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

In [2]:
```
df=pd.read_csv("https://raw.githubusercontent.com/anshupandey/Machine_Learning_T
```

In [3]:
```
df
```

Out[3]:

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider
0	56	0	92.178854	104.230204	96.517159	TeamA	Provider4
1	81	1	72.075938	183.065701	87.271062	TeamC	Provider4
2	60	0	96.272254	77.801376	112.196170	TeamA	Provider1
3	86	1	94.406461	178.493608	72.025374	TeamC	Provider2
4	34	0	97.752899	99.413492	103.756271	TeamB	Provider1
5	30	0	87.678801	115.712262	89.792105	TeamA	Provider1
6	68	0	94.614174	85.702236	142.827001	TeamB	Provider2
7	65	1	96.483303	193.046797	98.316190	TeamB	Provider3
8	23	0	105.486158	118.291997	96.028822	TeamB	Provider2
9	81	1	99.178235	199.138717	95.492965	TeamC	Provider4
10	38	0	97.817844	111.074168	94.942443	TeamB	Provider4
11	29	0	67.812251	96.107846	122.371809	TeamA	Provider1
12	65	1	86.366111	92.561972	96.667950	TeamA	Provider3
13	65	1	76.144654	93.973454	108.944273	TeamA	Provider3
14	82	1	103.107263	103.673197	79.504532	TeamC	Provider4
15	80	1	88.414079	97.362755	89.319813	TeamB	Provider1
16	48	0	84.355049	103.174919	102.399715	TeamC	Provider3
17	80	1	79.669255	95.926908	99.735635	TeamB	Provider1
18	92	1	86.229109	86.111900	111.842548	TeamB	Provider2
19	88	1	84.179420	114.074066	83.627450	TeamB	Provider4
20	74	1	100.005923	103.502876	96.876035	TeamC	Provider1
21	65	0	115.607560	103.013573	101.616843	TeamB	Provider2
22	61	0	97.697189	101.884961	130.135676	TeamC	Provider1
23	35	0	101.415623	89.753746	114.909434	TeamC	Provider2
24	26	0	118.978697	105.298916	115.247085	TeamA	Provider2
25	63	0	102.112775	106.250695	92.315195	TeamA	Provider3
26	88	1	129.124338	103.344720	92.846359	TeamA	Provider4
27	79	1	109.033036	92.754120	124.760445	TeamB	Provider1
28	53	0	107.298070	85.055291	109.841003	TeamA	Provider3
29	73	1	127.263954	103.557292	133.859669	TeamC	Provider1
...	...	...	...	...	...	...	...
970	81	1	77.699000	109.949997	143.951991	TeamC	Provider4
971	81	1	79.749386	78.281154	87.937399	TeamC	Provider4
972	81	0	90.780638	99.865378	99.601190	TeamC	Provider2

	lifetime	broken	pressureInd	moistureInd	temperatureInd	team	provider
973	60	1	117.706933	114.818523	104.299780	TeamC	Provider3
974	64	0	119.678005	99.679383	114.954666	TeamA	Provider4
975	88	1	70.168029	93.124583	86.059308	TeamB	Provider4
976	86	1	83.276925	109.779808	101.179966	TeamC	Provider2
977	80	1	111.991442	106.257524	110.334142	TeamB	Provider1
978	65	1	102.776756	111.451780	112.955770	TeamA	Provider3
979	55	0	87.946098	96.137489	107.111784	TeamA	Provider2
980	25	0	92.573859	103.656374	103.262856	TeamA	Provider2
981	52	0	88.205357	110.346931	99.496485	TeamC	Provider4
982	29	0	140.372907	104.539031	82.253512	TeamA	Provider1
983	58	0	95.307528	97.490135	102.313446	TeamA	Provider3
984	66	1	76.769908	110.104446	76.717516	TeamB	Provider3
985	36	0	119.033947	120.111932	70.325114	TeamA	Provider1
986	88	1	113.766501	88.423676	110.093054	TeamB	Provider4
987	67	0	109.768711	93.378117	122.159959	TeamB	Provider2
988	45	0	125.701872	101.996722	58.300958	TeamA	Provider2
989	30	0	93.882557	103.129015	110.232049	TeamB	Provider3
990	65	1	86.987576	123.656124	106.650884	TeamA	Provider3
991	76	0	87.852558	94.286687	114.103152	TeamC	Provider2
992	60	1	115.750787	98.762229	101.206650	TeamC	Provider3
993	65	1	94.683966	108.961422	118.027394	TeamA	Provider3
994	80	1	110.229747	101.445523	75.630577	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 [4]:
```
df.shape
```

Out[4]: (1000, 7)

Data Exploration-

In [5]:
```
df.describe()
```

Out[5]:

	lifetime	broken	pressureInd	moistureInd	temperatureInd
count	1000.000000	1000.000000	996.000000	1000.000000	997.000000
mean	55.195000	0.397000	98.681100	111.088723	100.553499
std	26.472737	0.489521	19.879703	41.839005	19.592059
min	1.000000	0.000000	33.481917	70.928815	42.279598
25%	34.000000	0.000000	85.562282	94.532547	87.672094
50%	60.000000	0.000000	97.311091	102.844084	100.528015
75%	80.000000	1.000000	112.253190	113.532970	113.522496
max	93.000000	1.000000	173.282541	1156.493254	172.544140

Here we observe that max(moistureInd)>> mean(75%of moistureInd) i.e moistureInd have one unrealistic value.

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
lifetime          1000 non-null int64
broken            1000 non-null int64
pressureInd       996 non-null float64
moistureInd       1000 non-null float64
temperatureInd    997 non-null float64
team              1000 non-null object
provider          1000 non-null object
dtypes: float64(3), int64(2), object(2)
memory usage: 54.8+ KB
```

From above table we observe that "pressureInd" coloumn have 4 missing values & 'temperature' coloumn have 3 missing value

In [7]:
```
df.median()
```

Out[7]:
```
lifetime           60.000000
broken              0.000000
pressureInd        97.311091
moistureInd       102.844084
temperatureInd    100.528015
dtype: float64
```

In [8]:
```
a=df.duplicated().sum()
a
```

Out[8]:
```
0
```

Therfore dataframe doesn't contain any duplicated row.

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
lifetime          0
broken            0
pressureInd       4
moistureInd       0
temperatureInd    3
team              0
provider          0
dtype: int64
```

4 & 3 Null values occur in pressureInd & TemperatureInd coloumn respectively

In [10]:
```
df.skew()
```

Out[10]:
```
lifetime          -0.407597
broken             0.421663
pressureInd        0.117541
moistureInd       15.982324
temperatureInd    -0.070839
dtype: float64
```

Data cleaning:-

As pressureInd coloumn have positive skew value, filling null attributes with mean value will be more appropriate.

In [11]:
```
df.pressureInd.fillna(df.pressureInd.mean(),inplace=(True))
df.isnull().sum()
```

Out[11]:
```
lifetime          0
broken            0
pressureInd       0
moistureInd       0
temperatureInd    3
team              0
provider          0
dtype: int64
```

In [12]:
```
df.fillna(df.median(),inplace=(True))
df.isnull().sum()
```

Out[12]:
```
lifetime          0
broken            0
pressureInd       0
moistureInd       0
temperatureInd    0
team              0
provider          0
dtype: int64
```

As 'moistureInd' have one unrealistic value it must be removed. using the following method:-

In [13]:
```
df['moistureInd'].replace({df.moistureInd.max():df.moistureInd.mean()},inplace=Tr
df.moistureInd.max()
```

Out[13]:
```
199.13871701
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
lifetime          1000 non-null int64
broken            1000 non-null int64
pressureInd       1000 non-null float64
moistureInd       1000 non-null float64
temperatureInd    1000 non-null float64
team              1000 non-null object
provider          1000 non-null object
dtypes: float64(3), int64(2), object(2)
memory usage: 54.8+ KB
```

In [15]:
```
df.columns
```

Out[15]:
```
Index(['lifetime', 'broken', 'pressureInd', 'moistureInd', 'temperatureInd',
       'team', 'provider'],
      dtype='object')
```

In [16]:
```
df.team.unique()
```

Out[16]:
```
array(['TeamA', 'TeamC', 'TeamB'], dtype=object)
```

In [17]:
```
df.provider.unique()
```

Out[17]:
```
array(['Provider4', 'Provider1', 'Provider2', 'Provider3'], dtype=object)
```

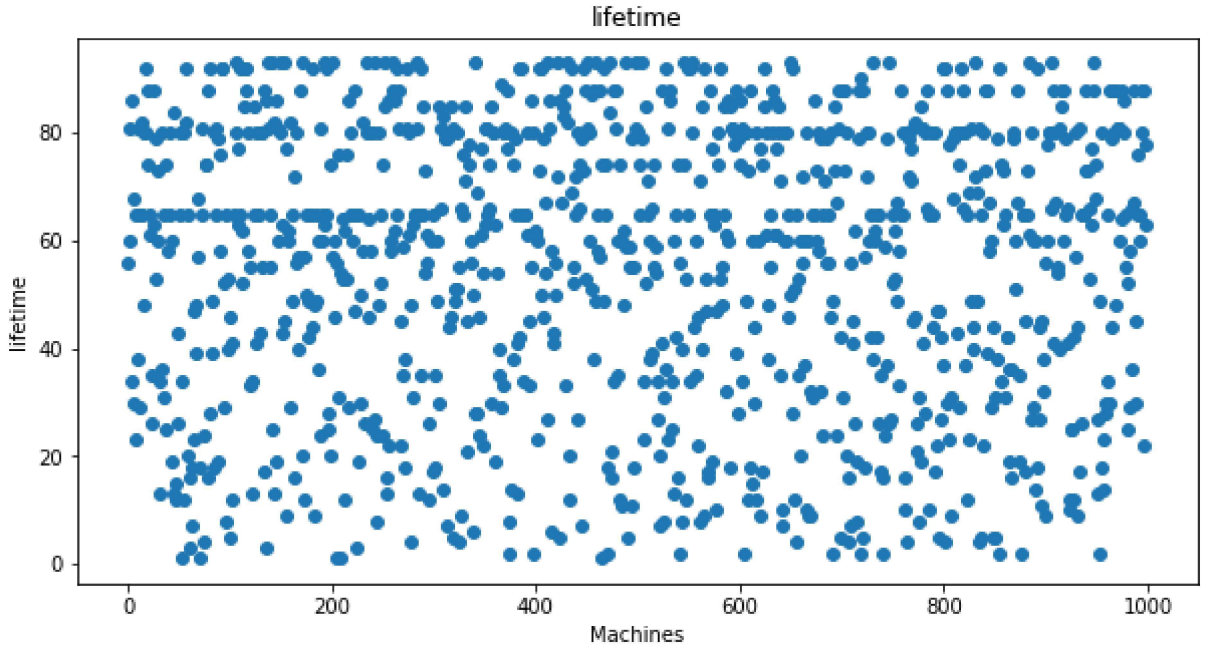
In [18]:
```
df.broken.unique()
```

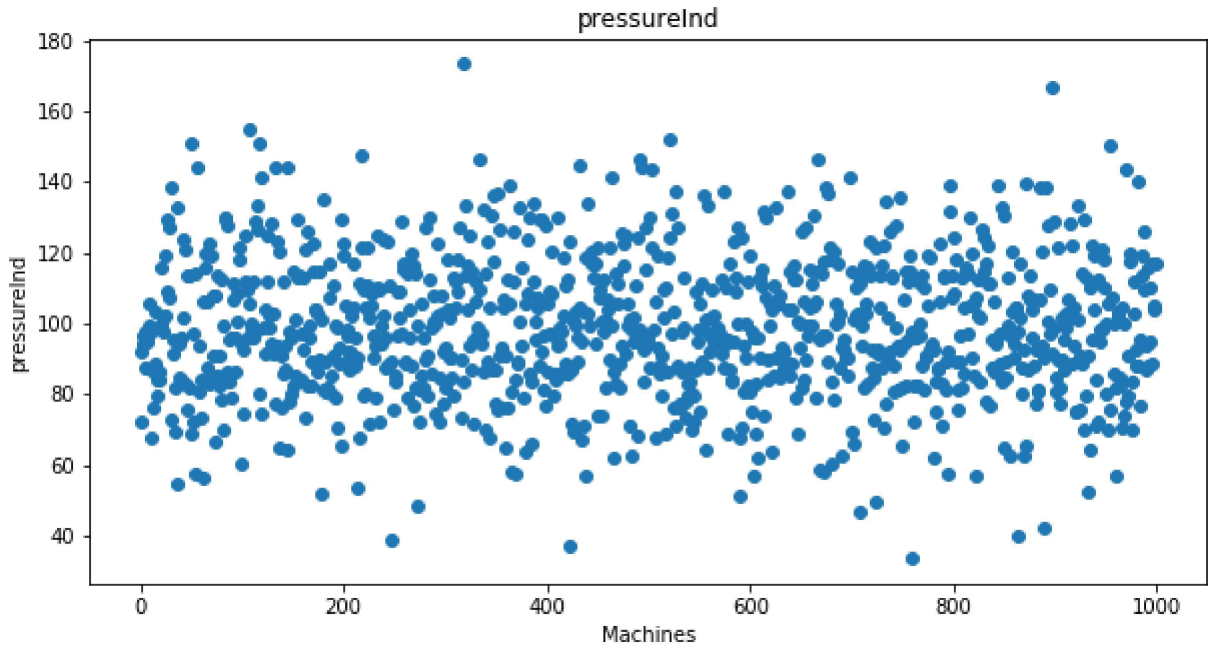
Out[18]:
```
array([0, 1], dtype=int64)
```

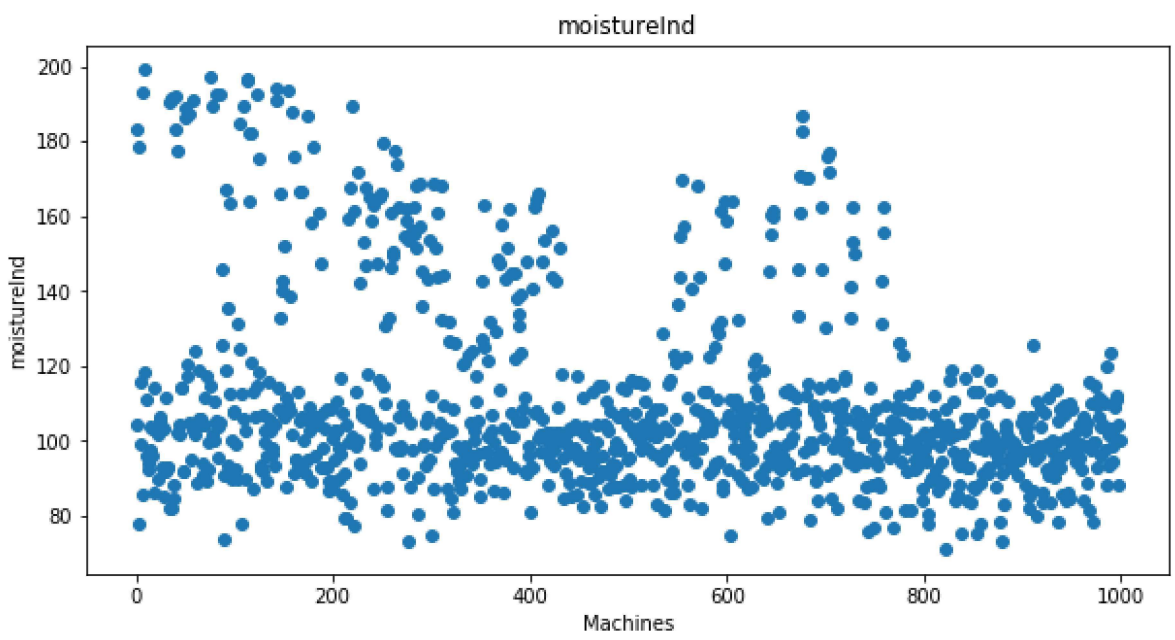
Univariate Analysis

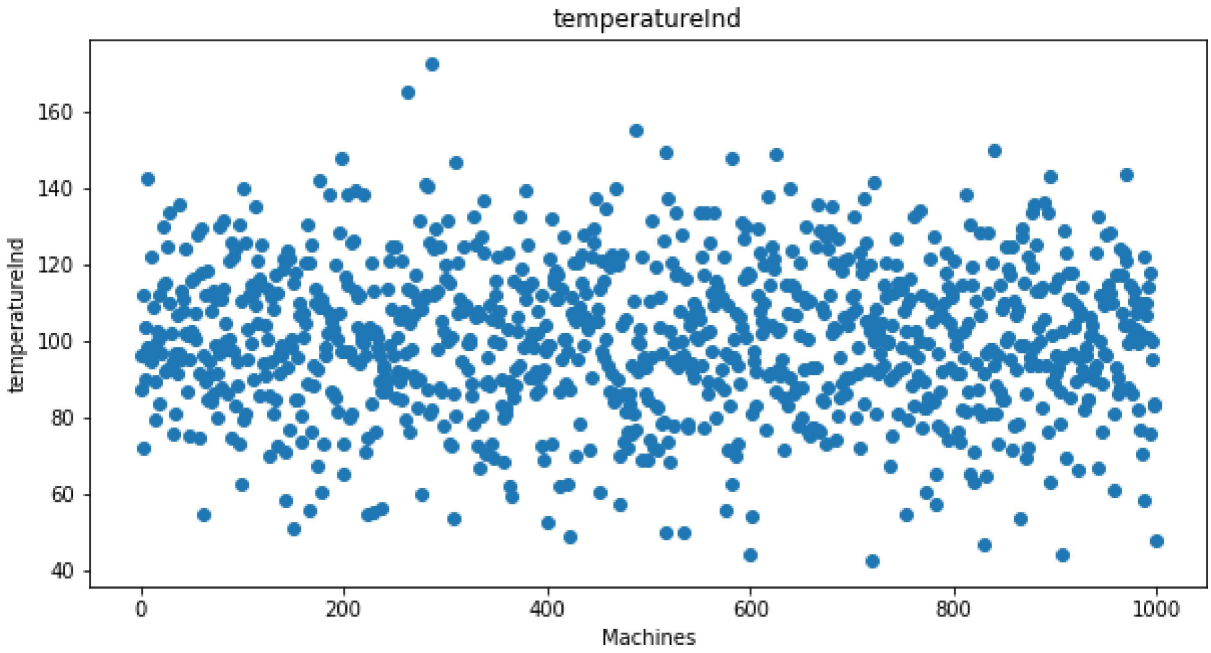
In [19]:
```
numerics=['lifetime', 'pressureInd', 'moistureInd', 'temperatureInd']
cat=['broken','team', 'provider']
```

In [20]:
```
for col in numerics:
    plt.figure(figsize=(10,5))
    plt.scatter(np.arange(1000),df[col])
    plt.xlabel("Machines")
    plt.ylabel(col)
    plt.title(col)
    plt.show()
```

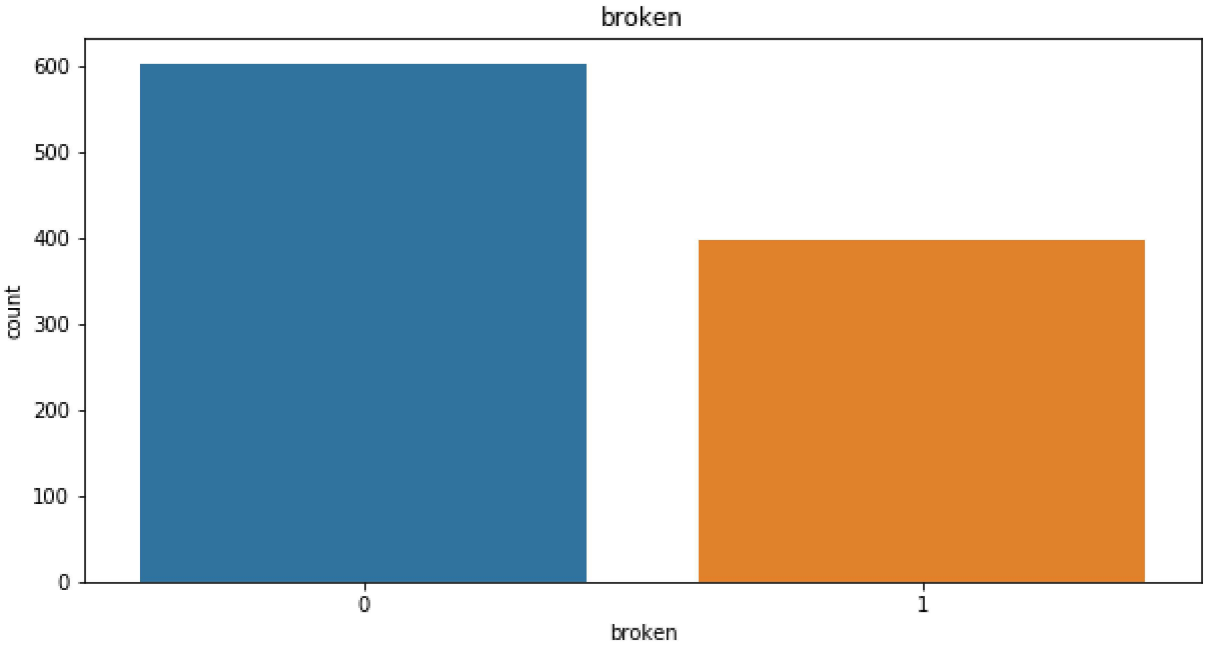


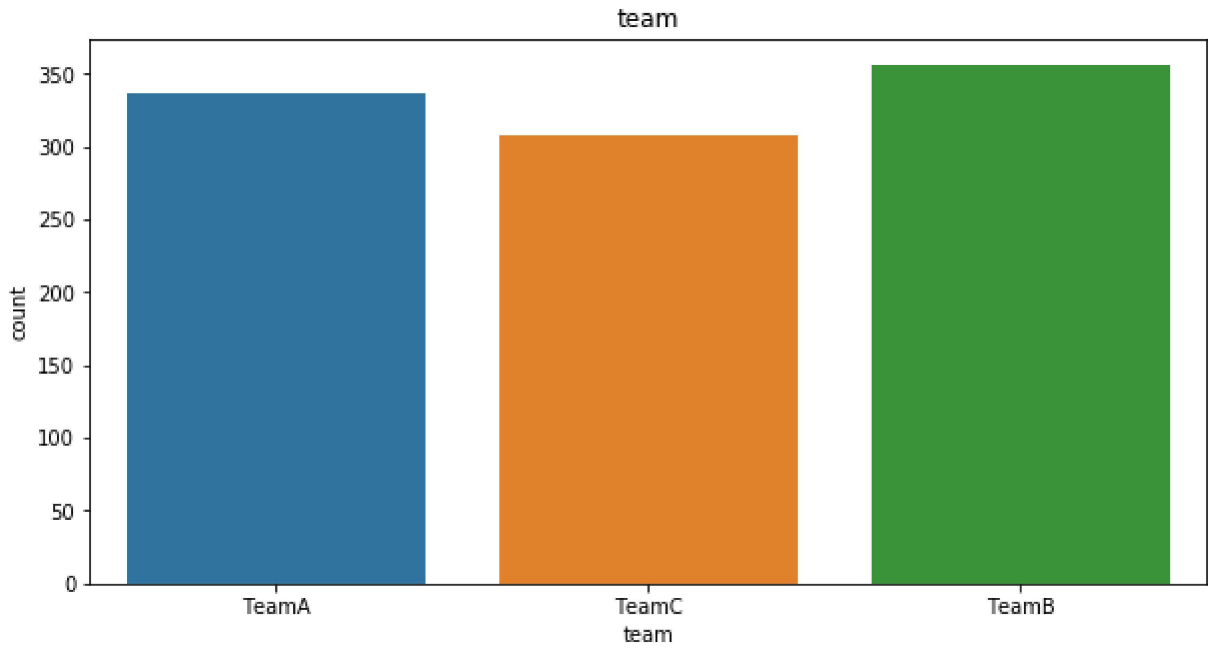


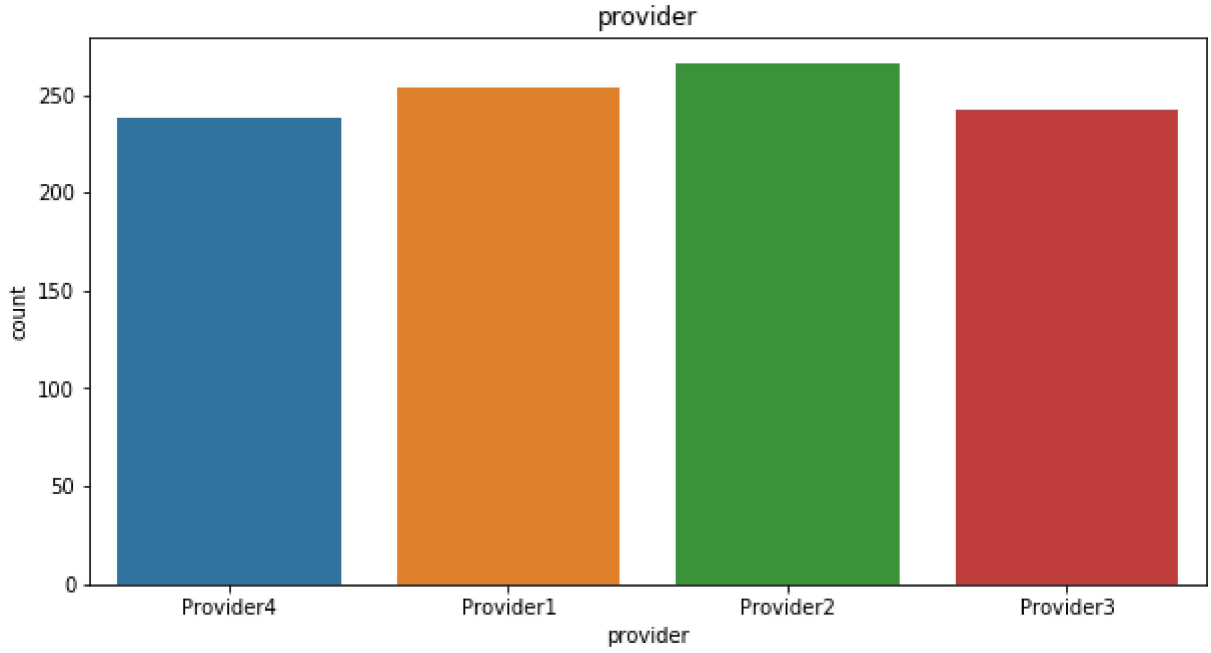

moistureInd
200
180
160
140
120
100
80
moistureInd
0
200
400
600
800
1000
Machines


temperatureInd
160
140
120
100
80
60
40
temperatureInd
0
200
400
600
800
1000
Machines

In [21]:
```
for col in cat:
    plt.figure(figsize=(10,5))
   # plt.scatter(np.arange(10000),df[col])
    sns.countplot(df[col])
    #plt.xlabel("Customers")
    #plt.ylabel(col)
    plt.title(col)
    plt.show()
```



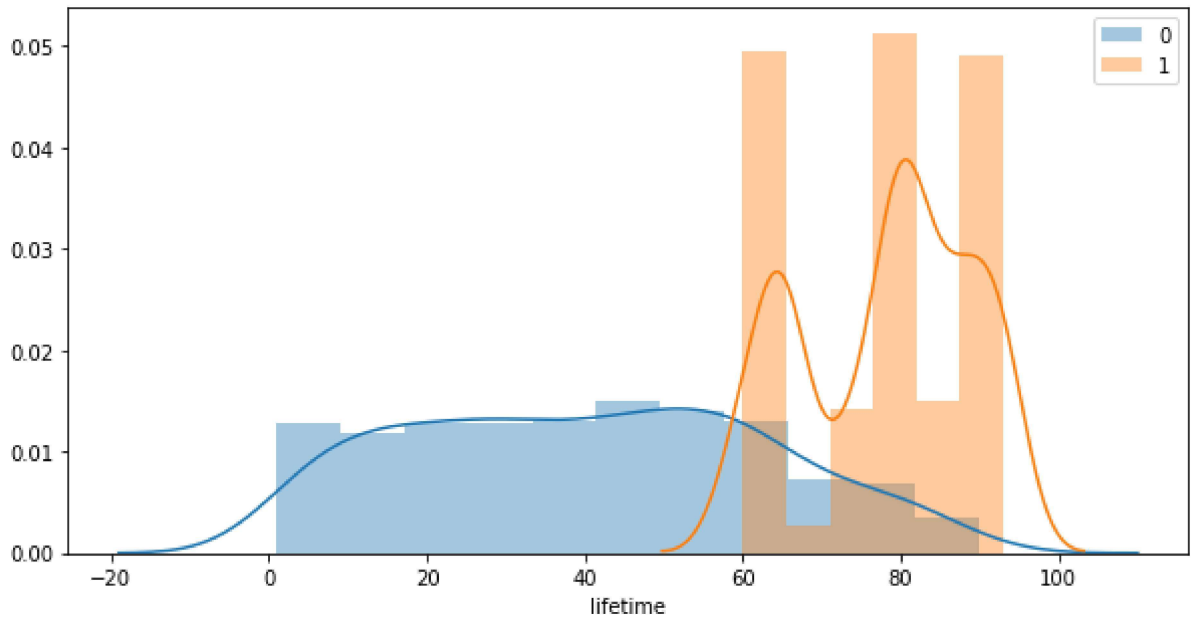

team
350
300
250
200
150
100
50
0
count
TeamA
TeamC
TeamB
team


provider
250
200
150
100
50
0
count
Provider4
Provider1
Provider2
Provider3
provider

BIVARIATE ANALYSIS

In [22]:
```
plt.figure(figsize=(10,5))
sns.distplot(df.lifetime[df.broken==0])
sns.distplot(df.lifetime[df.broken==1])
plt.legend(['0','1'])
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWar
ning: Using a non-tuple sequence for multidimensional indexing is deprecated; u
se `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpre
ted as an array index, `arr[np.array(seq)]`, which will result either in an err
or or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

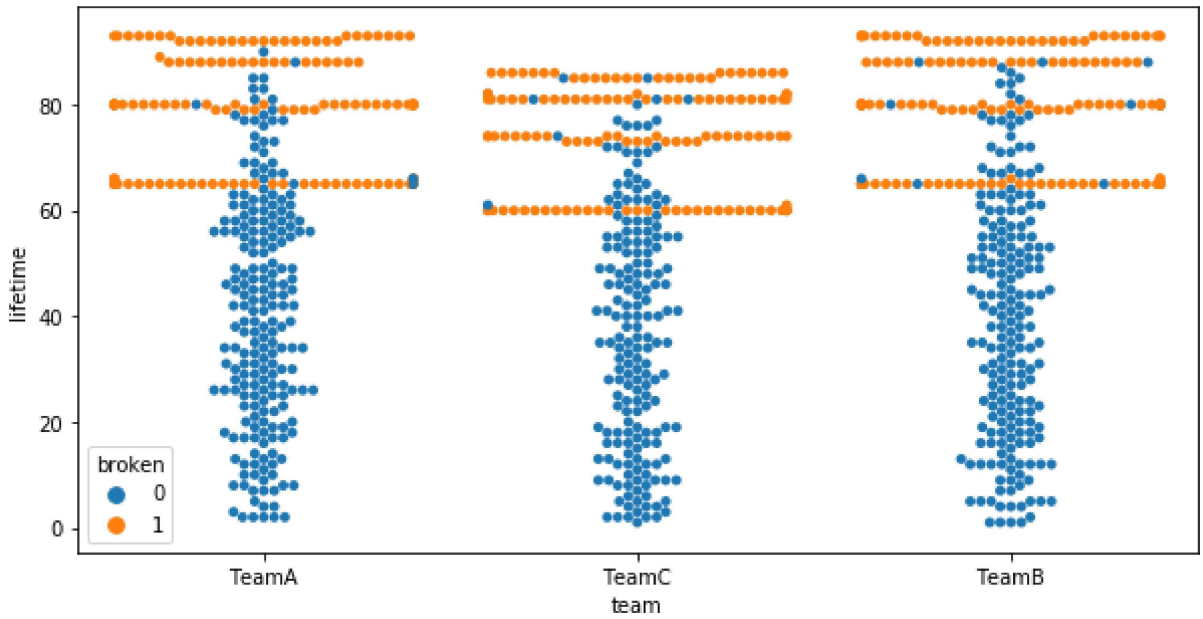


By analysing this graph we can say that:-

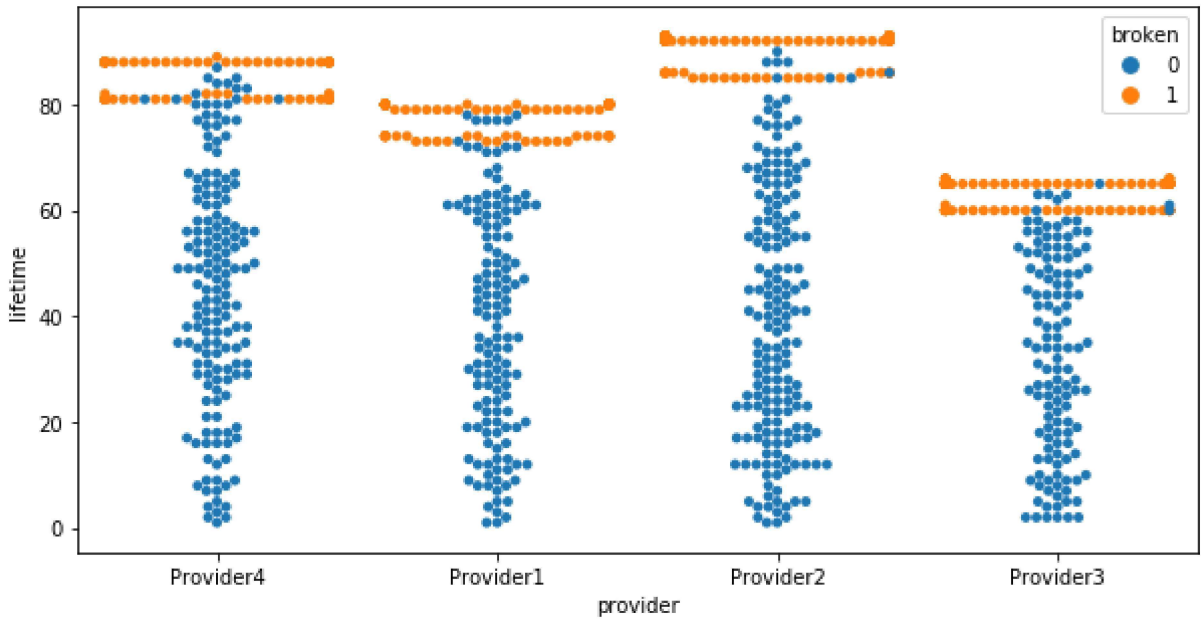
```
Machines that have lifetime>60 have more probability of being broken.
```

Multivariate analysis

In [23]:
```
plt.figure(figsize=(10,5))
sns.swarmplot(x='team',y='lifetime',hue='broken',data=df)
plt.show()
```

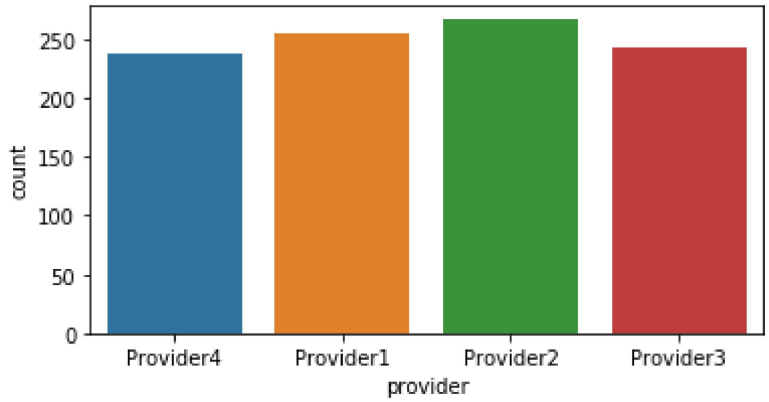


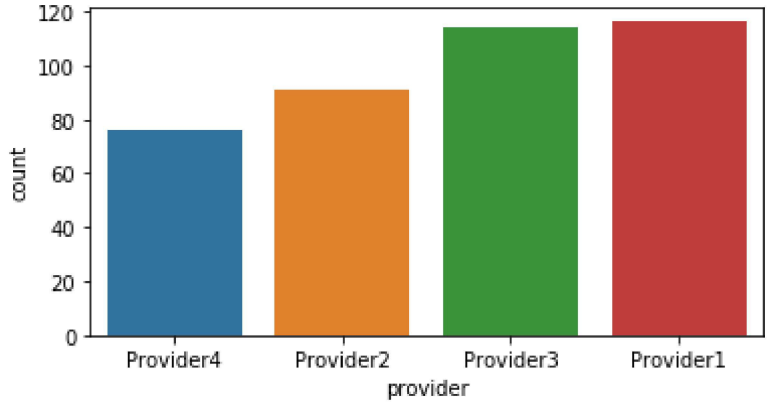
In [24]:
```
plt.figure(figsize=(10,5))
sns.swarmplot(x='provider',y='lifetime',hue='broken',data=df)
plt.show()
```



Above graphs suggests that lifetime of the machine is the phenomenon that affects every Team as well as providers.

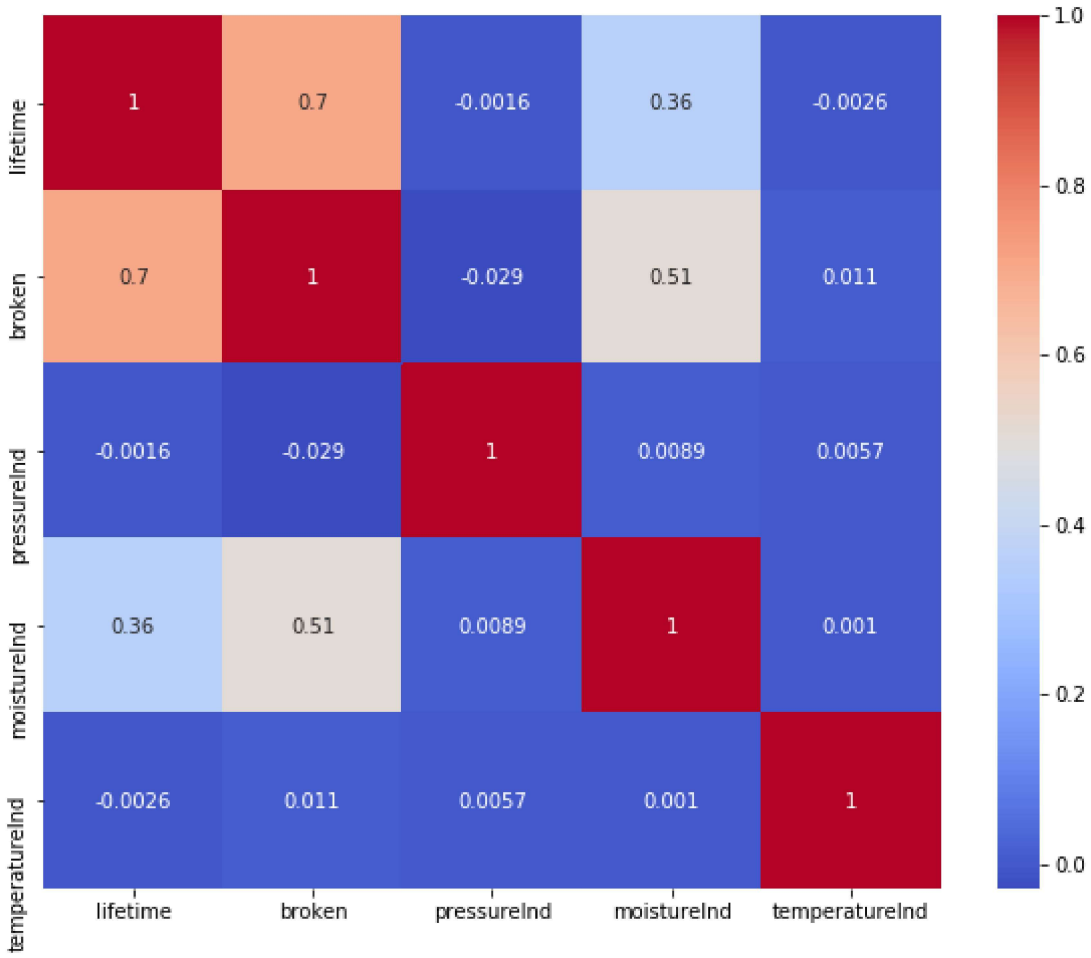
In [25]:
```
plt.figure(figsize=(6,3))
sns.countplot(df['provider'])

plt.show()

plt.figure(figsize=(6,3))
sns.countplot(df['provider'][df.broken==1])
plt.show()
```



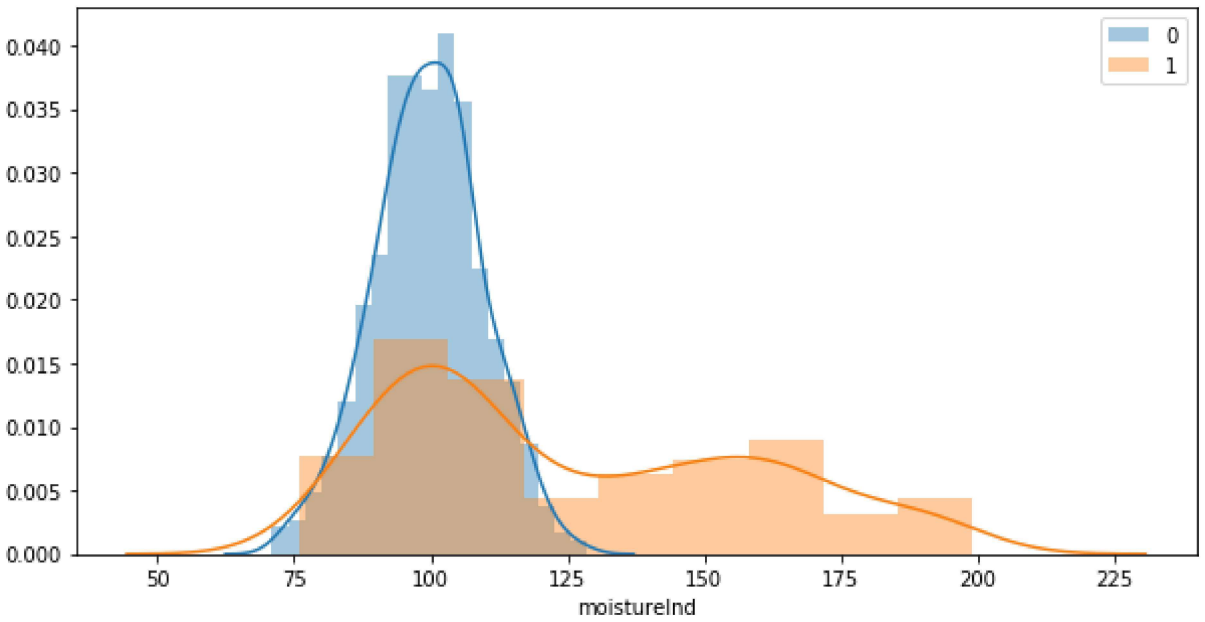


Above graph shows Provider 3 & 4 have most faulty machines.

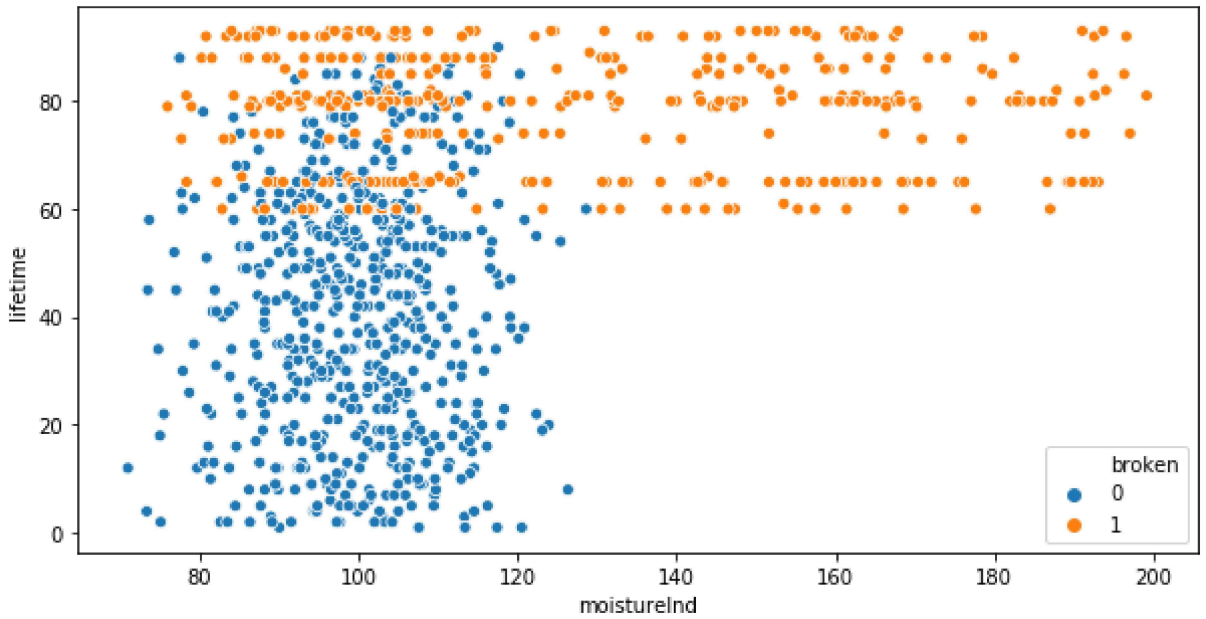
In [26]:
```
cor=df.corr()
plt.figure(figsize=(10,8))
sns.heatmap(cor, annot=True,cmap='coolwarm')
plt.show()
```



In [27]:
```
plt.figure(figsize=(10,5))
sns.distplot(df.moistureInd[df.broken==0])
sns.distplot(df.moistureInd[df.broken==1])
plt.legend(['0','1'])
plt.show()
```



In [28]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(x='moistureInd',y='lifetime',hue='broken',data=df)
plt.show()
```



Above graph suggests that Machines that have lifetime>60 have larger probability of being broken and Machines working at moistureind > 120 have lifetime >60 hence moisture may or may not be the factor affecting machines to be broken.

Conclusion:-

From all above graphs & various analysis, it can be concluded that the lifetime of the Machine is an affecting factor why machines are getting damaged(broken). Its is also logical as the longer the machines have been working and running, higher are the chances of them to have wear and tear. Machines with lifetime >60 have larger probablity of being broken hence adequate measures need to be taken for old/long run machines.