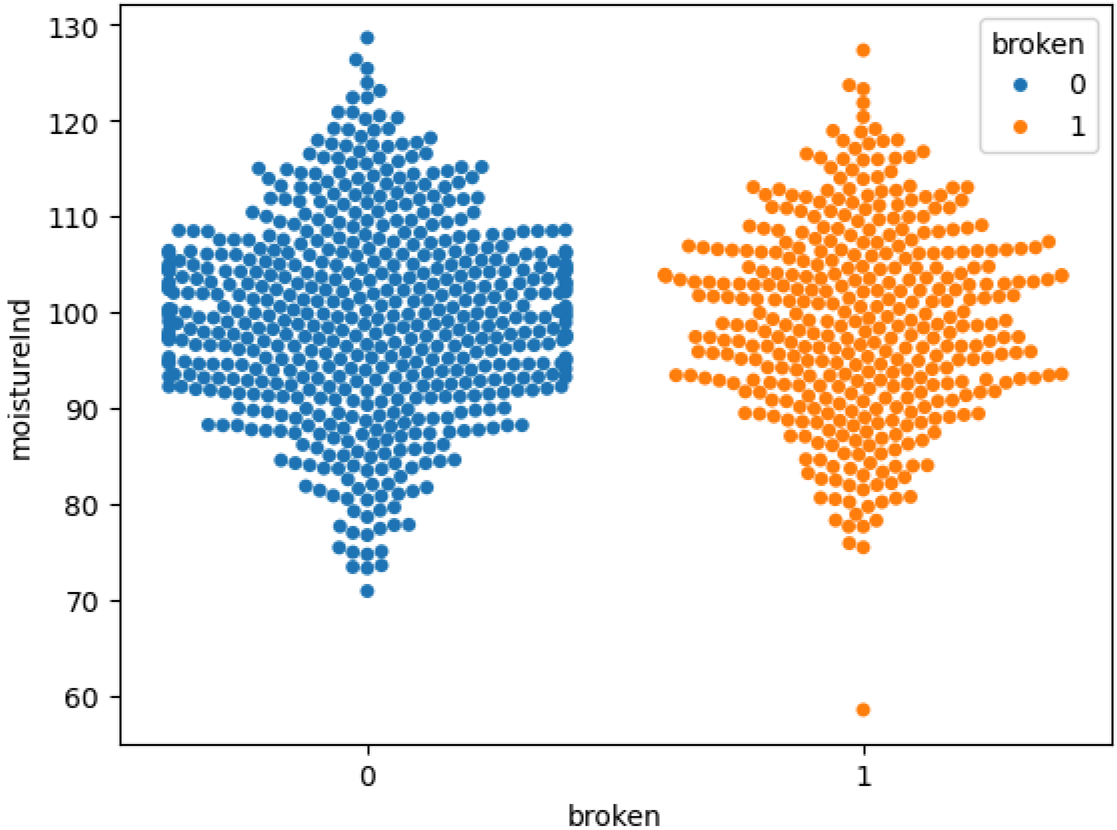

broken
0
1
lifetime
0
20
40
60
80
100
0
1
broken

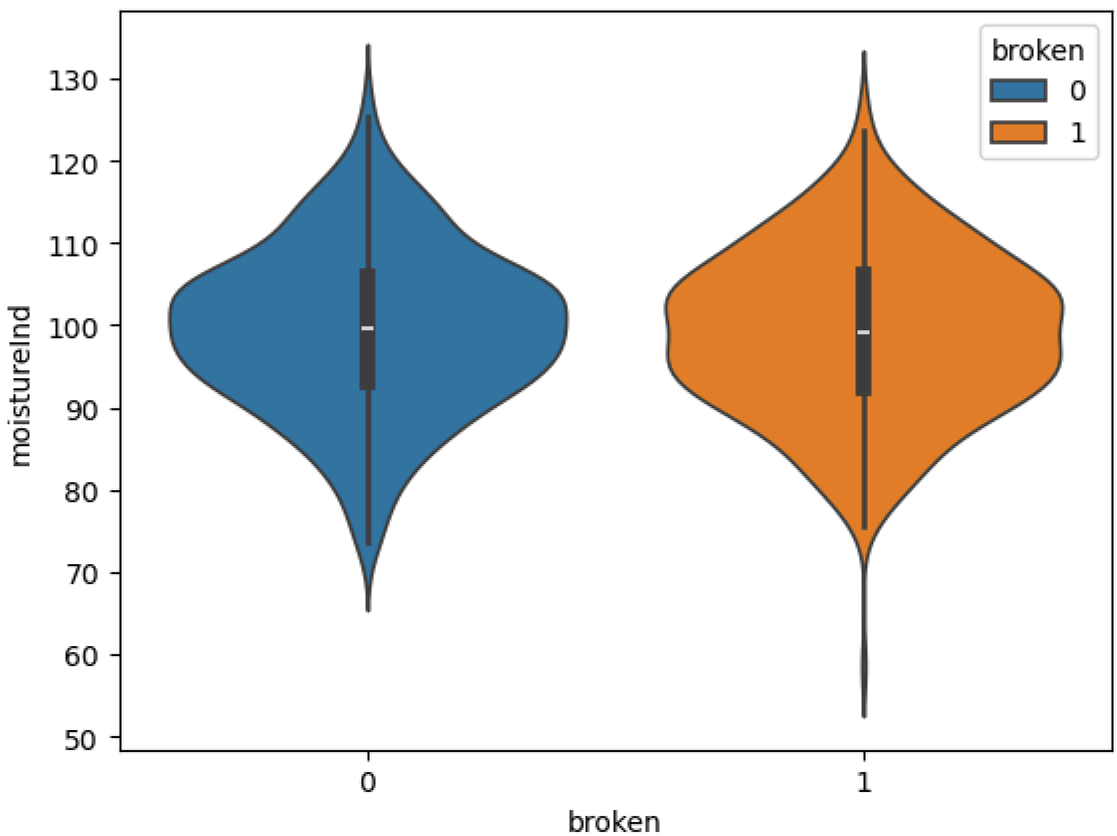


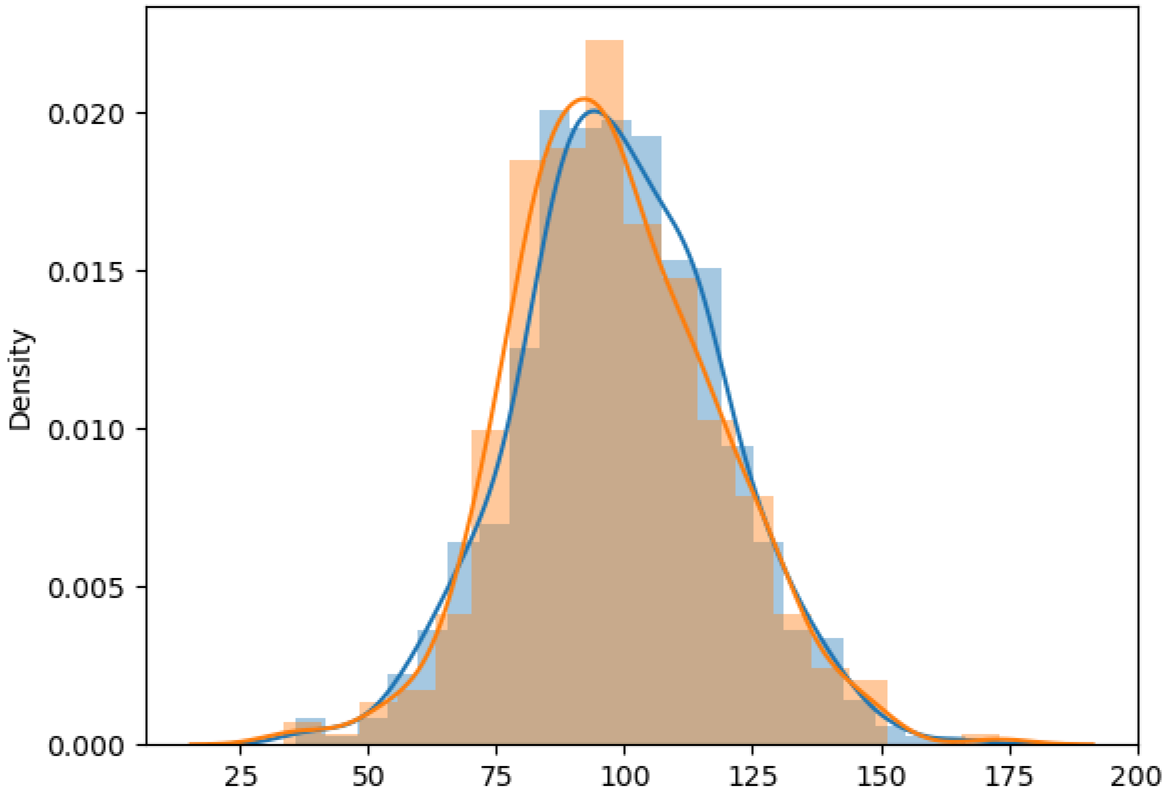
Density
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
60
80
100
120
140

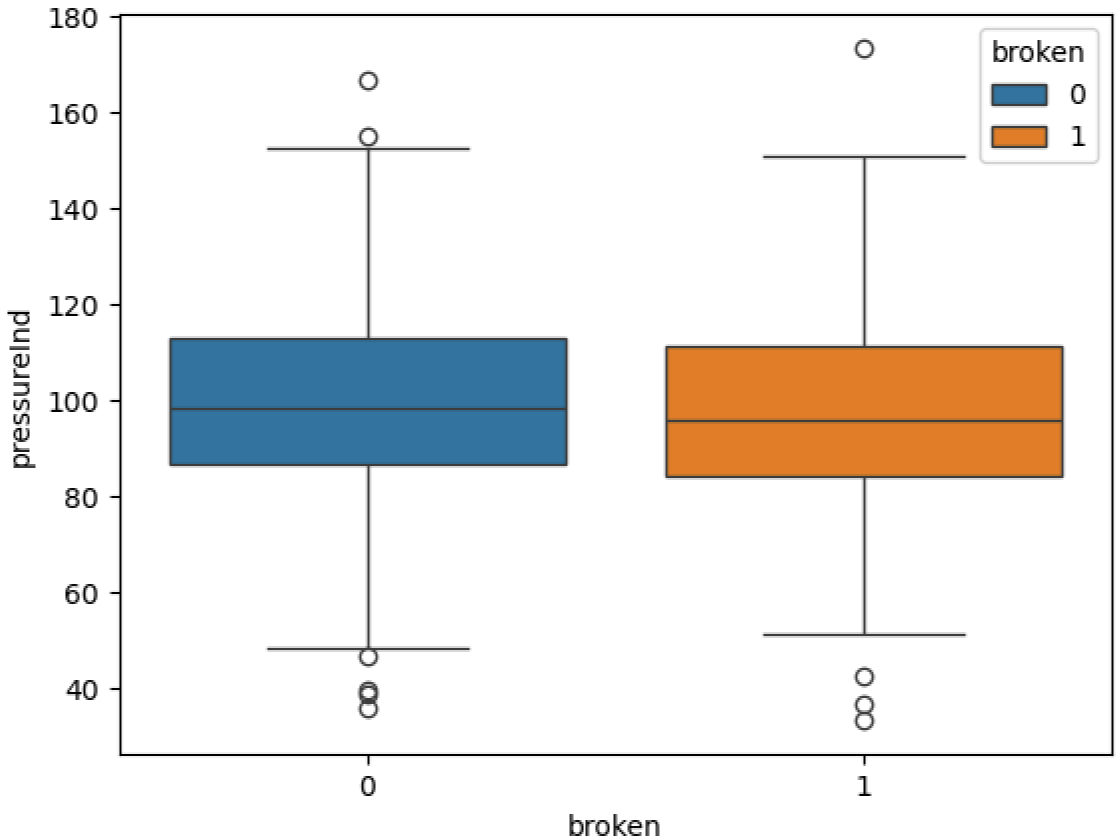


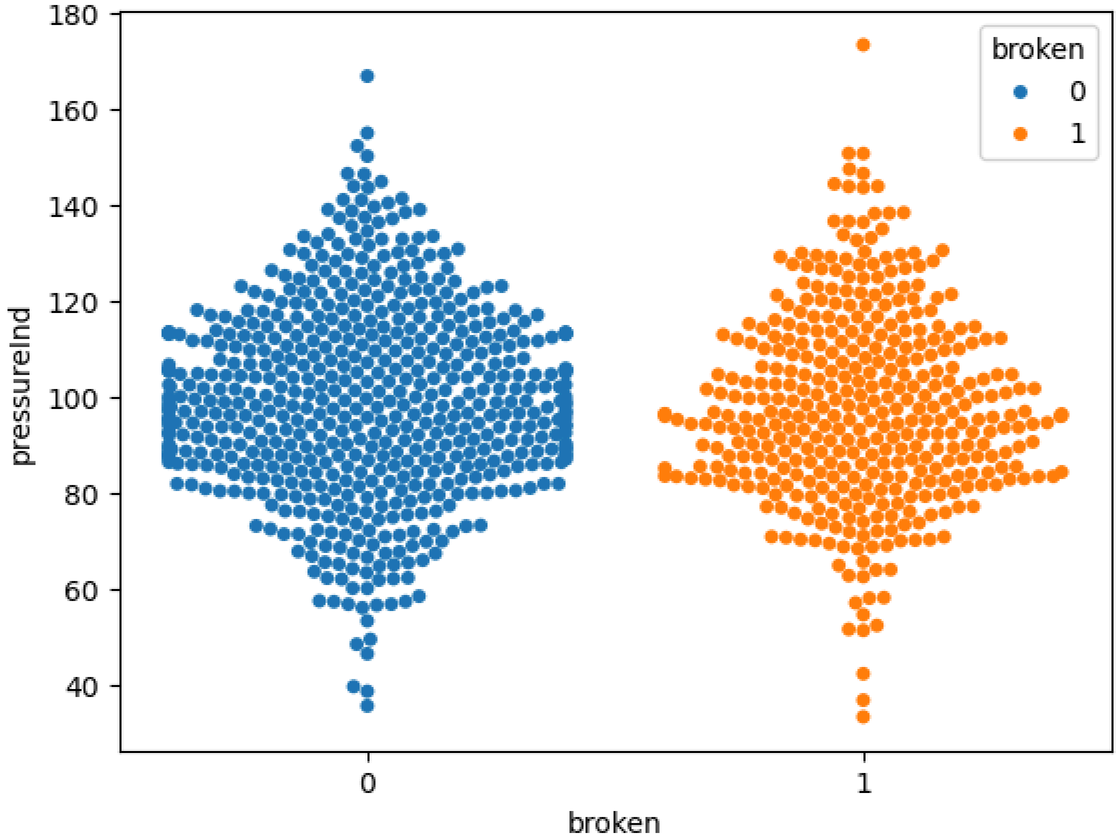
broken
0
1
moistureInd
130
120
110
100
90
80
70
60
0
1
broken

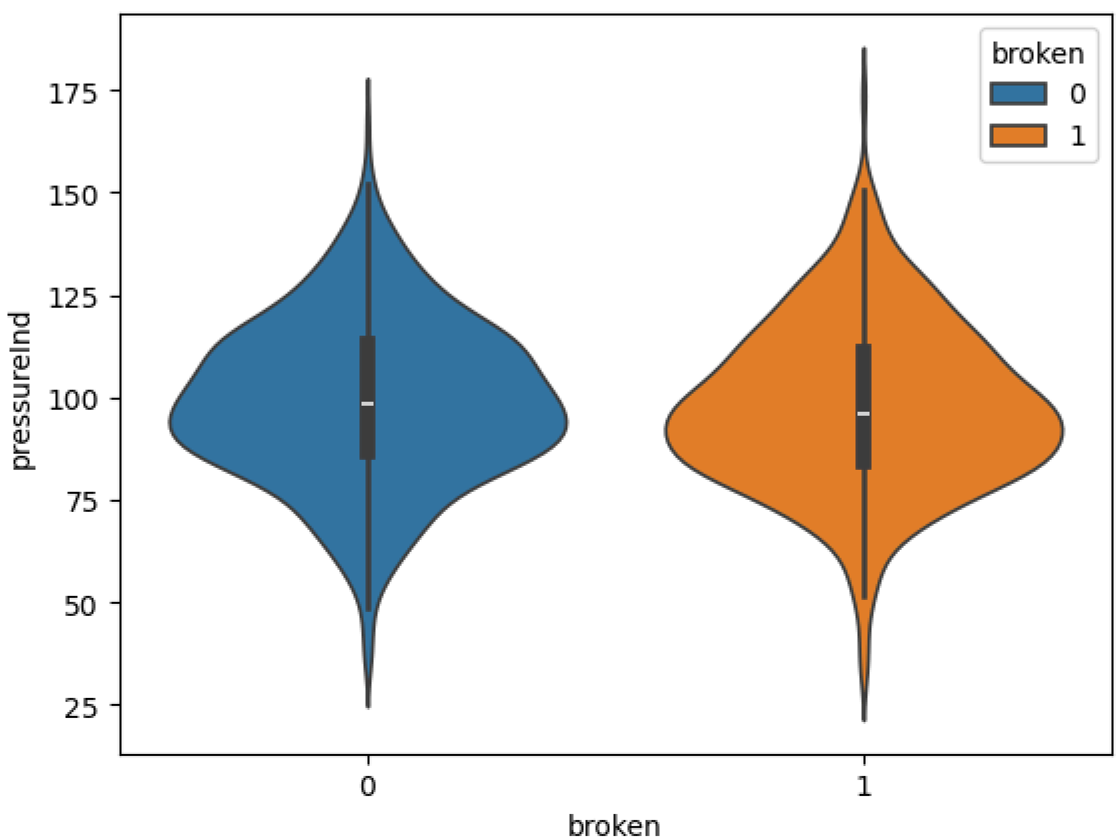

broken
0
1
moistureInd
130
120
110
100
90
80
70
60
0
1
broken

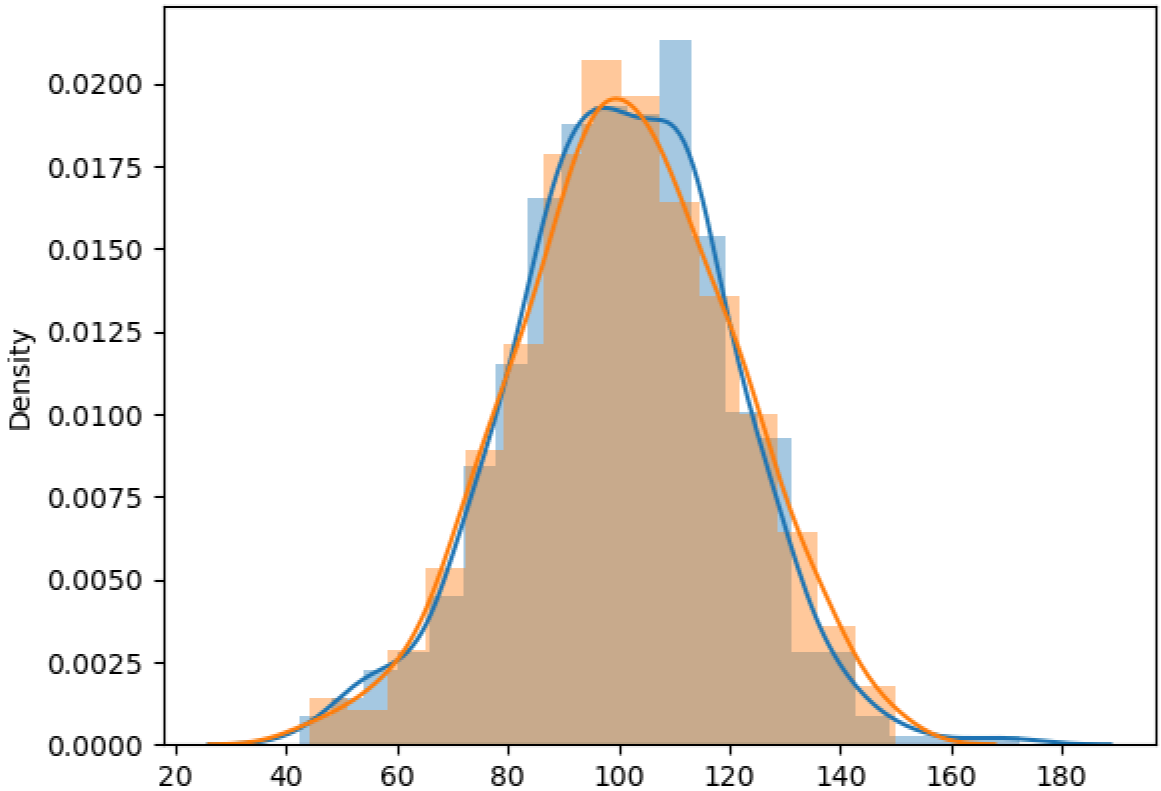

broken
0
1
moistureInd
130
120
110
100
90
80
70
60
50
0
1
broken

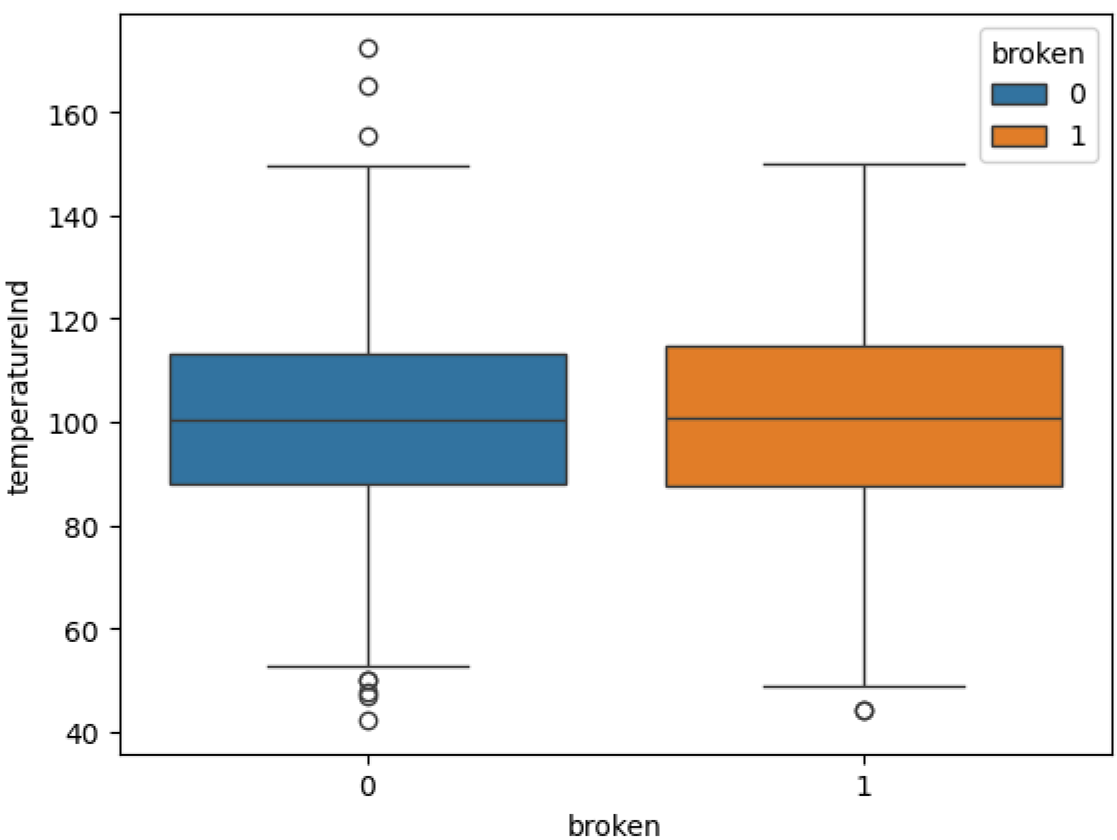

Density
0.000
0.005
0.010
0.015
0.020
25
50
75
100
125
150
175
200

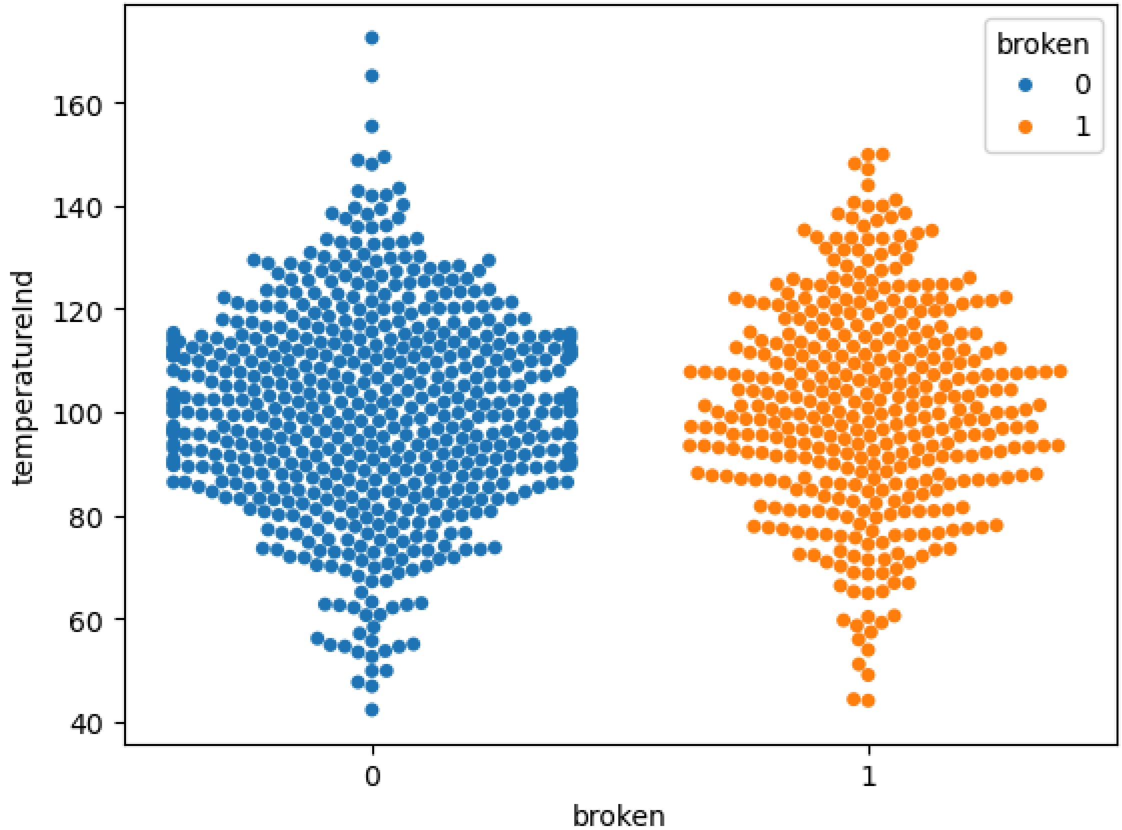

pressureInd
broken
0
1
40
60
80
100
120
140
160
180

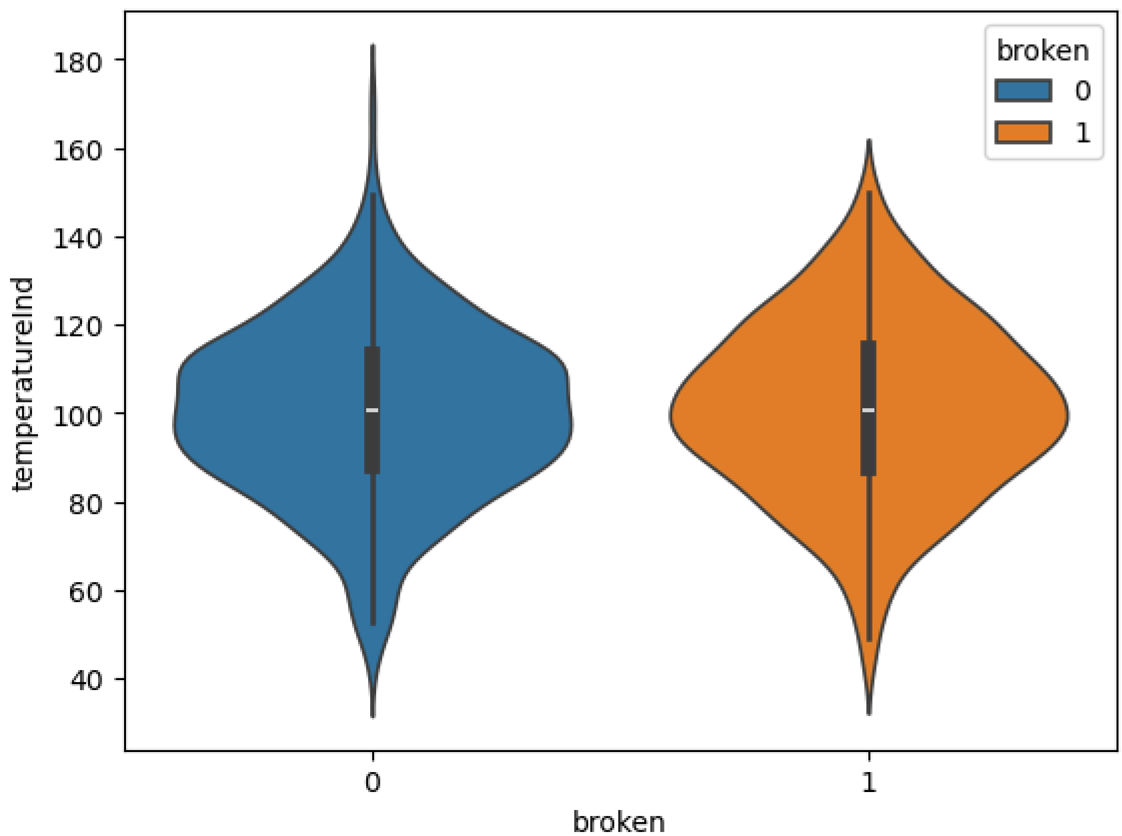

broken
0
1
pressureInd
broken
0
1
180
160
140
120
100
80
60
40


broken
0
1
pressureInd
broken
0
1
175
150
125
100
75
50
25

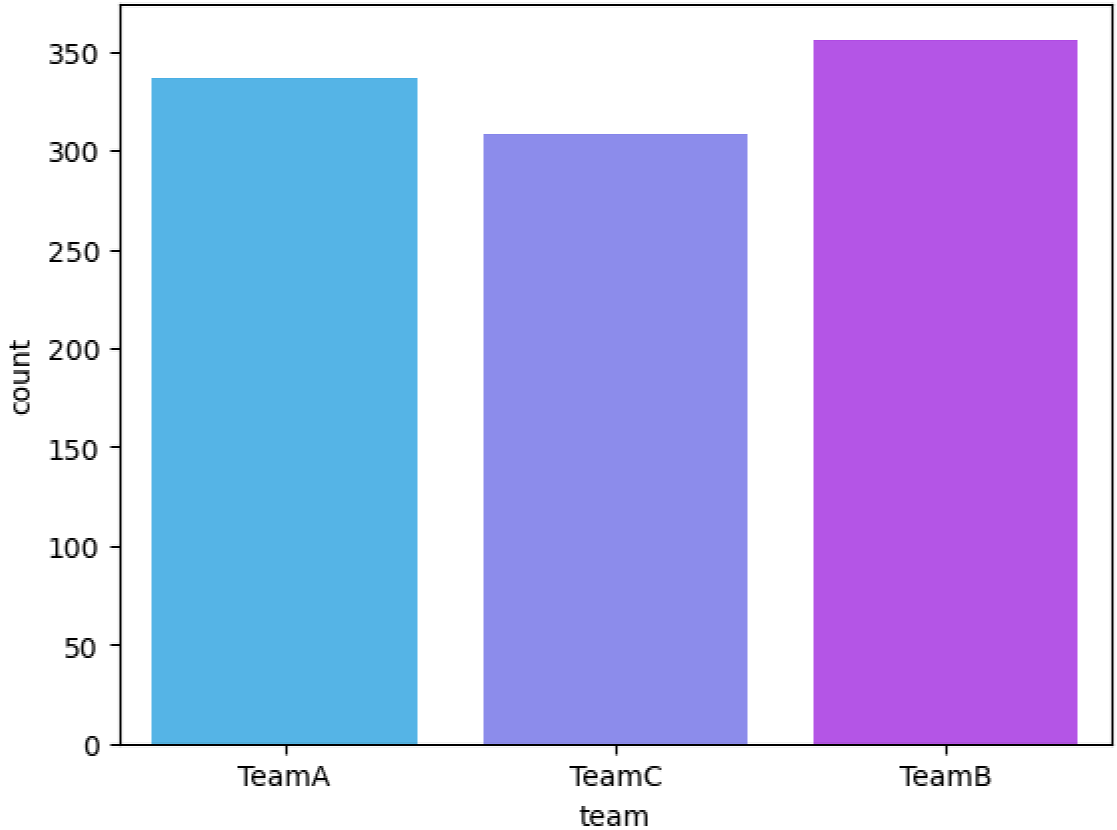

0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Density
20
40
60
80
100
120
140
160
180


broken
0
1
160
140
120
100
80
60
40
temperatureInd
0
1
broken

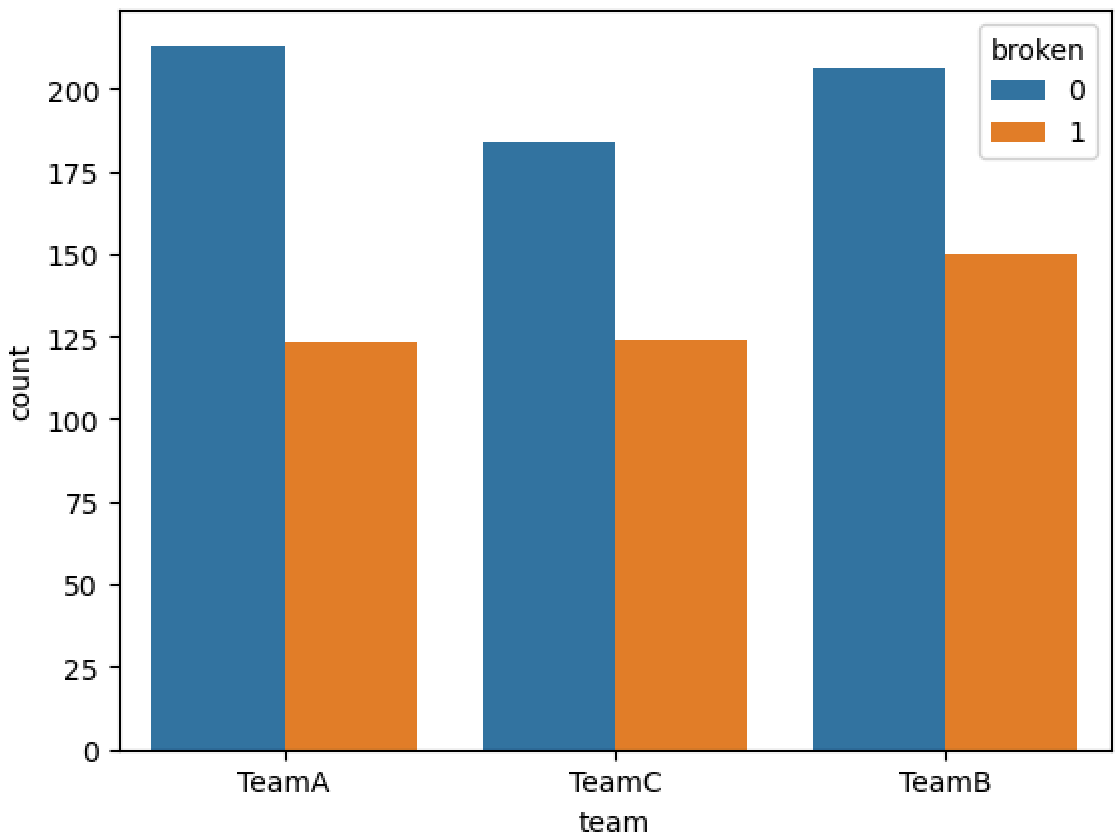




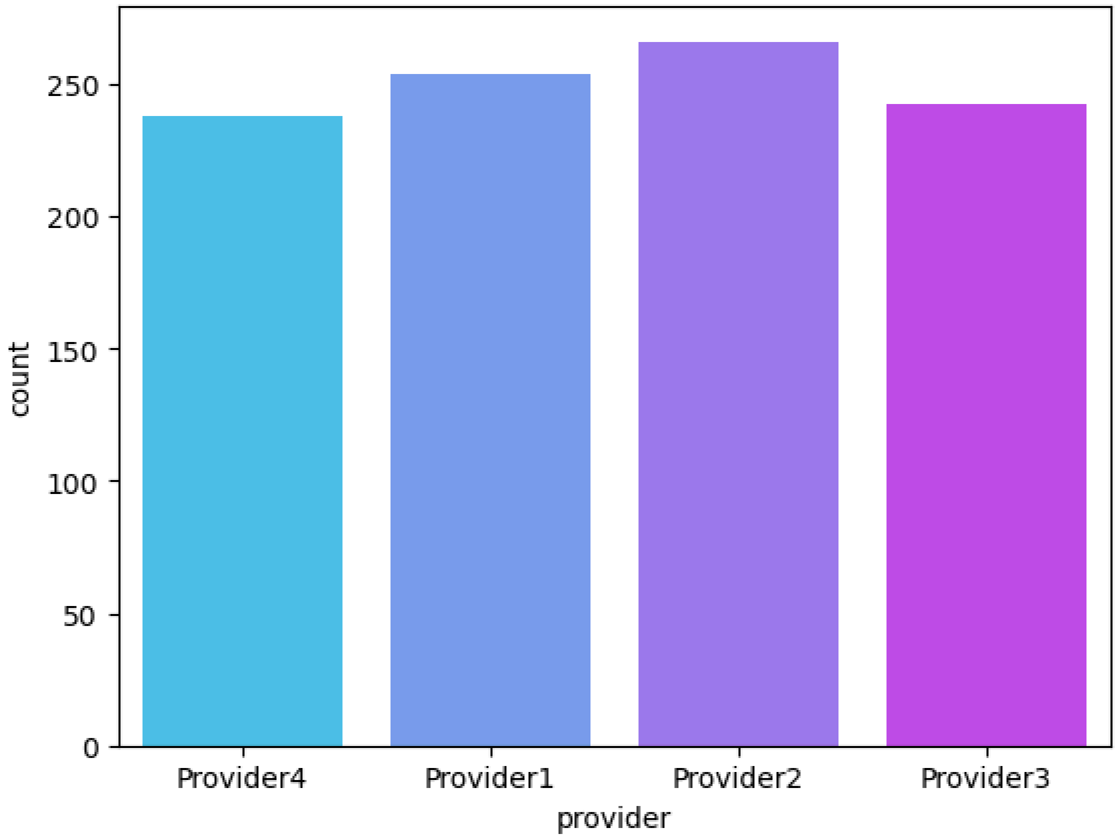
In [45]:
```
sns.countplot(x=main.team,palette='cool')
plt.show()
```



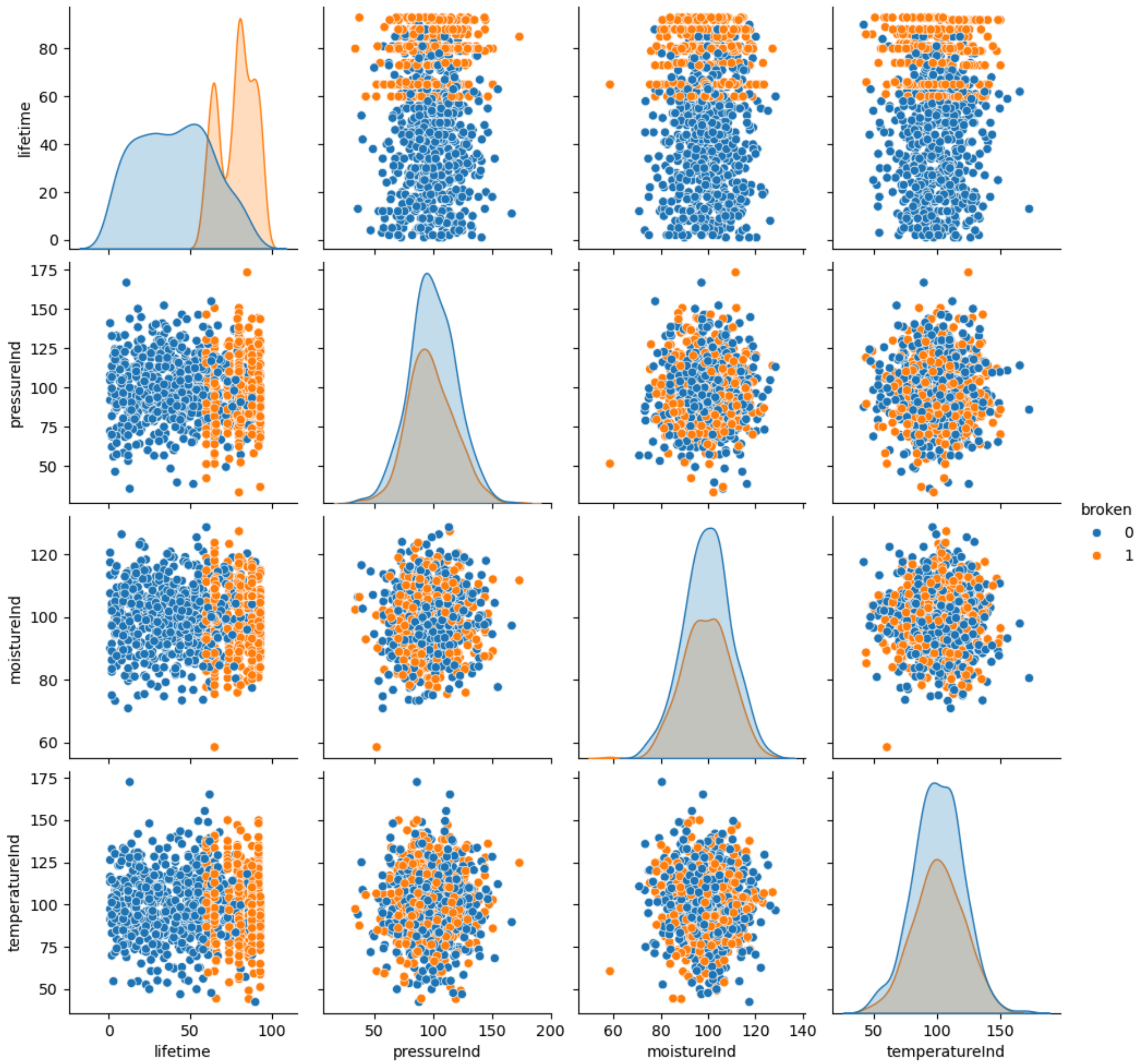
In [46]:
```
sns.countplot(x=main.team,hue=main.broken)
plt.show()
```



In [47]:
```
sns.countplot(x=main.provider,palette='cool')
plt.show()
```



In [48]:
```
sns.pairplot(main,hue='broken')
plt.show()
```



In [49]:
```
# Encoding:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [50]:
```
# changing the team's column:
main.team = le.fit_transform(main.team)
```

In [51]:
```
main.team.unique()
```

Out[51]:
```
array([0, 2, 1])
```

In [52]:
```
le.inverse_transform([0, 2, 1])
```

Out[52]:
```
array(['TeamA', 'TeamC', 'TeamB'], dtype=object)
```

In [53]:
```
main.provider = le.fit_transform(main.provider)
main.provider.unique()
```

Out[53]:
```
array([3, 0, 1, 2])
```

In [54]:
```
le.inverse_transform([3, 0, 1, 2])
```

Out[54]:
```
array(['Provider4', 'Provider1', 'Provider2', 'Provider3'], dtype=object)
```

In [55]:
```
main
```

Out[55]:

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider
0	56	0	92.178854	104.230204	96.517159	0	3
1	81	1	72.075938	103.065701	87.271062	2	3
2	60	0	96.272254	77.801376	112.196170	0	0
3	86	1	94.406461	108.493608	72.025374	2	1
4	34	0	97.752899	99.413492	103.756271	1	0
...	...	...	...	...	...	...	...
995	88	1	88.589759	112.167556	99.861456	1	3
996	88	1	116.727075	110.871332	95.075631	0	3
997	22	0	104.026778	88.212873	83.221220	1	0
998	78	0	104.911649	104.257296	83.421491	0	3
999	63	0	116.901354	99.998694	47.641493	1	0

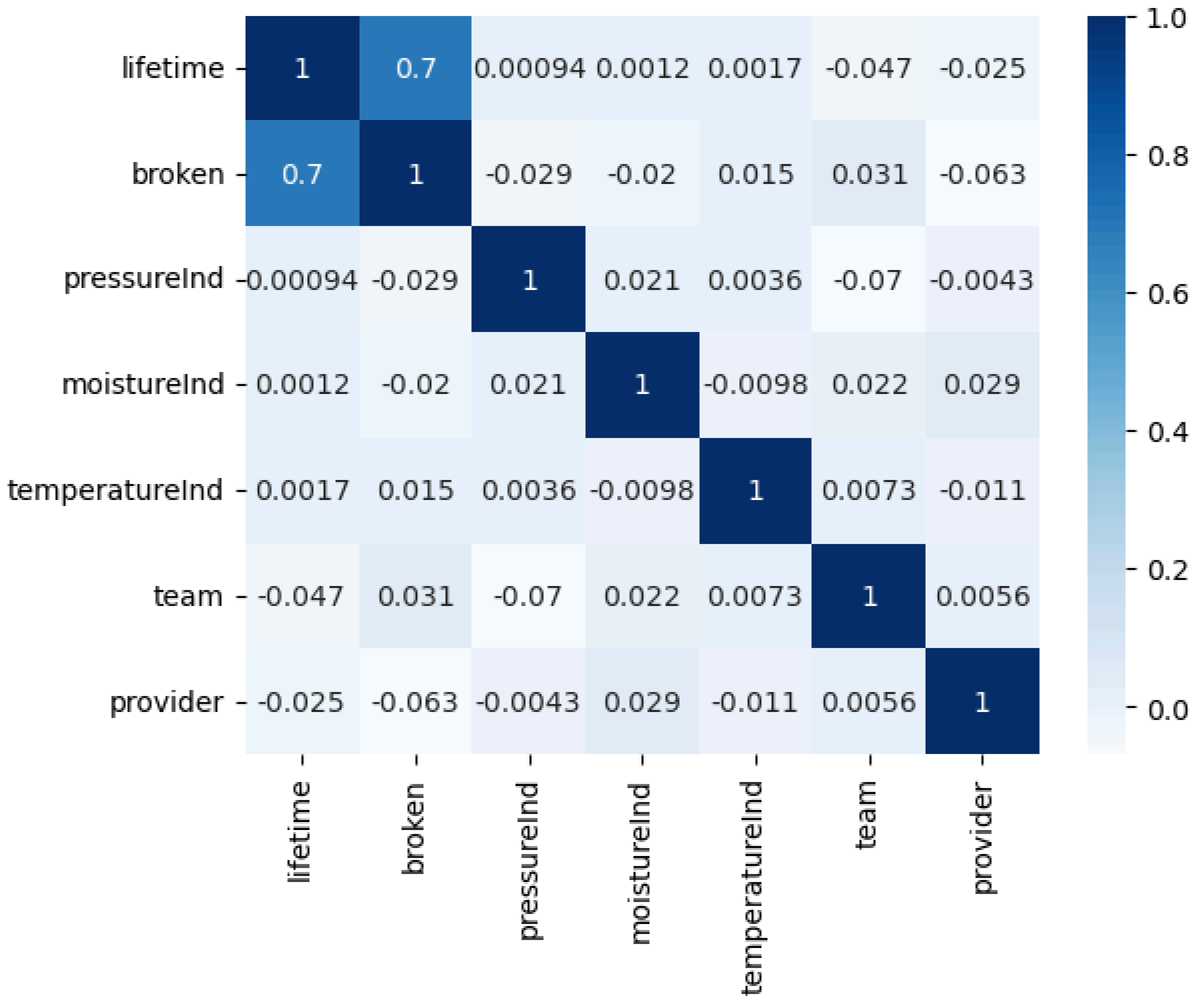
1000 rows × 7 columns

In [56]:
```
# finding the correlation
cr = main.corr()
cr
```

Out[56]:

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team
lifetime	1.000000	0.702656	0.000943	0.001196	0.001744	-0.046537
broken	0.702656	1.000000	-0.028942	-0.019520	0.015364	0.030876
pressureInd	0.000943	-0.028942	1.000000	0.020543	0.003641	-0.069528
moistureInd	0.001196	-0.019520	0.020543	1.000000	-0.009842	0.022420
temperatureInd	0.001744	0.015364	0.003641	-0.009842	1.000000	0.007310
team	-0.046537	0.030876	-0.069528	0.022420	0.007310	1.000000
provider	-0.025172	-0.062972	-0.004337	0.028906	-0.010822	0.005606

In [57]:
```
sns.heatmap(cr,annot=True,cmap='Blues')
plt.show()
```



In [58]:
```
# Creation of ip/op:-
ip = main.drop('broken',axis=1)
```

In [59]:
```
ip.head()
```

Out[59]:

	lifetime	pressureInd	moistureInd	temperatureInd	team	provider
0	56	92.178854	104.230204	96.517159	0	3
1	81	72.075938	103.065701	87.271062	2	3
2	60	96.272254	77.801376	112.196170	0	0
3	86	94.406461	108.493608	72.025374	2	1
4	34	97.752899	99.413492	103.756271	1	0

In [60]:
```
op = main.broken
op.head()
```

Out[60]:
```
0    0
1    1
2    0
3    1
4    0
Name: broken, dtype: int64
```

In [61]:
```
# Train Test Split:
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(ip,op,train_size=0.8)
```

In [62]:
```
xtrain.head()
```

Out[62]:

	lifetime	pressureInd	moistureInd	temperatureInd	team	provider
755	58	111.645399	87.898850	79.455241	2	3
677	80	136.581015	92.822475	87.678519	1	0
368	33	111.737241	96.470432	85.473700	1	1
492	55	146.333342	98.237506	103.209237	1	0
333	45	146.482610	98.848252	106.979033	2	3

In [63]:
```
xtest.head()
```

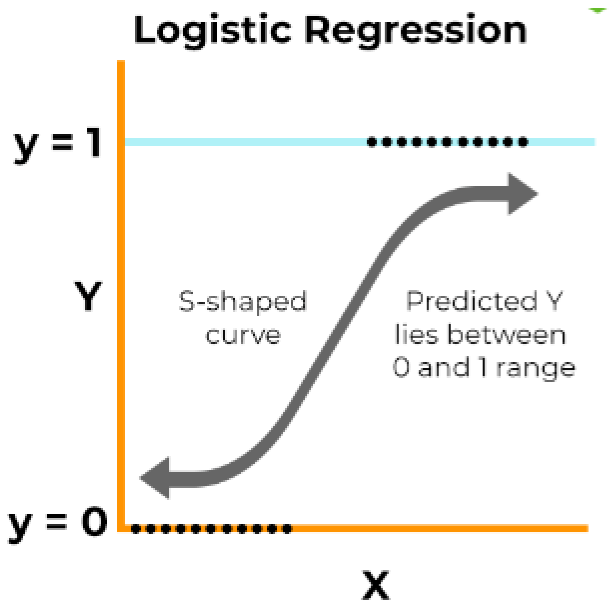
Out[63]:

	lifetime	pressureInd	moistureInd	temperatureInd	team	provider
285	65	112.585712	101.614264	80.835685	1	2
671	31	114.023531	111.605584	108.894301	0	3
24	26	118.978697	105.298916	115.247085	0	1
562	85	109.626240	99.704141	112.418278	2	1
932	9	90.785683	94.055371	84.737974	2	0

In [64]:
```
# Standardizing the data:-
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [65]:
```
xtrain = sc.fit_transform(xtrain)
xtest = sc.fit_transform(xtest)
```

Logistic Regression :- uses a logistic function called a sigmoid function for predictions. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1.



Uses Sigmoid Function Formula:-

$$f(x) = \frac{1}{1 + e^{-x}}$$

In [66]:
```
# Applying ML Algorithm:-
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

In [67]:
```
lr.fit(xtrain,ytrain)
```

Out[67]:
```
▾ LogisticRegression
LogisticRegression()
```

In [68]:
```
# Prediction:-
ypred = lr.predict(xtest)
ypred
```

Out[68]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1], dtype=int64)
```

Accuracy:-

In a classification model accuracy is found out by using Confusion Matrix

Accuracy:- (TN + TP)/All values

Recall:- (TP)/(FN+TP)

In [69]:
```python
from sklearn.metrics import recall_score,accuracy_score
acc = accuracy_score(ypred,ytest)
rec = recall_score(ypred,ytest)
```

In [70]:
```python
print(f"Accuracy:",acc)
print(f"Recall:",rec)
```

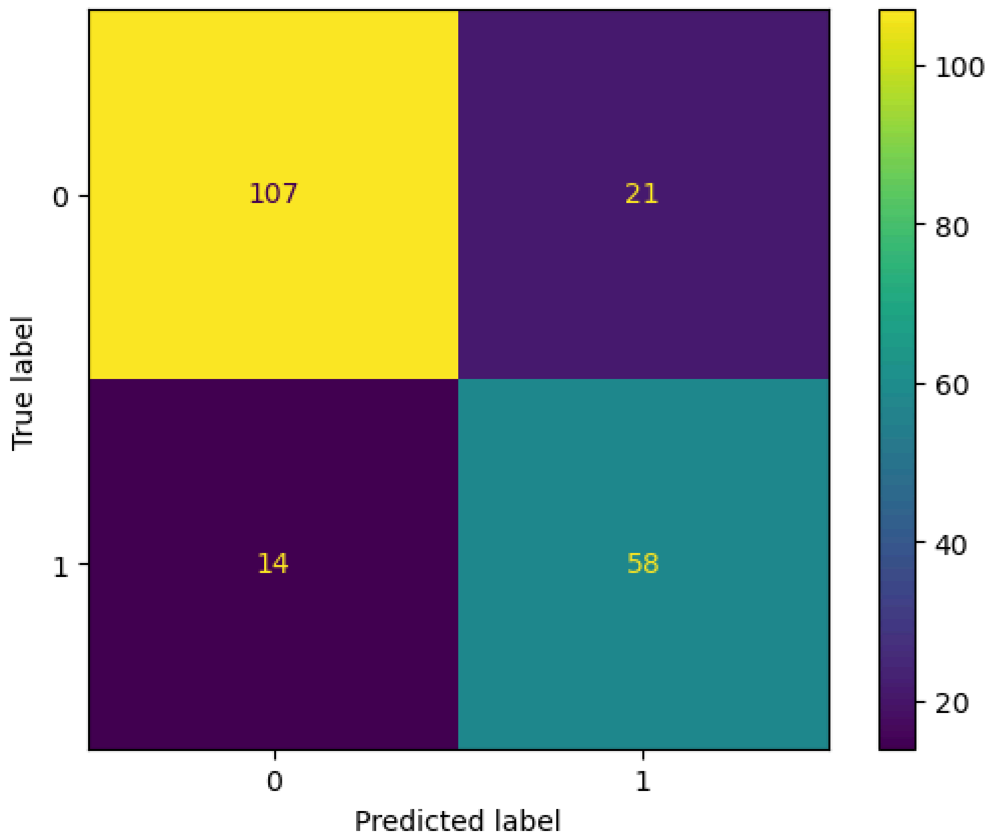
```
Accuracy: 0.825
Recall: 0.8055555555555556
```

In [71]:
```python
# Confusion matrix:-
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix
cm = confusion_matrix(ypred,ytest)
```

In [72]:
```python
cm
```

Out[72]:
```
array([[107,  21],
       [ 14,  58]], dtype=int64)
```

In [73]:
```python
cmd = ConfusionMatrixDisplay(cm)
cmd.plot()
plt.show()
```



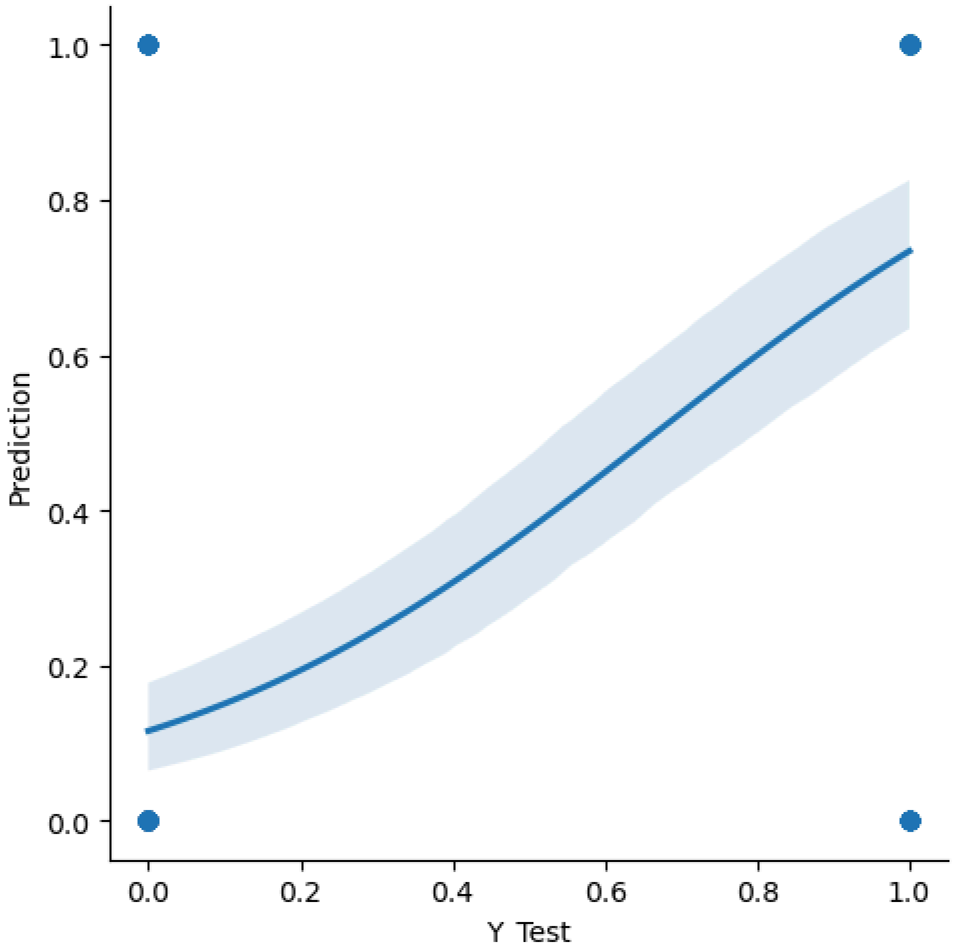
In [74]:
```
df1 = pd.DataFrame({'Y_Test':list(ytest),
                    'Prediction':list(ypred)})
df1
```

Out[74]:

	Y_Test	Prediction
0	1	0
1	0	0
2	0	0
3	0	1
4	0	0
...	...	...
195	0	0
196	1	1
197	0	0
198	0	0
199	1	1

200 rows × 2 columns

In [75]:
```
# to plot the best fit line:
sns.lmplot(x='Y_Test',y='Prediction',data=df1,logistic=True)
plt.show()
```



KNN:- (K-Nearest Neighbor):- based on the principle of proximity, where data points are classified or predicted based on the majority vote of their nearest neighbors.

In [76]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
```

In [77]:
```
knn.fit(xtrain,ytrain)
```

Out[77]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [78]:
```
pred1 = knn.predict(xtest)
```

In [79]:
```
from sklearn.metrics import recall_score,accuracy_score
acc = accuracy_score(pred1,ytest)
rec = recall_score(pred1,ytest)
print(f"Accuracy:",acc)
print(f"Recall:",rec)
```

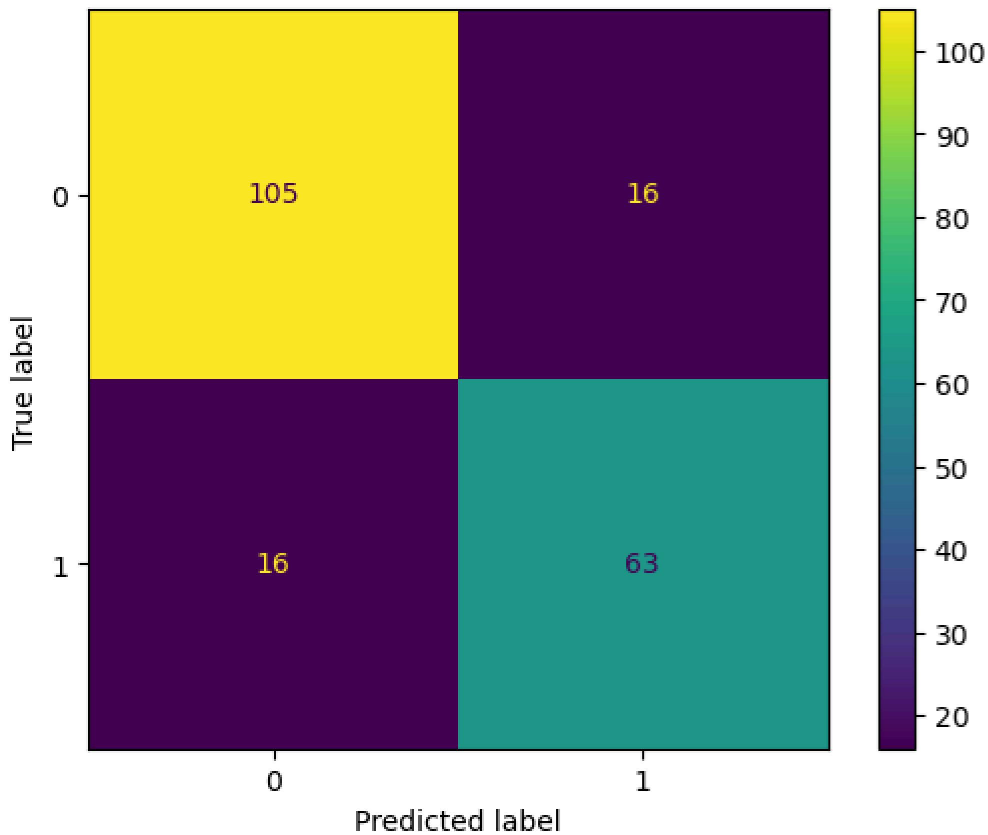
```
Accuracy: 0.84
Recall: 0.7974683544303798
```

In [80]:
```
# Confusion matrix:-
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix
cm1 = confusion_matrix(pred1,ytest)
```

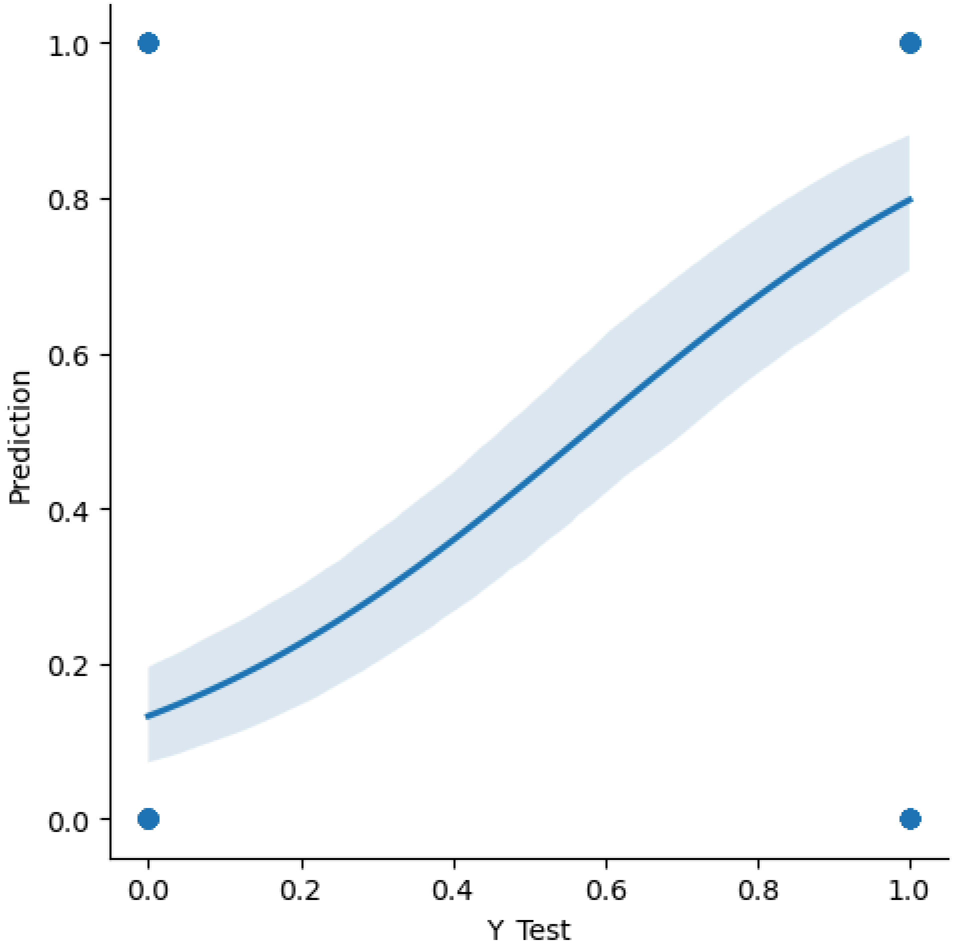
In [81]:
```
cm1
```

Out[81]:
```
array([[105,  16],
       [ 16,  63]], dtype=int64)
```

In [82]:
```
cmd = ConfusionMatrixDisplay(cm1)
cmd.plot()
plt.show()
```



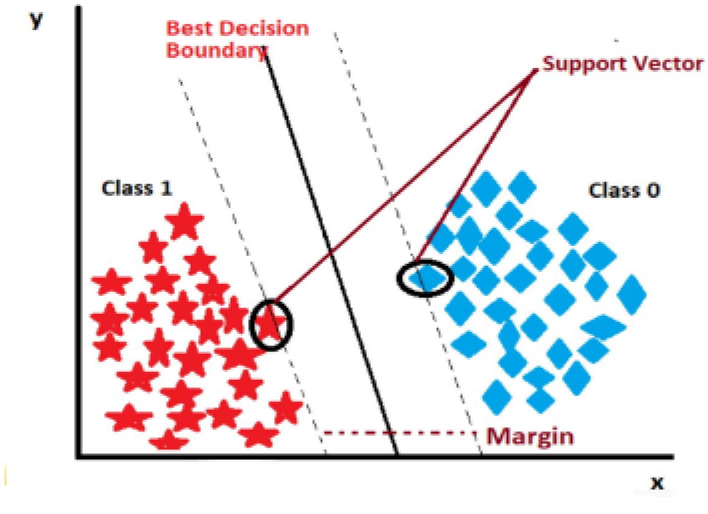
In [83]:
```
df2 = pd.DataFrame({'Y_Test':list(ytest),
                    'Prediction':list(pred1)})
sns.lmplot(x='Y_Test',y='Prediction',data=df2,logistic=True)
plt.show()
```



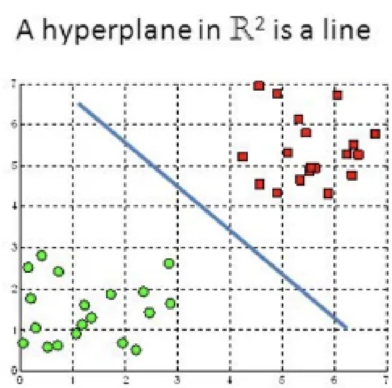
SVM:- Support Vector Machine:-

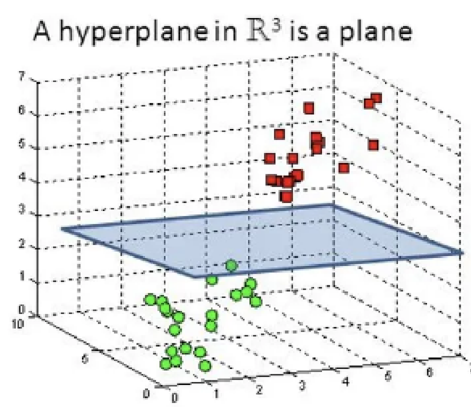
SVC :- Support Vector Classifier :- to find the best fit hyperplane/line thats separates the categories in the best way.

```
 A Hyperplane is a decision boundary that helps classifying data
points.
```



Categorizing the data





Kernel Function:-
1. linear -> 1D to 2D
2. Poly -> 2D to 3D
3. Sigmoid -> 2D to 3D
4. rbf (Radial Basis Function) -> 1D - 2D - 3D

gamma -> coefficient of the kernel function.

C :- Regularization Parameter (this restrict our model to be overfitted.)
Range of C:-
i. (0.01 - 1) - Small (noisy data, wants to avoid underfitting)

ii. (1 - 10) - Medium (good training starting point)
iii. (10 - 1000) - High (Data is clean, helps to minimize overfitting)

In [84]:
```
from sklearn.svm import SVC
sv = SVC(kernel = 'rbf', gamma = 0.01, C = 1000 )
sv.fit(xtrain,ytrain)
```

Out[84]:
```
SVC
SVC(C=1000, gamma=0.01)
```

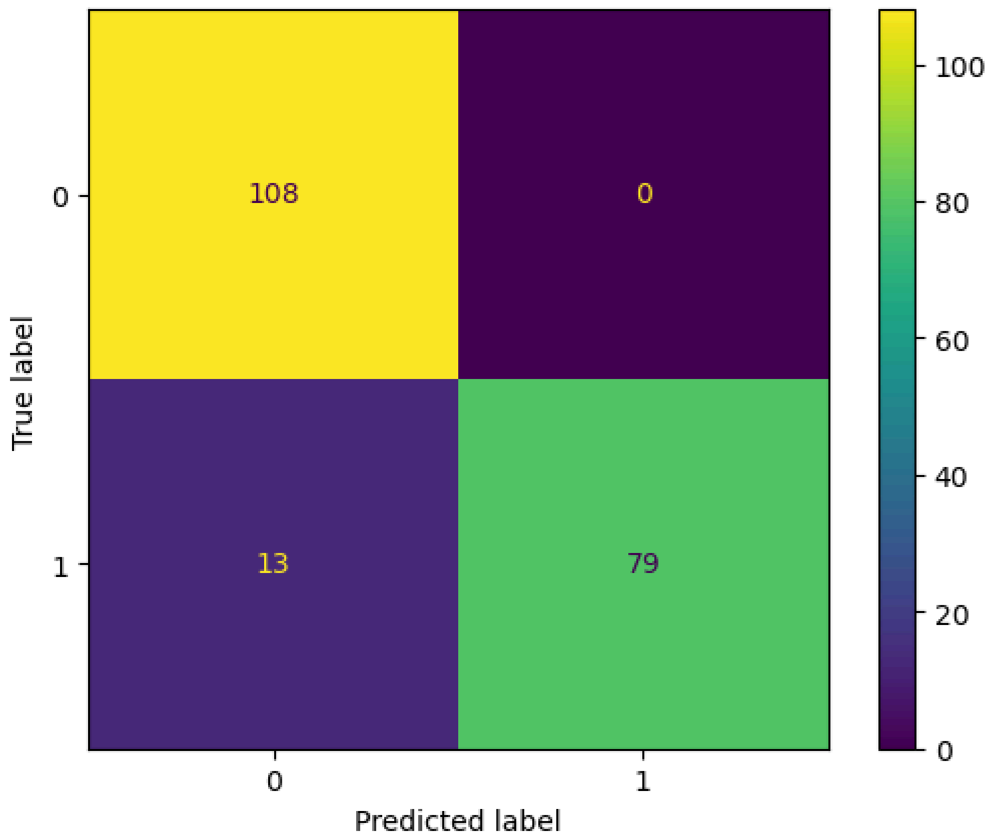
In [85]:
```
pred2 = sv.predict(xtest)
pred2
```

Out[85]:
```
array([1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1], dtype=int64)
```

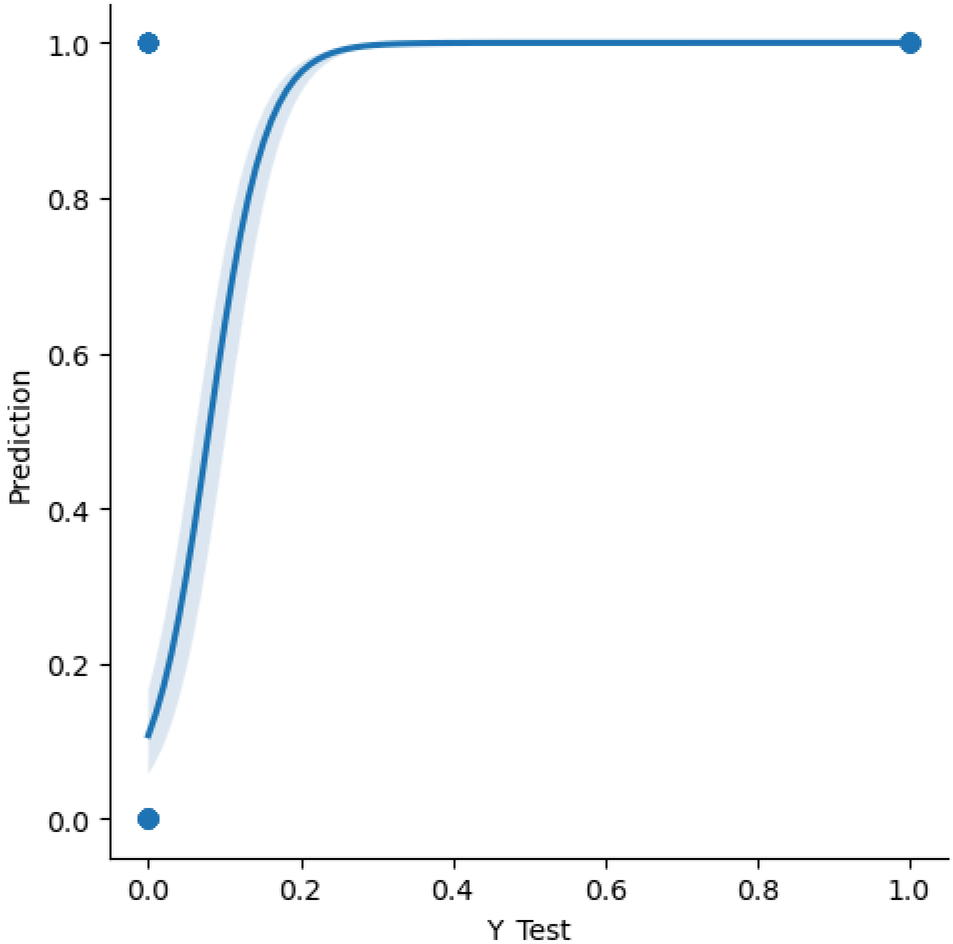
In [86]:
```
from sklearn.metrics import accuracy_score, recall_score
acc = accuracy_score(pred2,ytest)
rec = recall_score(pred2,ytest)
print(acc)
print(rec)
```

```
0.935
0.8586956521739131
```

In [87]:
```
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix
cm = confusion_matrix(pred2,ytest)
cm2 = ConfusionMatrixDisplay(cm)
cm2.plot()
plt.show()
```

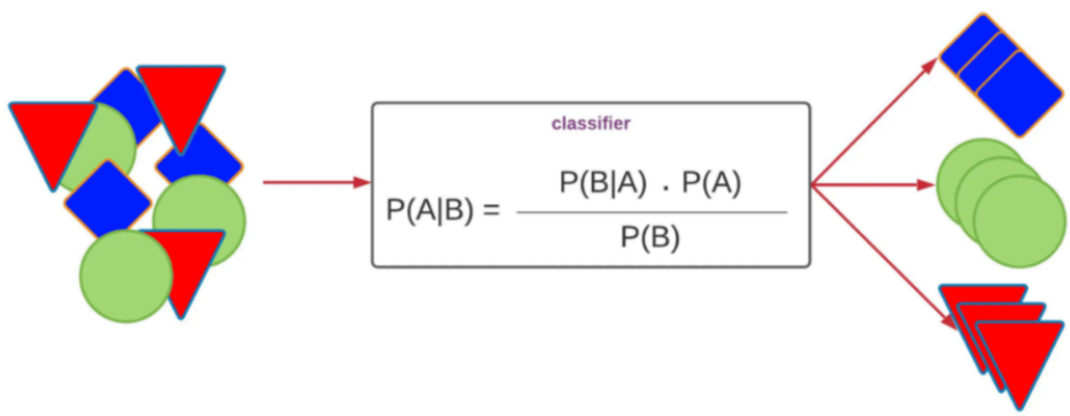


In [88]:
```
df3 = pd.DataFrame({'Y_Test':list(ytest),
                    'Prediction':list(pred2)})
sns.lmplot(x='Y_Test',y='Prediction',data=df3,logistic=True)
plt.show()
```



Naive Bayes:- it works on conditional probability.

```
Naive Bayes uses Bayes Theorem to predict the category of a new
data point.
```



```
There are 3 kind of Bayes Theorem:-
1. Gaussian NB :- data is numerical as well as categorical.
2. Multinomial NB :- only for text datas.
3. Bernoulli NB :- only for categorical datas.
```

In [89]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(xtrain,ytrain)
```

Out[89]:
```
▾ GaussianNB
GaussianNB()
```

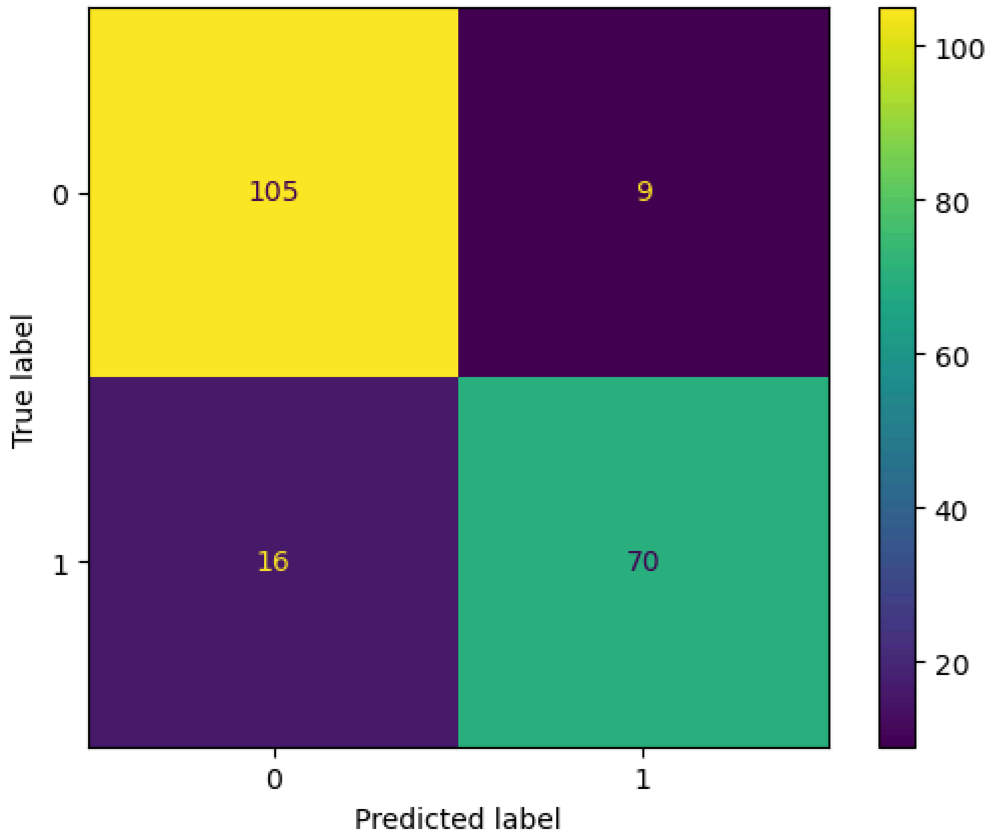
In [90]:
```
pred4 = gnb.predict(xtest)
pred4
```

Out[90]:
```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1], dtype=int64)
```

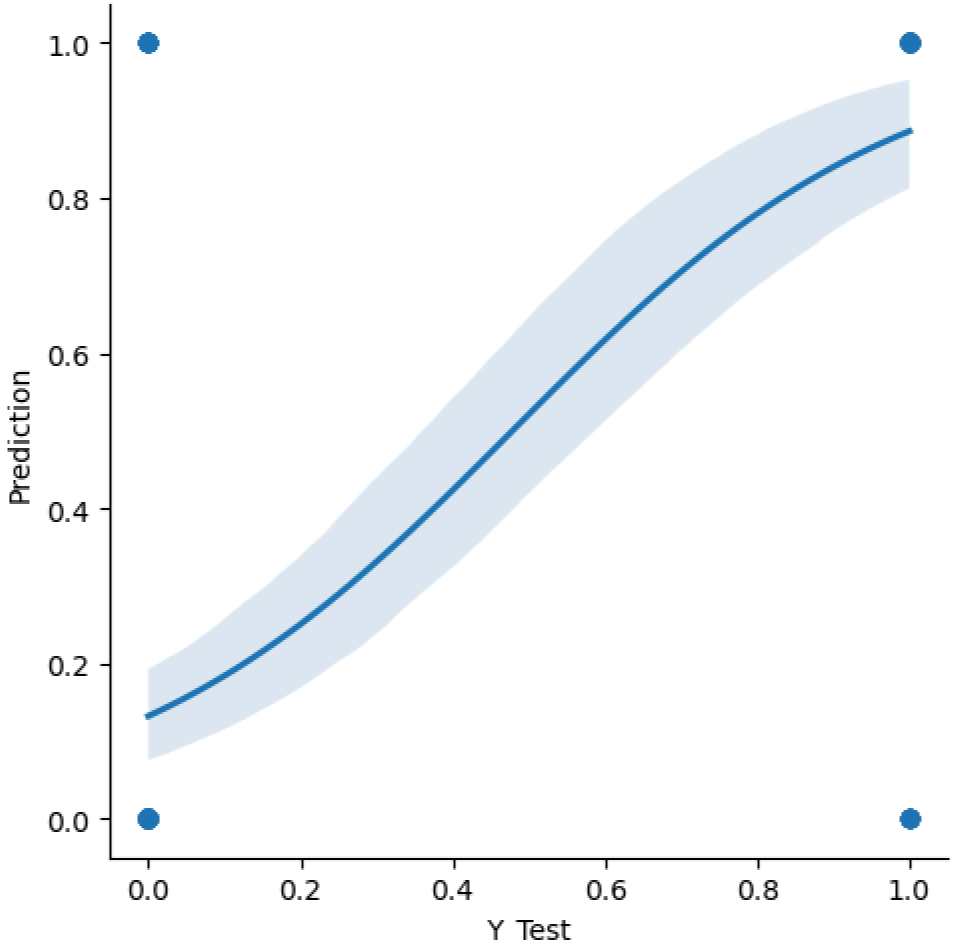
In [91]:
```
from sklearn.metrics import accuracy_score, recall_score
acc = accuracy_score(pred4,ytest)
rec = recall_score(pred4,ytest)
print(acc)
print(rec)
```

```
0.875
0.813953488372093
```

In [92]:
```
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix
cmm = confusion_matrix(pred4,ytest)
cm_2 = ConfusionMatrixDisplay(cmm)
cm_2.plot()
plt.show()
```



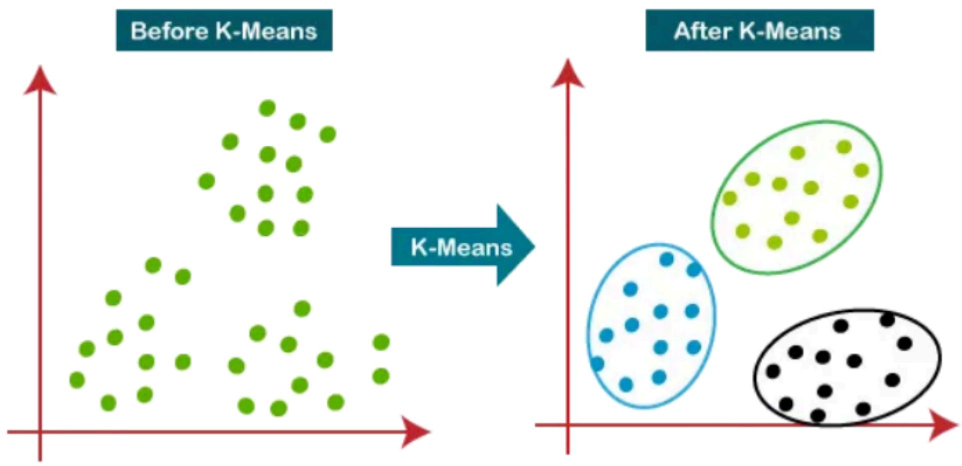
In [93]:
```
df4 = pd.DataFrame({'Y_Test':list(ytest),
                    'Prediction':list(pred4)})
sns.lmplot(x='Y_Test',y='Prediction',data=df4,logistic=True)
plt.show()
```



Unsupervised Learning:-

Machines tries to group the datas based on similar patterns or features.

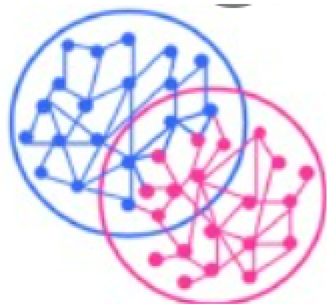
- It refers to the algorithm that learns patterns from unlabelled data.



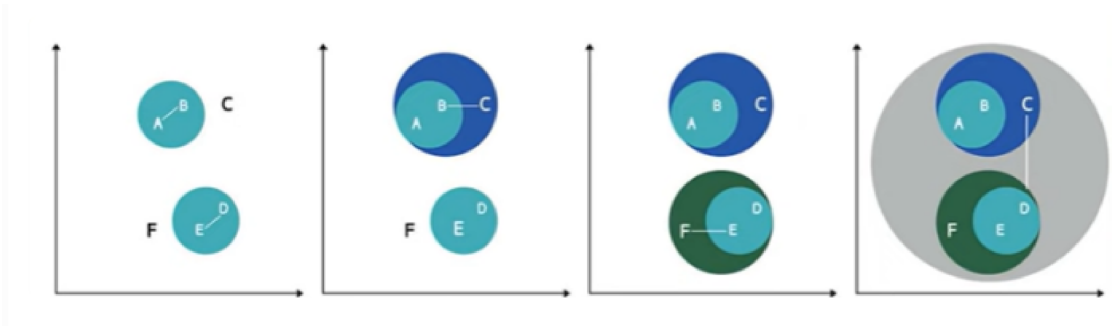
Types of Unsupervised Machine Learning:-
1. Exclusive Unsupervised ML
2. Overlapping Unsupervised ML
3. Hierarchical Unsupervised ML

Exclusive Clustering

Overlapping Clustering



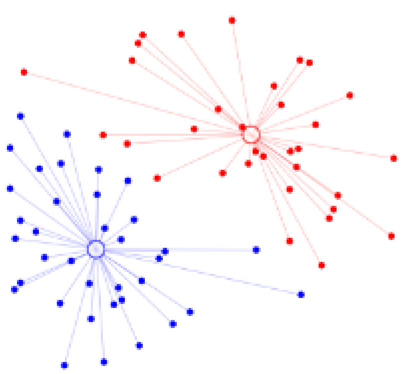
Hierarchical Clustering



K-Means Clustering:- it belongs to exclusive unsupervised machine learning algorithm.

Clustering :- means grouping up of the datas.

1st Iteration :- Centroids were chosen complete randomly.



2nd Iteration :- Again centroids will be reassigned based on mean data points (means are found out by considering all the data points of the particular cluster)

```
The process will be repeated till all the data points have been
clearly clustered.
```

In [94]:
```
# importing the libraries:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [95]:
```
iris = pd.read_csv(r"C:\Users\lab25\Downloads\iris (1)\iris.data",header=None)
iris
```

Out[95]:

	0	1	2	3	4
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 × 5 columns

In [96]:
```
iris.columns = ['SL','SW','PL','PW','Flower']
iris.head()
```

Out[96]:

	SL	SW	PL	PW	Flower
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [97]:
```
# Data Cleaning:-
iris.dtypes
```

Out[97]:
```
SL        float64
SW        float64
PL        float64
PW        float64
Flower     object
dtype: object
```

In [98]:
```
iris.isnull().sum()
```

Out[98]:
```
SL        0
SW        0
PL        0
PW        0
Flower    0
dtype: int64
```

In [99]:
```
for i in iris.columns:
    print(f"{i}:\n{iris[i].unique()}\n")
```

```
SL:
[5.1 4.9 4.7 4.6 5.  5.4 4.4 4.8 4.3 5.8 5.7 5.2 5.5 4.5 5.3 7.  6.4 6.9
 6.5 6.3 6.6 5.9 6.  6.1 5.6 6.7 6.2 6.8 7.1 7.6 7.3 7.2 7.7 7.4 7.9]

SW:
[3.5 3.  3.2 3.1 3.6 3.9 3.4 2.9 3.7 4.  4.4 3.8 3.3 4.1 4.2 2.3 2.8 2.4
 2.7 2.  2.2 2.5 2.6]

PL:
[1.4 1.3 1.5 1.7 1.6 1.1 1.2 1.  1.9 4.7 4.5 4.9 4.  4.6 3.3 3.9 3.5 4.2
 3.6 4.4 4.1 4.8 4.3 5.  3.8 3.7 5.1 3.  6.  5.9 5.6 5.8 6.6 6.3 6.1 5.3
 5.5 6.7 6.9 5.7 6.4 5.4 5.2]

PW:
[0.2 0.4 0.3 0.1 0.5 0.6 1.4 1.5 1.3 1.6 1.  1.1 1.8 1.2 1.7 2.5 1.9 2.1
 2.2 2.  2.4 2.3]

Flower:
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

In [100…
```
# We need to drop the op column 'flower'.
ip = iris.drop('Flower',axis=1)
ip.head()
```

Out[100…

	SL	SW	PL	PW
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

In [101…

```
# To Apply the K-Means Clustering Algorithm:-
from sklearn.cluster import KMeans
km = KMeans(n_clusters=3)
km.fit(ip)
```

Out[101…

```
▾ KMeans
KMeans(n_clusters=3)
```

In [102…

```
# prediction:
km_pred = km.predict(ip)
km_pred
```

Out[102…

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

In [103…

```
centroids = km.cluster_centers_
centroids
```

Out[103…

```
array([[5.9016129 , 2.7483871 , 4.39354839, 1.43387097],
       [5.006     , 3.418     , 1.464     , 0.244     ],
       [6.85      , 3.07368421, 5.74210526, 2.07105263]])
```

In [104…

```
iris['Prediction'] = km_pred
iris
```

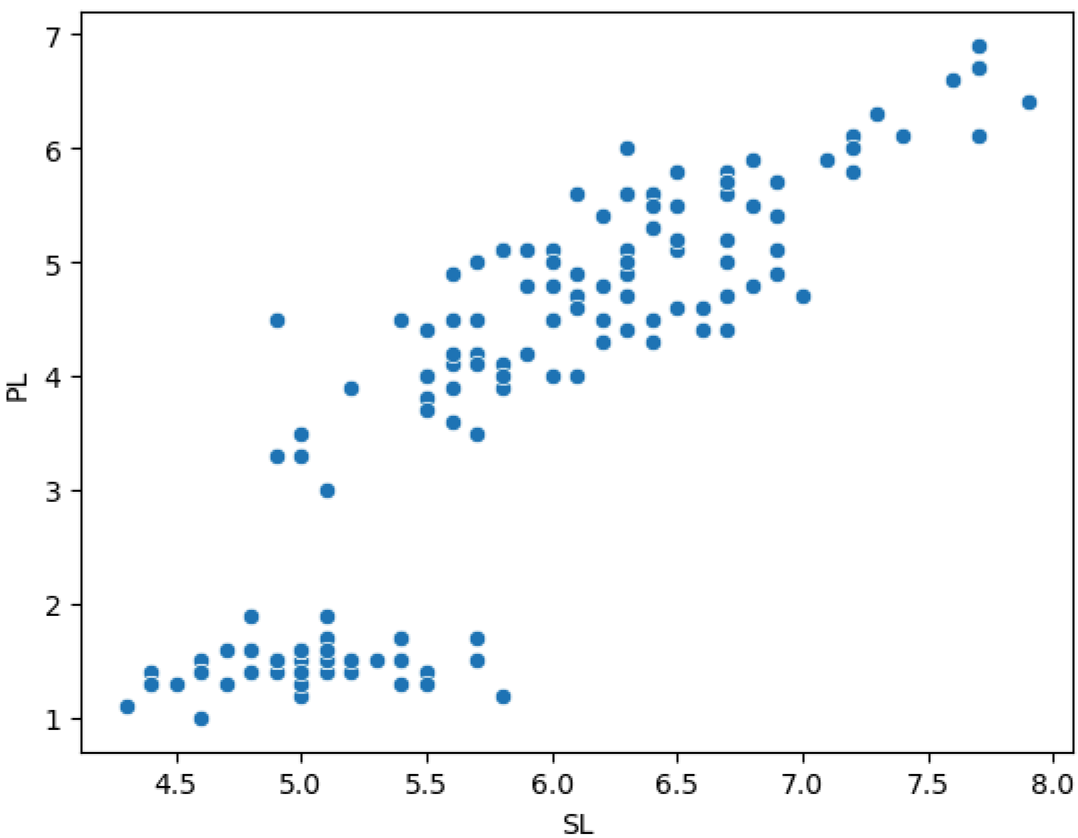
Out[104…

	SL	SW	PL	PW	Flower	Prediction
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	2
146	6.3	2.5	5.0	1.9	Iris-virginica	0
147	6.5	3.0	5.2	2.0	Iris-virginica	2
148	6.2	3.4	5.4	2.3	Iris-virginica	2
149	5.9	3.0	5.1	1.8	Iris-virginica	0

150 rows × 6 columns

In [105…

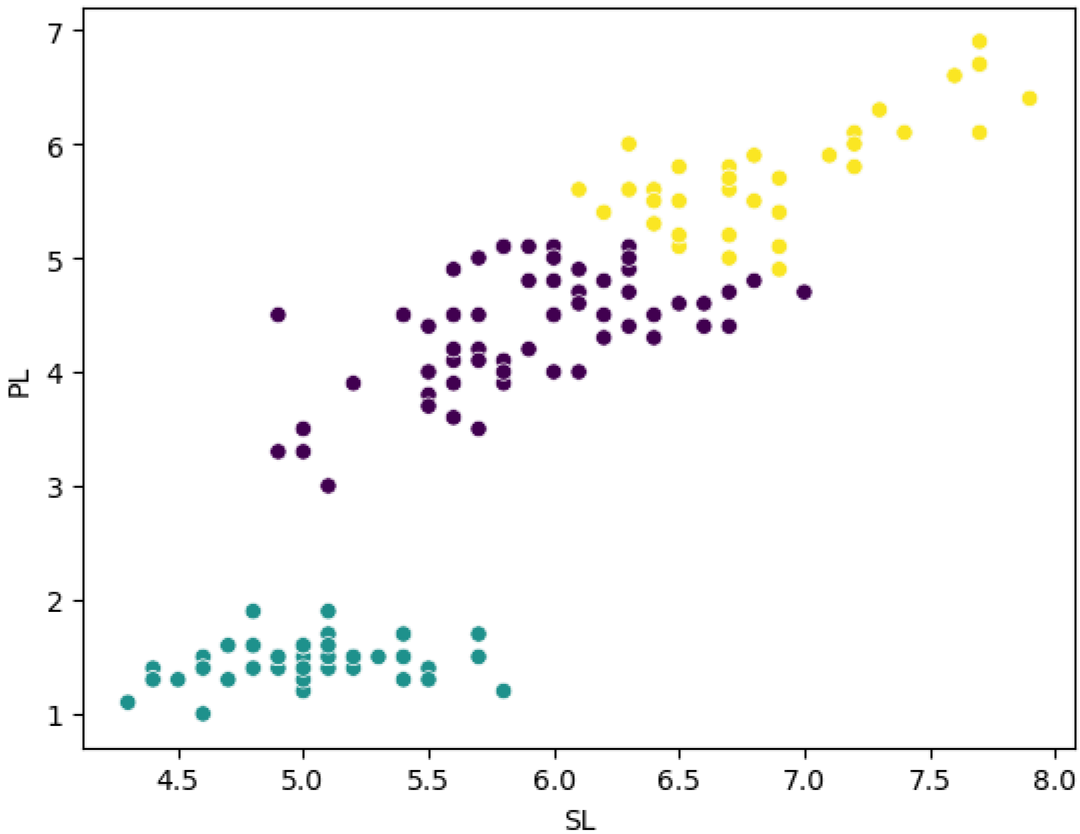
```
sns.scatterplot(x=iris.SL,y=iris.PL)
plt.show()
```



In [106…

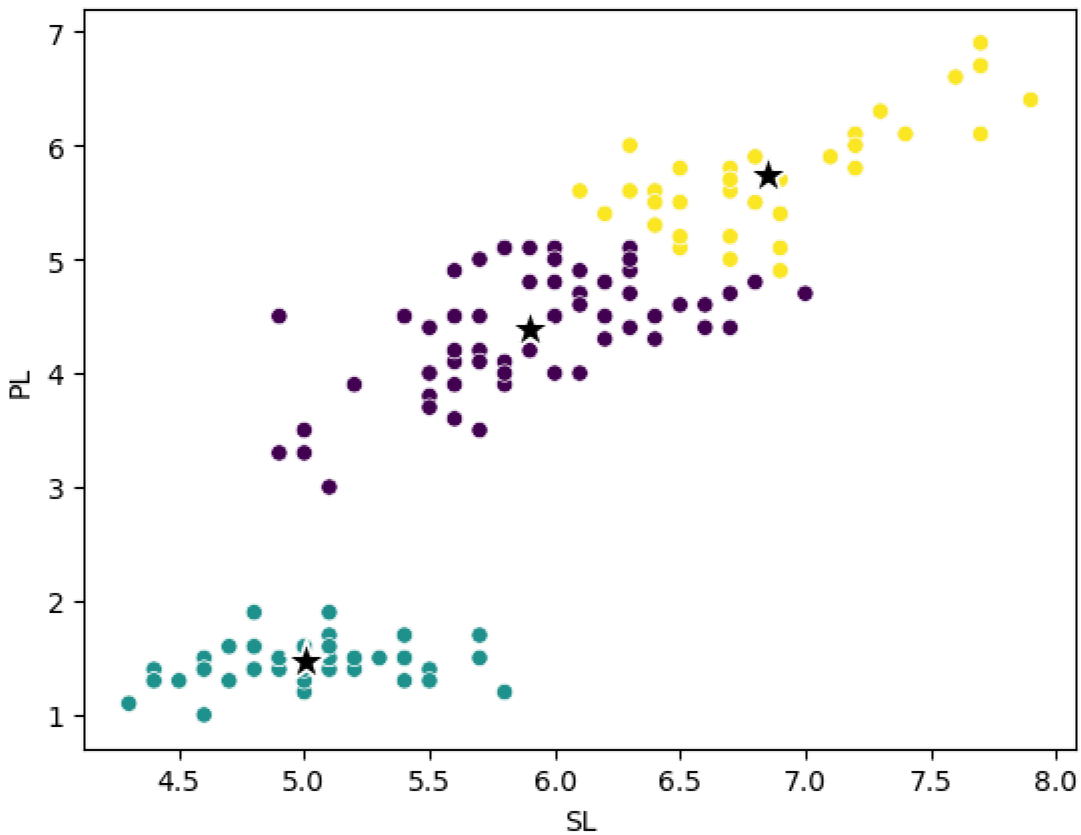
```
sns.scatterplot(x=iris.SL,y=iris.PL,
                c = km_pred)
```

```
plt.show()
```



In [107…

```
# Plotting along the centroid values:-
sns.scatterplot(x=iris.SL,y=iris.PL,
                c = km_pred)
sns.scatterplot(x=centroids[:,0],y=centroids[:,2],
               s=250,marker='*',color='k')
plt.show()
```



In [ ]: