# Why Machines Are Broken ?

In [1]:

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

In [2]:

```
df = pd.read_csv(r"https://raw.githubusercontent.com/anshupandey/Machine_Learning_Training/master/datasets/maintenance_data.csv")
df
```

Out[2]:

| | 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 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 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 [3]:

```
df.size
```

Out[3]:

```
7000
```

In [4]:

```
df.shape
```

Out[4]:

```
(1000, 7)
```

## Data Exploration

In [5]:

```
df.head()
```

Out[5]:

| | 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 |

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
```

In [7]:

```
df.describe()
```

Out[7]:

| | 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 |

## Data Cleaning

In [16]:

```
# Check for duplicates
df.duplicated().sum()
```

Out[16]:

```
0
```

In [10]:

```
# Checking For Missing Values
df.isnull().sum()
```

Out[10]:

```
lifetime          0
broken            0
pressureInd       4
moistureInd       0
temperatureInd    3
team              0
provider          0
dtype: int64
```

In [12]:

```
# Replacing The  missing values with mean

df.pressureInd.fillna(df.pressureInd.mean(), inplace = True)
df.isnull().sum()
```

Out[12]:

```
lifetime          0
broken            0
pressureInd       0
moistureInd       0
temperatureInd    3
team              0
provider          0
dtype: int64
```

In [13]:

```
# Replacing The  missing values with mean

df.temperatureInd.fillna(df.temperatureInd.mean(), inplace = True)
df.isnull().sum()
```

Out[13]:

```
lifetime          0
broken            0
pressureInd       0
moistureInd       0
temperatureInd    0
team              0
provider          0
dtype: int64
```

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.describe()
```

Out[15]:

| | lifetime | broken | pressureInd | moistureInd | temperatureInd |
|---|---|---|---|---|---|
| count | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| mean | 55.195000 | 0.397000 | 98.681100 | 111.088723 | 100.553499 |
| std | 26.472737 | 0.489521 | 19.839864 | 41.839005 | 19.562620 |
| min | 1.000000 | 0.000000 | 33.481917 | 70.928815 | 42.279598 |
| 25% | 34.000000 | 0.000000 | 85.574091 | 94.532547 | 87.676913 |
| 50% | 60.000000 | 0.000000 | 97.514448 | 102.844084 | 100.553499 |
| 75% | 80.000000 | 1.000000 | 112.217466 | 113.532970 | 113.517905 |
| max | 93.000000 | 1.000000 | 173.282541 | 1156.493254 | 172.544140 |

## Data Analytics

### *Univariate Analytics*

In [18]:

```
df.columns
```

Out[18]:

```
Index(['lifetime', 'broken', 'pressureInd', 'moistureInd', 'temperatureIn
d',
       'team', 'provider'],
      dtype='object')
```

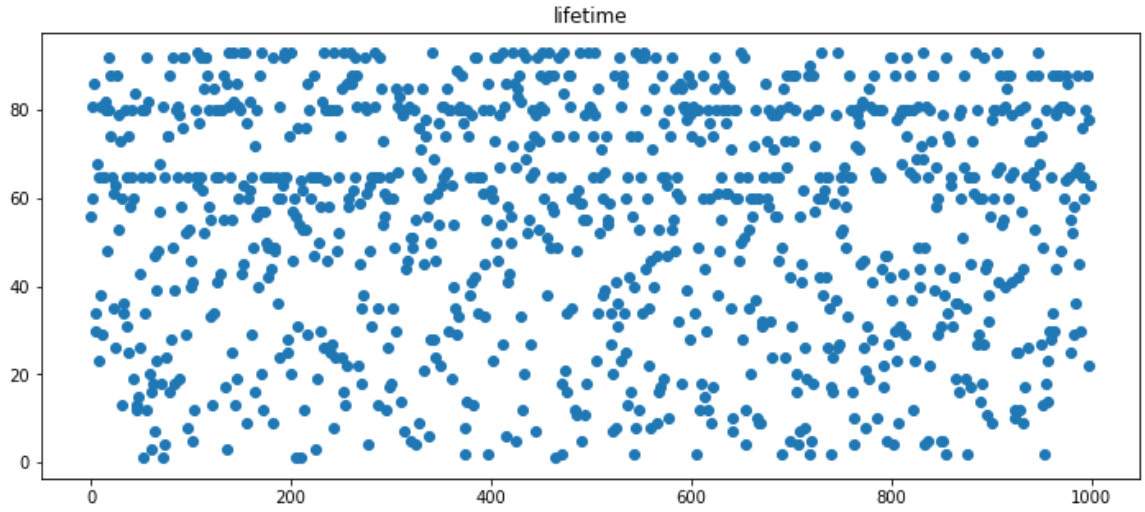
In [19]:

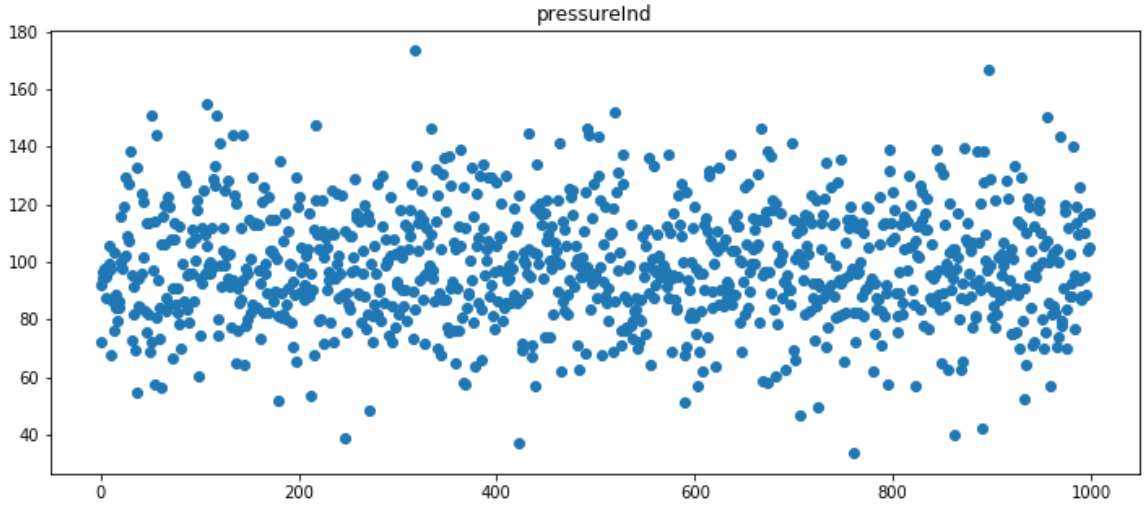
```
num = ['lifetime', 'pressureInd', 'moistureInd', 'temperatureInd']
cats = ['broken', 'team', 'provider']
```

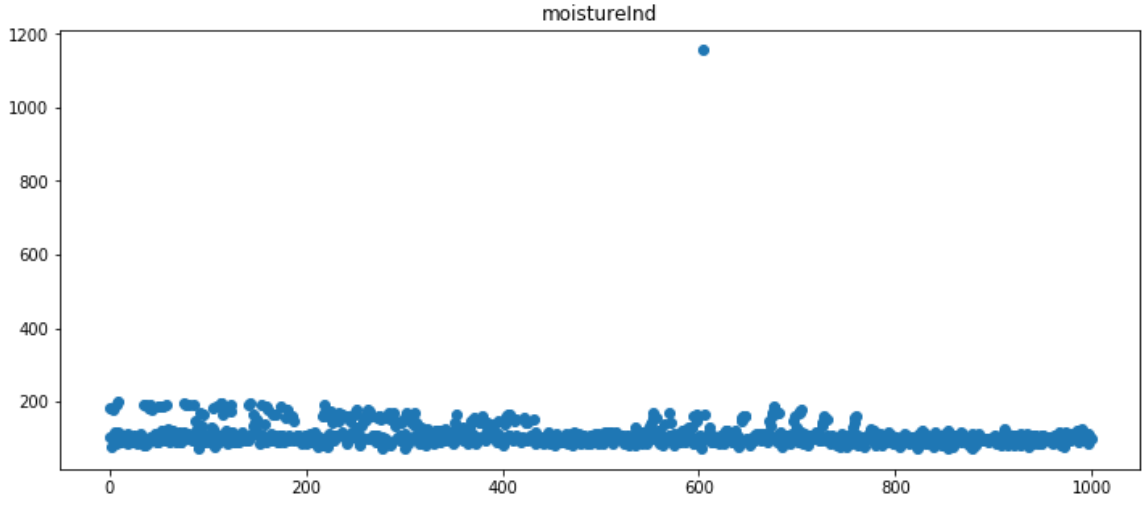
In [20]:

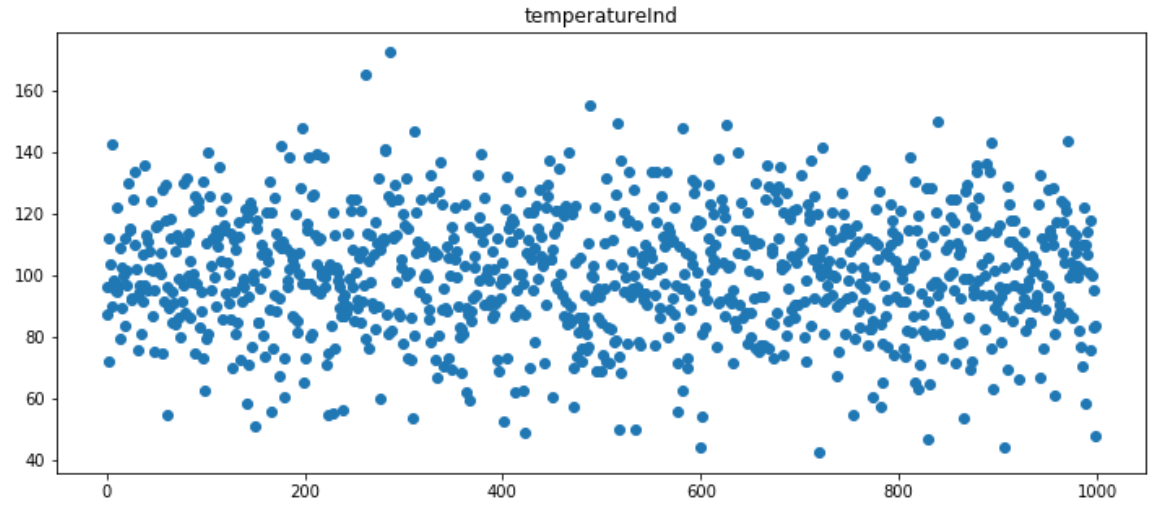
```
# numerics graph

for col in num:
    plt.figure(figsize=(12, 5))
    plt.scatter(np.arange(1000), df[col])
    plt.title(col)
    plt.show()
```


lifetime
0
20
40
60
80
0
200
400
600
800
1000


pressureInd
40
60
80
100
120
140
160
180
0
200
400
600
800
1000

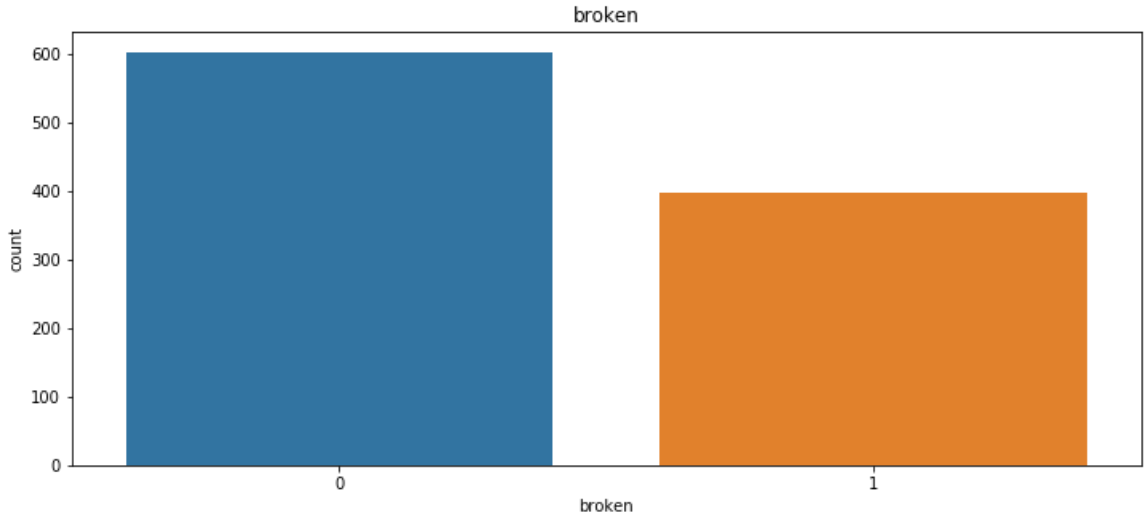

moistureInd
200
400
600
800
1000
1200
0
200
400
600
800
1000

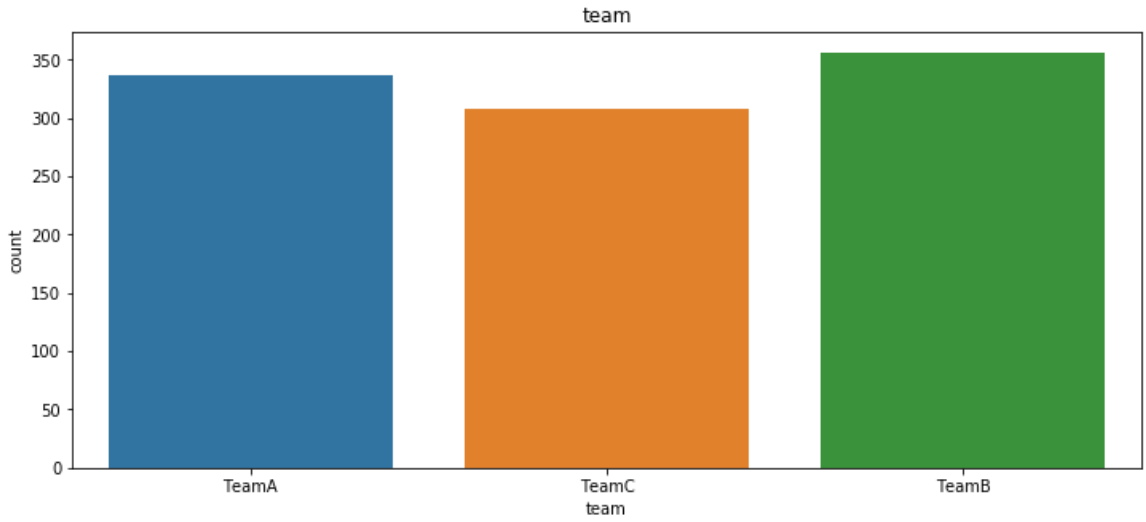

temperatureInd
160
140
120
100
80
60
40
0
200
400
600
800
1000

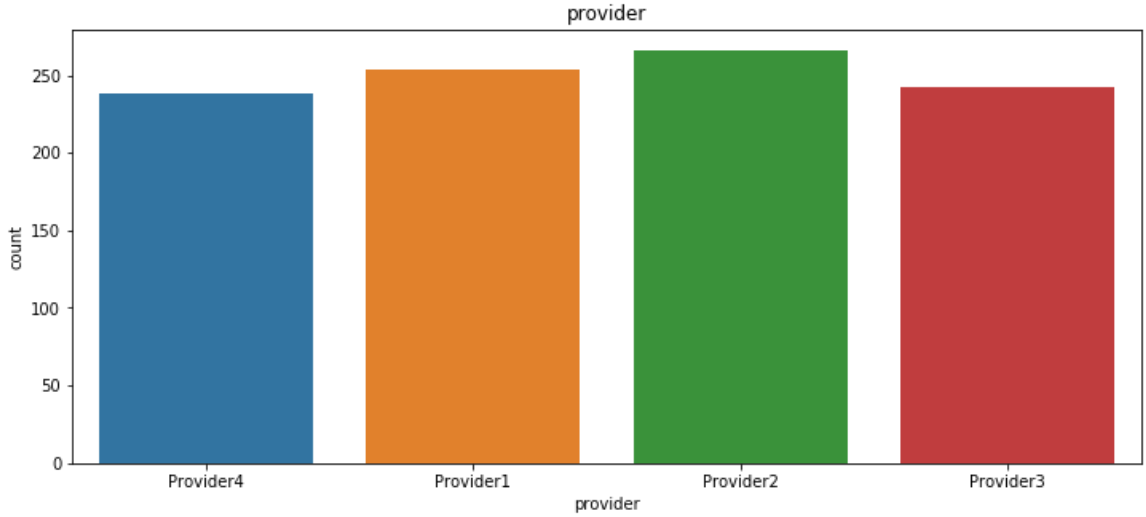
In [22]:

```
# categorical graph

for col in cats:
    plt.figure(figsize=(12, 5))
    sns.countplot(df[col])
    plt.title(col)
    plt.show()
```


broken
600
500
400
300
200
100
0
count
0
1
broken

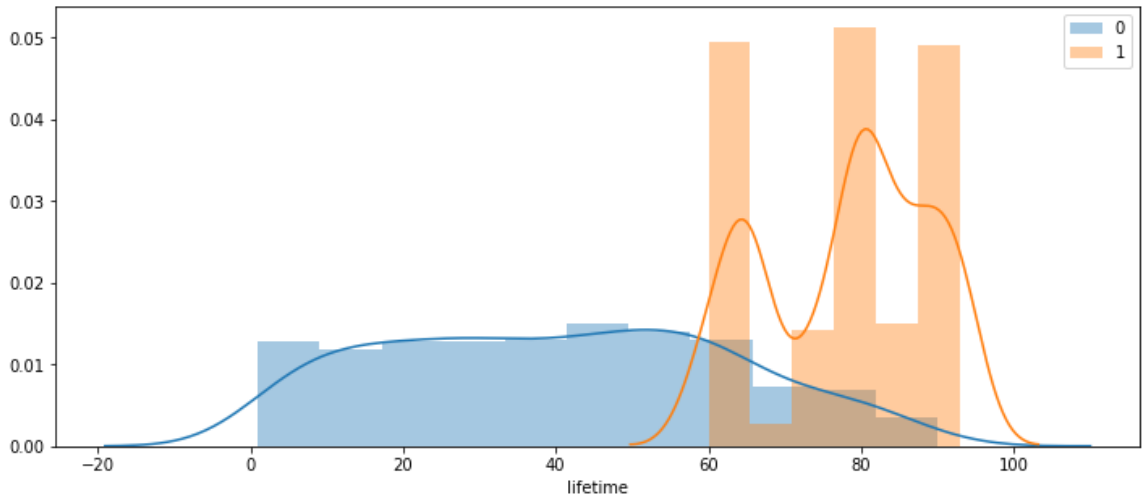

team
350
300
250
200
150
100
50
0
count
TeamA
TeamC
TeamB
team




## Bivariate Analysis

In [26]:

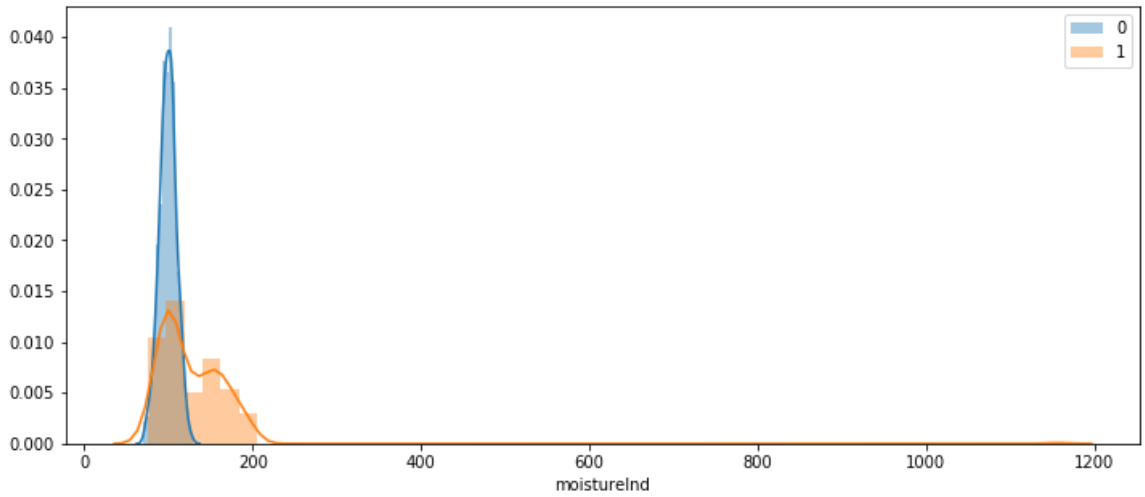
```
# numerical vs categorical
# lifetime vs broken
# Probablity density distribution
plt.figure(figsize = (12, 5))
sns.distplot(df.lifetime[df.broken==0])
sns.distplot(df.lifetime[df.broken==1])
plt.legend(['0', '1'])
plt.show()
```




NOTE: The graph shows that the lifetime of the machine more than 60 months are broken then the machine whose lifetime is less than 60. The reason for that being the machine gets older with time need proper maintenance after certain point.

In [27]:

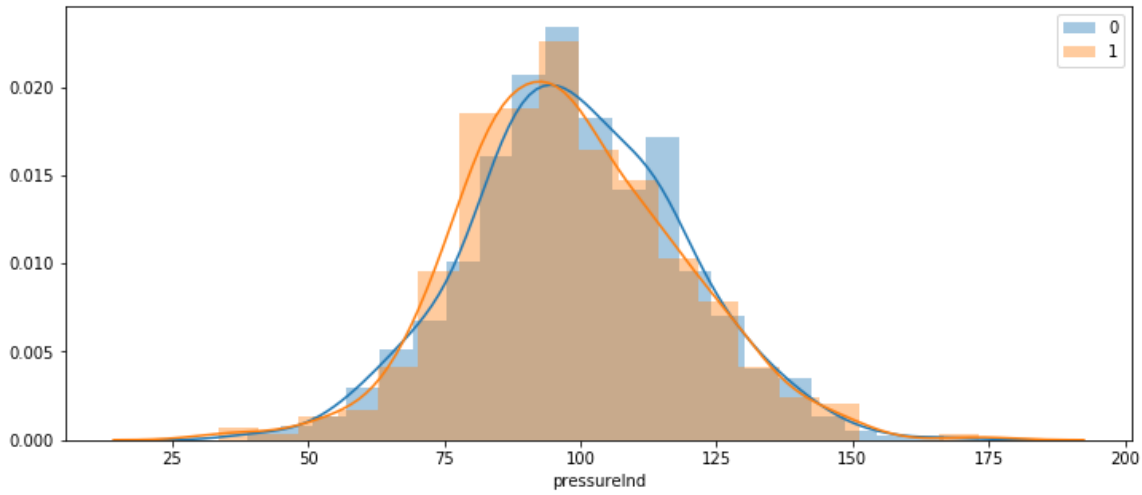
```
# numerical vs categorical
# moistureInd vs broken
# Probablity density distribution
plt.figure(figsize = (12, 5))
sns.distplot(df.moistureInd[df.broken==0])
sns.distplot(df.moistureInd[df.broken==1])
plt.legend(['0', '1'])
plt.show()
```




NOTE: Machines with moisture index less than 100 has less chance of being broken, then the machine with moisture index more than 100 but less than 200. So machine with moisture index more than 100 is broken. More the moisture index , more machine gets rusty that damages the machine.

In [28]:

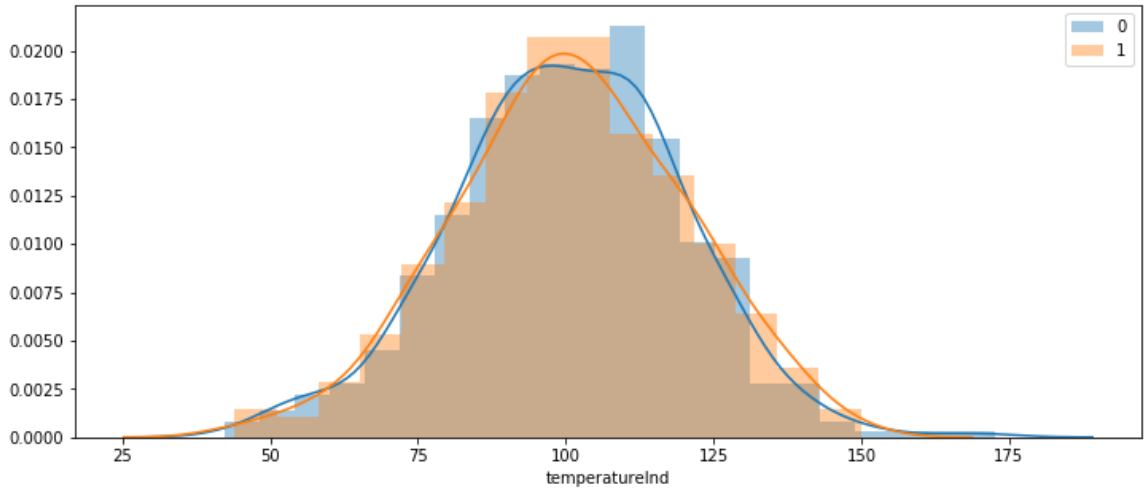
```
# numerical vs categorical
# pressureInd vs broken
# Probablity density distribution
plt.figure(figsize = (12, 5))
sns.distplot(df.pressureInd[df.broken==0])
sns.distplot(df.pressureInd[df.broken==1])
plt.legend(['0', '1'])
plt.show()
```



NOTE: The probablity density plot for pressure index and machine being broken is almost overlapping with the machine not being broken. Therefore , pressure index doesn't have sufficient information to say why machine is being broken.

In [29]:

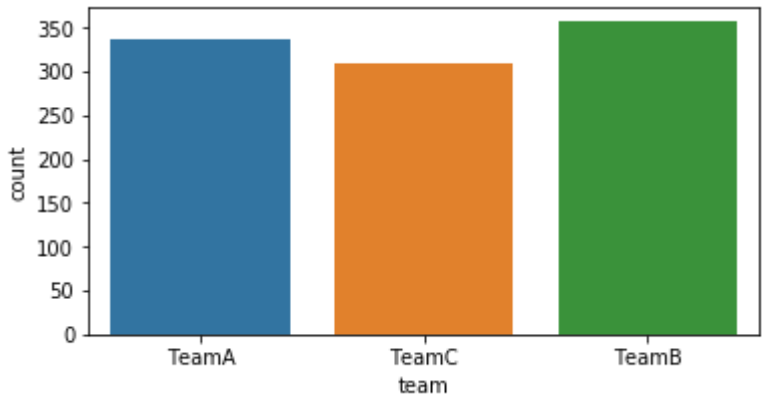
```
# numerical vs categorical
# Probablity density distribution
plt.figure(figsize=(12, 5))
sns.distplot(df.temperatureInd[df.broken==0])
sns.distplot(df.temperatureInd[df.broken==1])
plt.legend(['0', '1'])
plt.show()
```

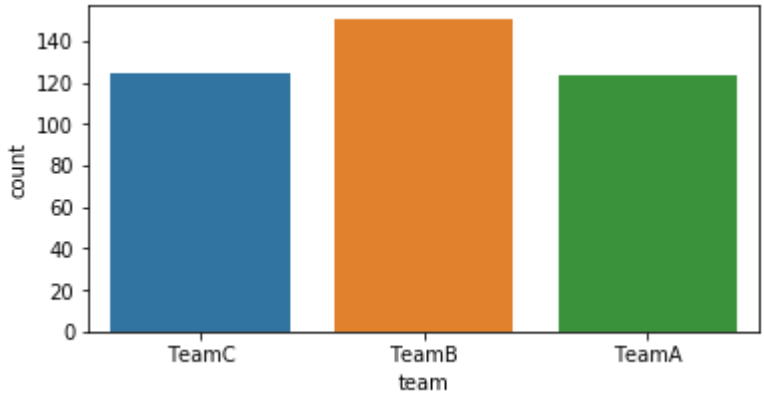



NOTE: The probablity density plot for temperature index and machine being broken is almost overlapping with the machine not being broken. Therefore , temperature index doesn't have sufficient information to say why machine is being broken

In [36]:

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

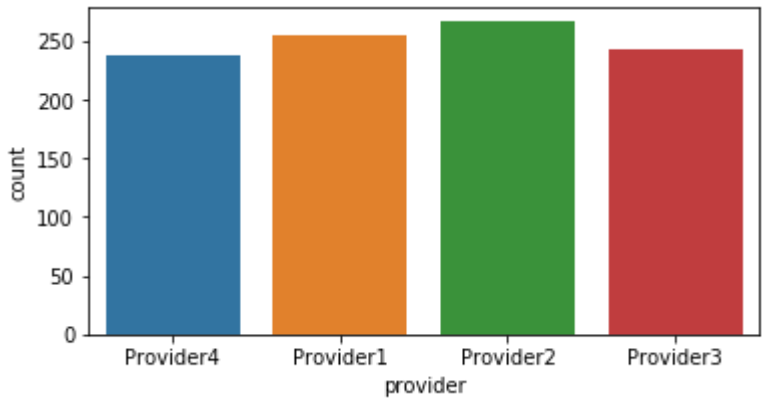


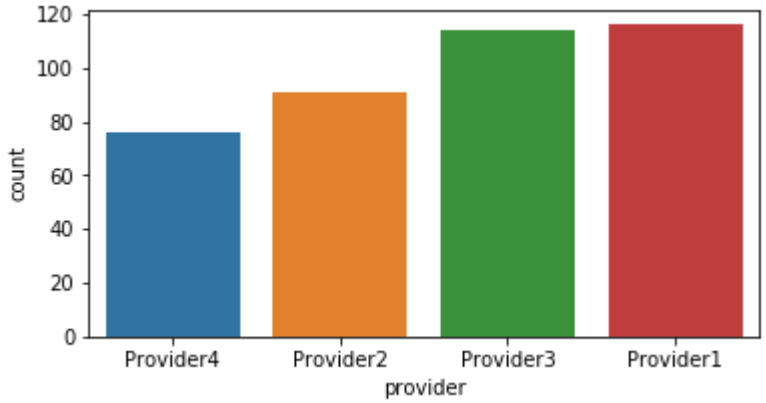




NOTE : Different Team doesn't give much information about the machine being broken.

In [37]:

```
# categorical vs categorical
# provider vs broken
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()
```



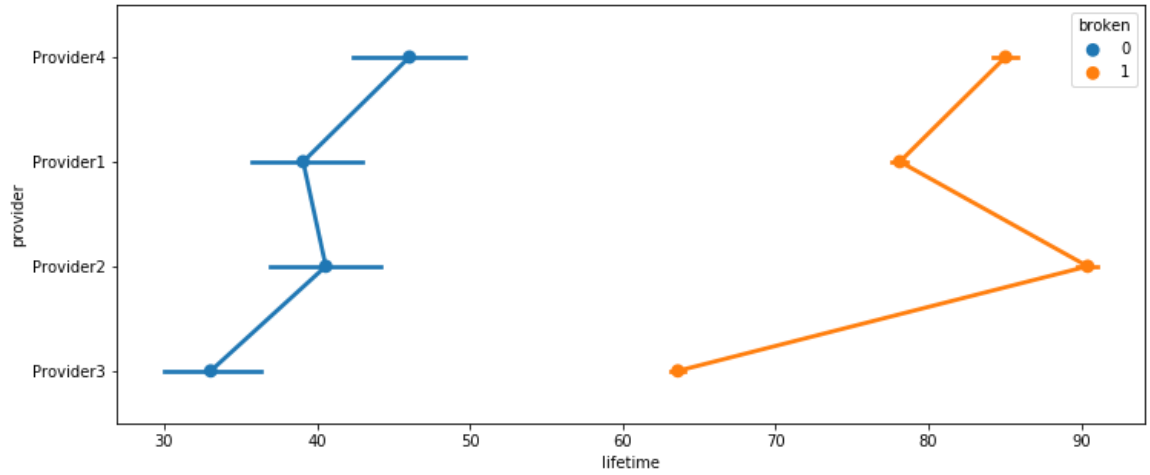




NOTE: Provider 1 and Provider 3 has more broken rate then the other two providers.

## MultiVariate Analysis

In [38]:

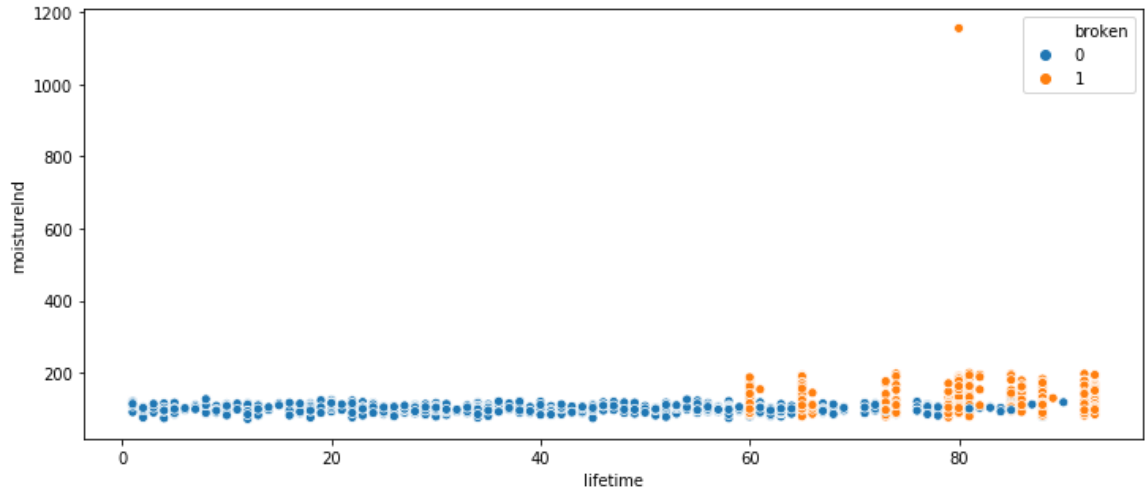
```
# numerical vs categorical vs categorical - point plot
# lifetime vs provider vs broken
plt.figure(figsize = (12, 5))
sns.pointplot(x = 'lifetime', y = 'provider', hue = 'broken', data = df)
plt.show()
```




NOTE : Lifetime of Provider 2 and Provider 4 is as compared to Provider 1 and Provider 3.

In [39]:

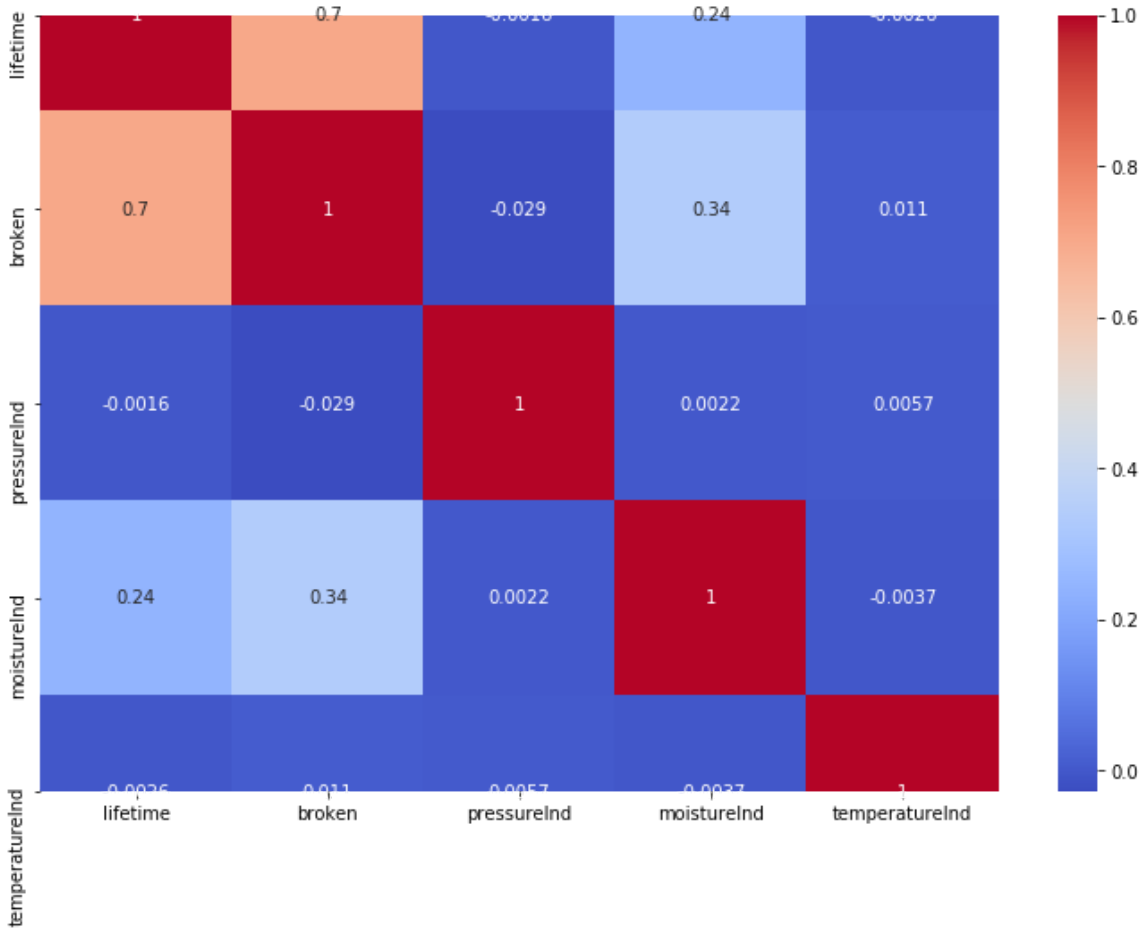
```
# numerical vs numerical vs categorical - scatter plot
# lifetime vs moistureIND vs broken
plt.figure(figsize = (12, 5))
sns.scatterplot(x = 'lifetime', y = 'moistureInd', hue = 'broken', data = df)
plt.show()
```




NOTE : Lifetime of more than 70 months and mosisture index less than 200 has more chance of machine being broken.

In [40]:

```
cor = df.corr() # clculating correlation matrix
# plotting correlation using heatmap
# cor>0.5 v.good
# cor<0.5 and >0.1 good
# cor<-0.5 v.good
# cor>-0.5 and <-0.1 good
# -0.1 to +0.1 ~0 bad
plt.figure(figsize=(12, 8))
sns.heatmap(cor, annot = True, cmap = 'coolwarm')
plt.show()
```




In [ ]: