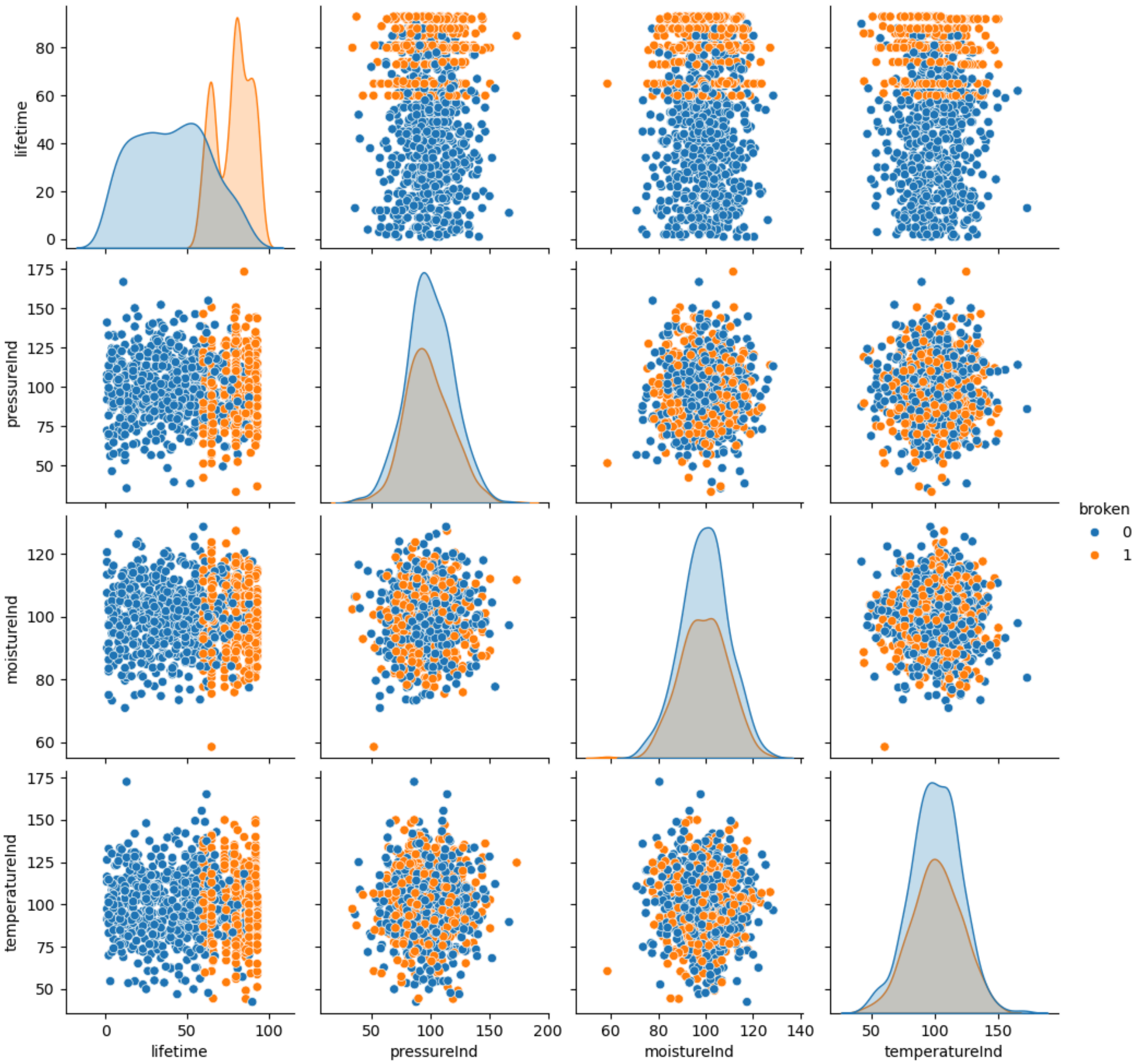


```
In [9]: # Encoding:
        from sklearn.preprocessing import LabelEncoder
        le = LabelEncoder()
```

```
In [10]: # changing the team's column:
         main.team = le.fit_transform(main.team)
```

```
In [11]: main.team.unique()
```

```
Out[11]: array([0, 2, 1])
```

```
In [12]: le.inverse_transform([0, 2, 1])
```

```
Out[12]: array(['TeamA', 'TeamC', 'TeamB'], dtype=object)
```

```
In [13]: main.provider = le.fit_transform(main.provider)
         main.provider.unique()
```

```
Out[13]: array([3, 0, 1, 2])
```

In [14]:
```
le.inverse_transform([3, 0, 1, 2])
```

Out[14]:
```
array(['Provider4', 'Provider1', 'Provider2', 'Provider3'], dtype=object)
```

In [15]:
```
main
```

Out[15]:

	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

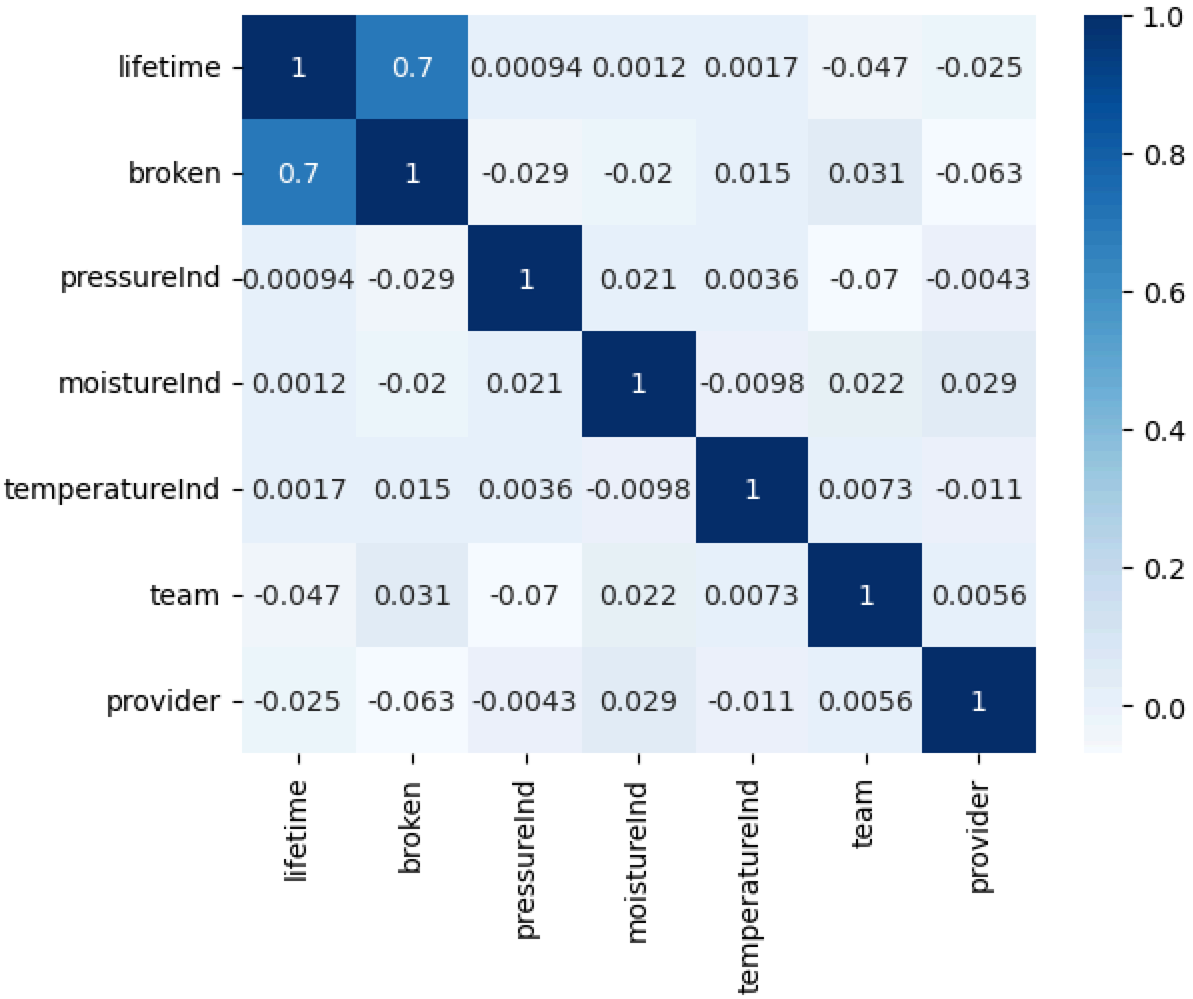
1000 rows × 7 columns

In [17]:
```
# finding the correlation
cr = main.corr()
cr
```

Out[17]:

	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 [19]:
```
sns.heatmap(cr,annot=True,cmap='Blues')
plt.show()
```



In [20]:
```
# Creation of ip/op:-
ip = main.drop('broken',axis=1)
```

In [21]:
```
ip.head()
```

Out[21]:

	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 [22]:
```
op = main.broken
op.head()
```

Out[22]:

```
0    0
1    1
2    0
3    1
4    0
Name: broken, dtype: int64
```

In [23]:

```
# 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 [24]:

```
xtrain.head()
```

Out[24]:

	lifetime	pressureInd	moistureInd	temperatureInd	team	provider
117	80	150.695689	111.988761	85.863547	1	0
389	65	100.356213	103.824801	86.087941	1	2
419	50	110.446074	99.596733	100.252123	1	3
662	72	105.003465	113.907966	92.929963	0	1
145	19	64.285657	114.037572	123.755117	2	3

In [25]:

```
xtest.head()
```

Out[25]:

	lifetime	pressureInd	moistureInd	temperatureInd	team	provider
217	29	121.389010	83.710846	101.806530	2	3
190	60	90.846150	107.245503	102.928899	2	2
212	62	96.140671	79.334977	139.352002	1	0
66	39	87.392841	106.166234	84.521713	1	3
833	49	111.472254	85.849245	128.333248	2	0

In [26]:

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

In [27]:

```
xtrain = sc.fit_transform(xtrain)
xtest = sc.fit_transform(xtest)
```

In [28]:

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

In [29]:

```
lr.fit(xtrain,ytrain)
```

Out[29]:

```
▾ LogisticRegression
LogisticRegression()
```

In [31]:

```python
# Prediction:-
ypred = lr.predict(xtest)
ypred
```

Out[31]:

```
array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0], 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 [33]:

```python
from sklearn.metrics import recall_score,accuracy_score
acc = accuracy_score(ypred,ytest)
rec = recall_score(ypred,ytest)
```

In [34]:

```python
print(f"Accuracy:",acc)
print(f"Recall:",rec)
```

```
Accuracy: 0.765
Recall: 0.6710526315789473
```

In [35]:

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

In [36]:

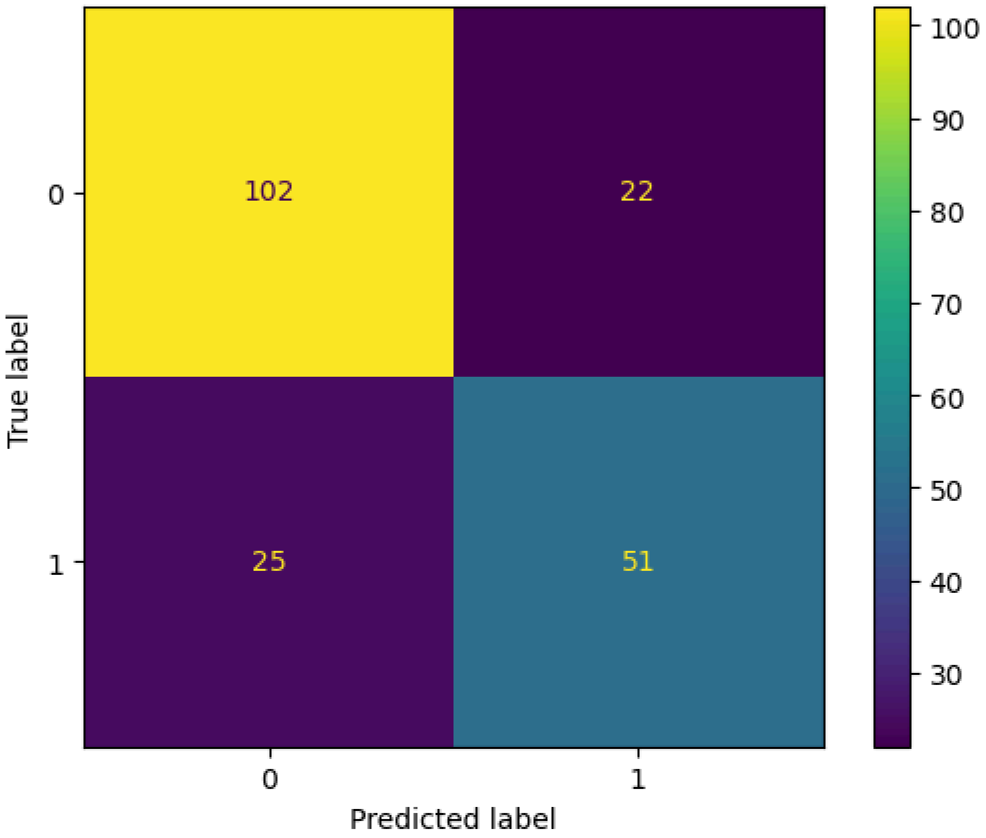
```python
cm
```

Out[36]:

```
array([[102,  22],
       [ 25,  51]], dtype=int64)
```

In [37]:

```
cmd = ConfusionMatrixDisplay(cm)
cmd.plot()
plt.show()
```



KNN:- (K-Nearest Neighbor)

In [38]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
```

In [40]:

```
knn.fit(xtrain,ytrain)
```

Out[40]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [41]:

```
knn.predict(xtest)
```

Out[41]:

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

In [42]:

```
from sklearn.metrics import recall_score,accuracy_score
acc = accuracy_score(ypred,ytest)
rec = recall_score(ypred,ytest)
print(f"Accuracy:",acc)
print(f"Recall:",rec)
```

```
Accuracy: 0.765
Recall: 0.6710526315789473
```

In [43]:

```
# Confusion matrix:-
from sklearn.metrics import
ConfusionMatrixDisplay,confusion_matrix
cm1 = confusion_matrix(ypred,ytest)
```

In [44]:

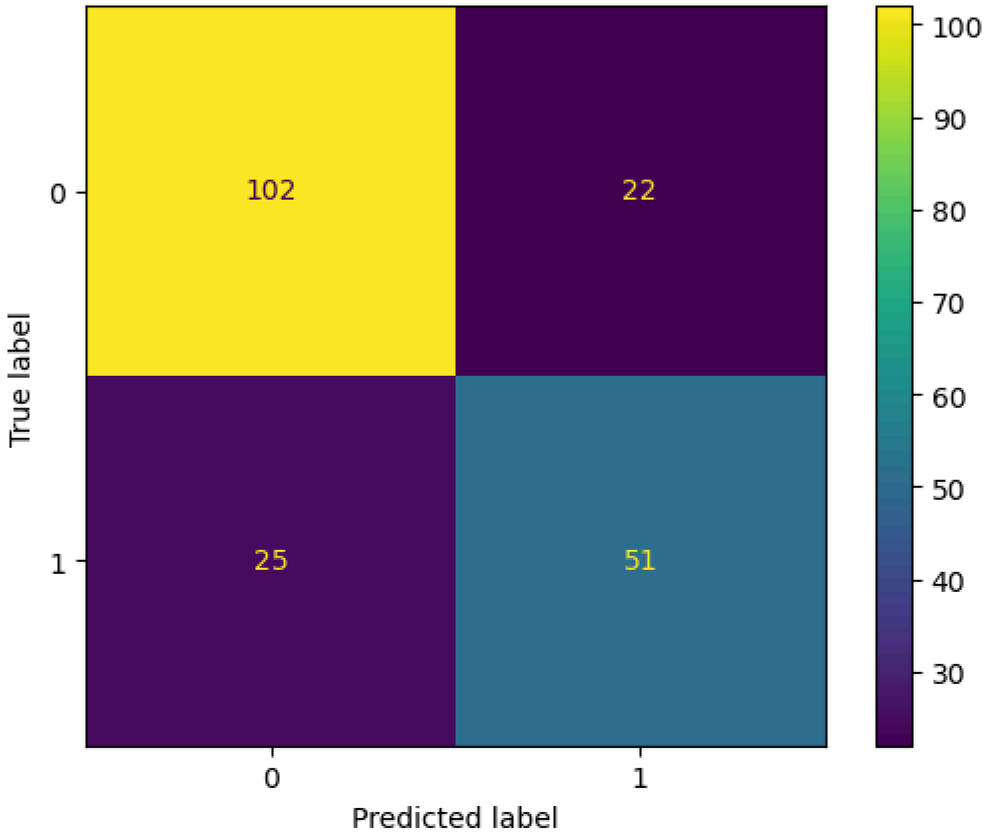
```
cm1
```

Out[44]:

```
array([[102,  22],
       [ 25,  51]], dtype=int64)
```

In [45]:

```
cmd = ConfusionMatrixDisplay(cm1)
cmd.plot()
plt.show()
```



In [ ]:

```

```