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

About the Data

- UTC: Time Stamp
- Temperature: Air Temperature
- Humidity%: Air Humidity
- TVOC[ppb]: Total Volatile Organic Compounds; measured in parts per billion
- eCO2[ppm]: co2 equivalent concentration; calculated from different values like TVCO
- Raw H2: raw molecular hydrogen; not compensated (Bias, temperature, etc.)
- Raw Ethanol: raw ethanol gas
- Pressure[hPa]: Air Pressure
- PM1.0: particulate matter size < 1.0 µm (PM1.0). 1.0 µm < 2.5 µm (PM2.5)
- PM2.5: particulate matter size < 1.0 µm (PM1.0). 1.0 µm < 2.5 µm (PM2.5)
- NC0.5: Number concentration of particulate matter. This differs from PM because NC gives the actual number of particles in the air
- NC1.0: Number concentration of particulate matter. This differs from PM because NC gives the actual number of particles in the air
- Number concentration of particulate matter. This differs from PM because NC gives the actual number of particles in the air
- CNT: Sample counter.
- Fire Alarm: Ground truth is "1" if a fire is there.

```
df = pd.read_csv('/content/sample_data/smoke_detection_iot.csv', index_col= 0)
df.head()
```

	UTC	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	NC0.5	NC1.0	NC2.5
0	1654733331	20.000	57.36	0	400	12306	18520	939.735	0.0	0.0	0.0	0.0	0.0
1	1654733332	20.015	56.67	0	400	12345	18651	939.744	0.0	0.0	0.0	0.0	0.0
2	1654733333	20.029	55.96	0	400	12374	18764	939.738	0.0	0.0	0.0	0.0	0.0
3	1654733334	20.044	55.28	0	400	12390	18849	939.736	0.0	0.0	0.0	0.0	0.0
4	1654733335	20.059	54.69	0	400	12403	18921	939.744	0.0	0.0	0.0	0.0	0.0

Next steps: Generate code with df View recommended plots New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 62630 entries, 0 to 62629
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   UTC             62630 non-null  int64
 1   Temperature[C]  62630 non-null  float64
 2   Humidity[%]     62630 non-null  float64
 3   TVOC[ppb]       62630 non-null  int64
 4   eCO2[ppm]       62630 non-null  int64
 5   Raw H2          62630 non-null  int64
 6   Raw Ethanol     62630 non-null  int64
 7   Pressure[hPa]   62630 non-null  float64
 8   PM1.0           62630 non-null  float64
 9   PM2.5           62630 non-null  float64
 10  NC0.5           62630 non-null  float64
 11  NC1.0           62630 non-null  float64
 12  NC2.5           62630 non-null  float64
 13  CNT             62630 non-null  int64
 14  Fire Alarm      62630 non-null  int64
dtypes: float64(8), int64(7)
memory usage: 7.6 MB
```

```
df.describe()
```

	UTC	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0
count	6.263000e+04	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000
mean	1.654792e+09	15.970424	48.539499	1942.057528	670.021044	12942.453936	19754.257912	938.627649	100.594309
std	1.100025e+05	14.359576	8.865367	7811.589055	1905.885439	272.464305	609.513156	1.331344	922.524245
min	1.654712e+09	-22.010000	10.740000	0.000000	400.000000	10668.000000	15317.000000	930.852000	0.000000
25%	1.654743e+09	10.994250	47.530000	130.000000	400.000000	12830.000000	19435.000000	938.700000	1.280000
50%	1.654762e+09	20.130000	50.150000	981.000000	400.000000	12924.000000	19501.000000	938.816000	1.810000
75%	1.654778e+09	25.409500	53.240000	1189.000000	438.000000	13109.000000	20078.000000	939.418000	2.090000
max	1.655130e+09	59.930000	75.200000	60000.000000	60000.000000	13803.000000	21410.000000	939.861000	14333.690000

```
df['UTC'] = pd.to_datetime(df['UTC'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 62630 entries, 0 to 62629
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   UTC             62630 non-null  datetime64[ns]
 1   Temperature[C]  62630 non-null  float64
 2   Humidity[%]     62630 non-null  float64
 3   TVOC[ppb]       62630 non-null  int64
 4   eCO2[ppm]       62630 non-null  int64
 5   Raw H2          62630 non-null  int64
 6   Raw Ethanol     62630 non-null  int64
 7   Pressure[hPa]   62630 non-null  float64
 8   PM1.0           62630 non-null  float64
 9   PM2.5           62630 non-null  float64
 10  NC0.5           62630 non-null  float64
 11  NC1.0           62630 non-null  float64
 12  NC2.5           62630 non-null  float64
 13  CNT             62630 non-null  int64
 14  Fire Alarm      62630 non-null  int64
dtypes: datetime64[ns](1), float64(8), int64(6)
memory usage: 7.6 MB
```

```
df.head()
```

	UTC	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	NC0.5	NC1.0	N
0	1970-01-01 00:00:01.654733331	20.000	57.36	0	400	12306	18520	939.735	0.0	0.0	0.0	0.0	
1	1970-01-01 00:00:01.654733332	20.015	56.67	0	400	12345	18651	939.744	0.0	0.0	0.0	0.0	
2	1970-01-01 00:00:01.654733333	20.029	55.96	0	400	12374	18764	939.738	0.0	0.0	0.0	0.0	
3	1970-01-01 00:00:01.654733334	20.044	55.28	0	400	12390	18849	939.736	0.0	0.0	0.0	0.0	
4	1970-01-01 00:00:01.654733335	20.059	54.69	0	400	12403	18921	939.744	0.0	0.0	0.0	0.0	

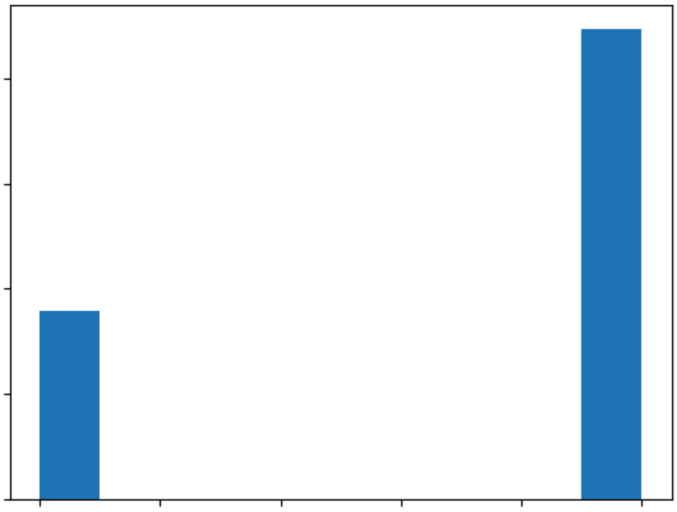
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.tail()
```

	UTC	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	NC0.5	NC1.
62625	1970-01-01 00:00:01.655130047	18.438	15.79	625	400	13723	20569	936.670	0.63	0.65	4.32	0.67
62626	1970-01-01 00:00:01.655130048	18.653	15.87	612	400	13731	20588	936.678	0.61	0.63	4.18	0.65
62627	1970-01-01 00:00:01.655130049	18.867	15.84	627	400	13725	20582	936.687	0.57	0.60	3.95	0.61
62628	1970-01-01 00:00:01.655130050	19.083	16.04	638	400	13712	20566	936.680	0.57	0.59	3.92	0.61
62629	1970-01-01 00:00:01.655130051	19.299	16.52	643	400	13696	20543	936.676	0.57	0.59	3.90	0.60

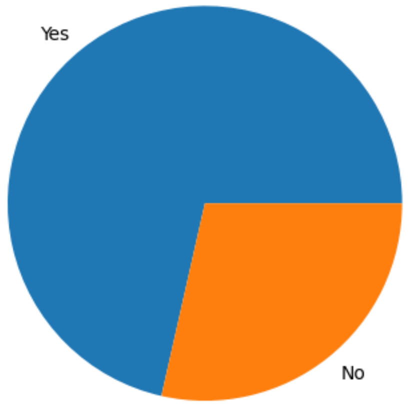
```
# sns.histplot(df['Fire Alarm'])
plt.hist(df['Fire Alarm'])
plt.xlabel('Fire Alarm', color= 'white', fontsize = 18, weight = 'bold')
plt.ylabel('Count', color= 'white', fontsize = 18, weight = 'bold')
plt.xticks(color = 'white', fontsize = 14)
plt.yticks(color = 'white', fontsize = 14)
```

```
(array([    0., 10000., 20000., 30000., 40000., 50000.]),
 [Text(0, 0.0, '0'),
  Text(0, 10000.0, '10000'),
  Text(0, 20000.0, '20000'),
  Text(0, 30000.0, '30000'),
  Text(0, 40000.0, '40000'),
  Text(0, 50000.0, '50000')])
```



```
x = df['Fire Alarm'].value_counts()
plt.pie(x, labels=['Yes', 'No'])
```

```
([<matplotlib.patches.Wedge at 0x7d5851b27520>,
  <matplotlib.patches.Wedge at 0x7d5851b27430>],
 [Text(-0.6867566913298193, 0.8592818204254757, 'Yes'),
  Text(0.6867567717815868, -0.8592817561266699, 'No')])
```



Temperature affect on Fire

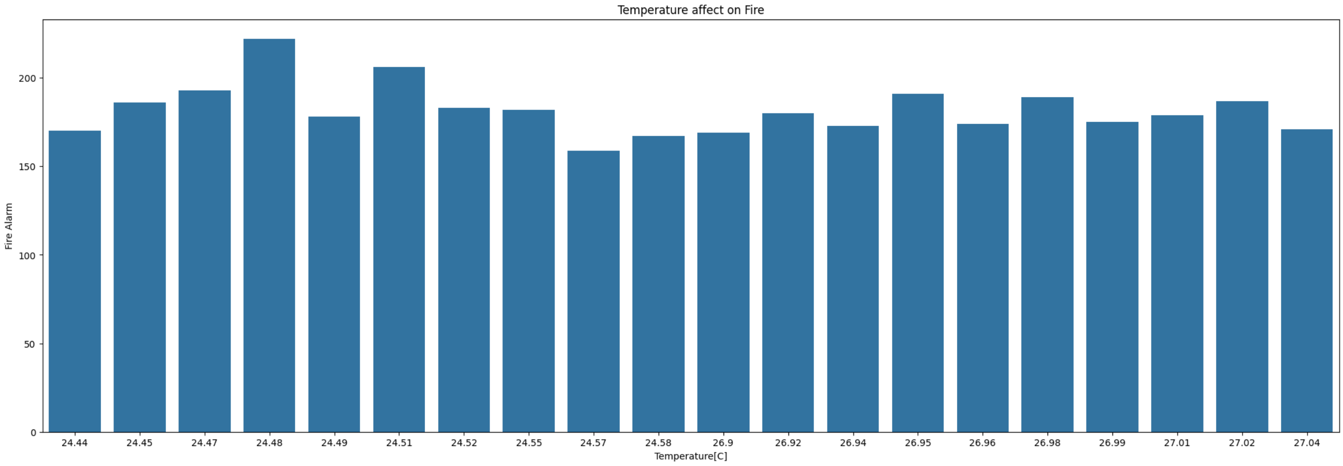
```
temperature_fire = df.groupby('Temperature[C]')[['Fire Alarm']].count().sort_values(by='Fire Alarm',ascending = False).head(20)
temperature_fire
```

	Fire Alarm
Temperature[C]	
24.48	222
24.51	206
24.47	193
26.95	191
26.98	189
27.02	187
24.45	186
24.52	183
24.55	182
26.92	180
27.01	179
24.49	178
26.99	175
26.96	174
26.94	173
27.04	171
24.44	170
26.90	169
24.58	167
24.57	159

Next steps: Generate code with `temperature_fire` View recommended plots New interactive sheet

```
plt.figure(figsize=[25,8])
sns.barplot(x=temperature_fire.index,y=temperature_fire['Fire Alarm'])
plt.title('Temperature affect on Fire')
```

```
Text(0.5, 1.0, 'Temperature affect on Fire')
```



This graph shows that high temperatures don't have a direct affect on fires

Humidity affect on fires

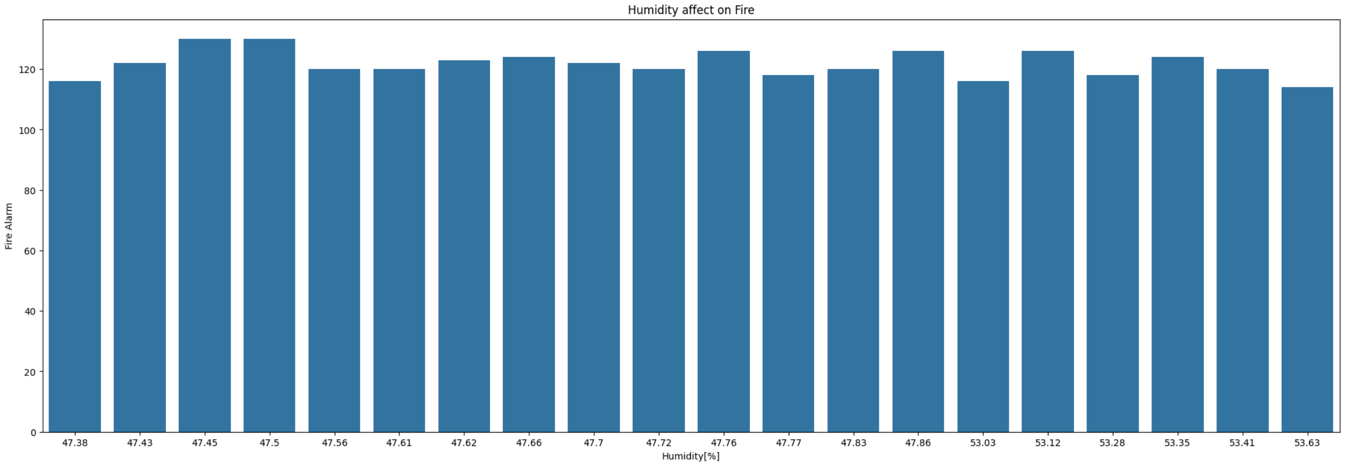
```
humidity_fire = df.groupby('Humidity[%]')[['Fire Alarm']].count().sort_values(by='Fire Alarm',ascending = False).head(20)
humidity_fire
```

	Fire Alarm
Humidity[%]	
47.45	130
47.50	130
47.86	126
53.12	126
47.76	126
47.66	124
53.35	124
47.62	123
47.70	122
47.43	122
47.56	120
47.72	120
47.83	120
47.61	120
53.41	120
53.28	118
47.77	118
53.03	116
47.38	116
53.63	114

Next steps: Generate code with humidity_fire View recommended plots New interactive sheet

```
plt.figure(figsize=[25,8])
sns.barplot(x=humidity_fire.index, y=humidity_fire['Fire Alarm']) # Added x= and y=
plt.title('Humidity affect on Fire')
```

```
Text(0.5, 1.0, 'Humidity affect on Fire')
```



This graph shows that high humidity doesn't have a direct affect on fires

Pressure affect on fires

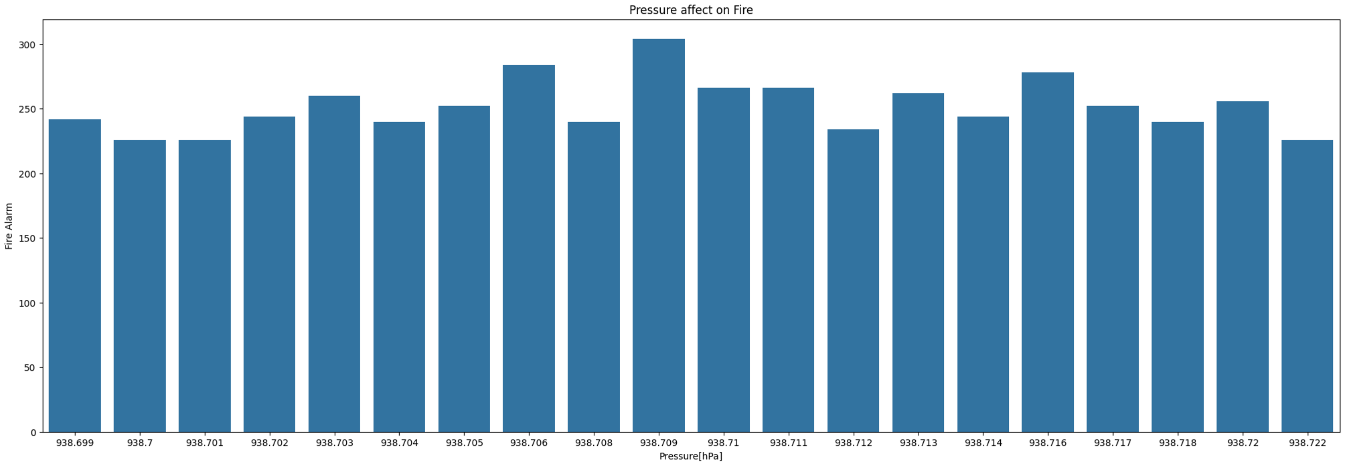
```
pressure_fire = df.groupby('Pressure[hPa]')[['Fire Alarm']].count().sort_values(by='Fire Alarm',ascending = False).head(20)
pressure_fire
```

	Fire Alarm
Pressure[hPa]	
938.709	304
938.706	284
938.716	278
938.711	266
938.710	266
938.713	262
938.703	260
938.720	256
938.717	252
938.705	252
938.702	244
938.714	244
938.699	242
938.718	240
938.704	240
938.708	240
938.712	234
938.701	226
938.700	226
938.722	226

Next steps: Generate code with pressure_fire View recommended plots New interactive sheet

```
plt.figure(figsize=[25,8])
sns.barplot(x=pressure_fire.index, y=pressure_fire['Fire Alarm']) # Added x= and y=
plt.title('Pressure affect on Fire')
```

```
Text(0.5, 1.0, 'Pressure affect on Fire')
```



Raw Ethanol affect on fires

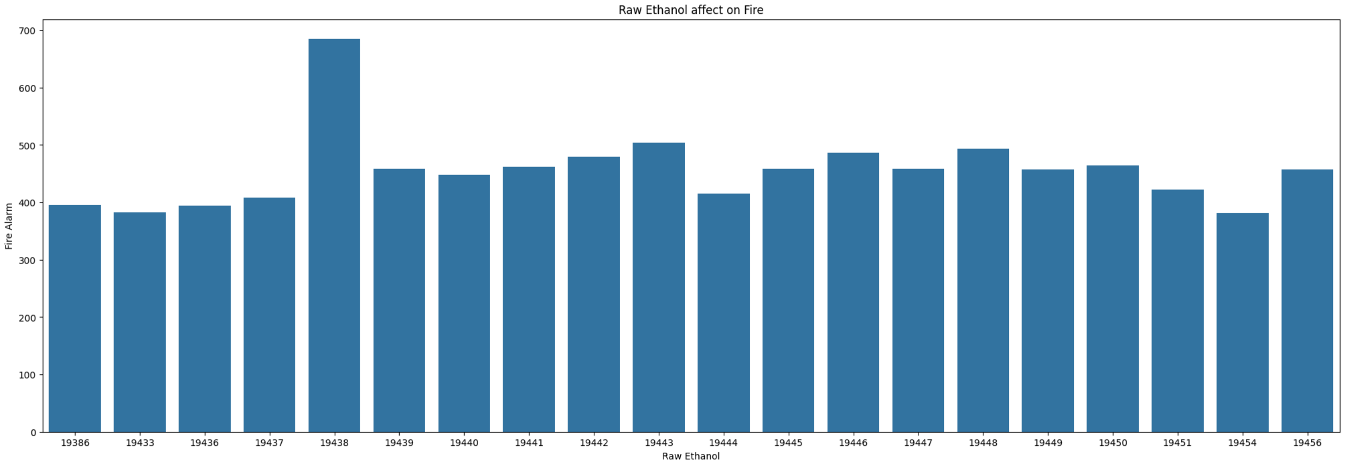
```
ethanol_fire = df.groupby('Raw Ethanol')[['Fire Alarm']].count().sort_values(by='Fire Alarm',ascending = False).head(20)
ethanol_fire
```

	Fire Alarm
Raw Ethanol	
19438	685
19443	504
19448	494
19446	486
19442	479
19450	464
19441	462
19445	459
19447	458
19439	458
19456	457
19449	457
19440	448
19451	422
19444	415
19437	408
19386	395
19436	394
19433	383
19454	382

Next steps: Generate code with ethanol_fire View recommended plots New interactive sheet

```
plt.figure(figsize=[25,8])
sns.barplot(x=ethanol_fire.index, y=ethanol_fire['Fire Alarm']) # Use x and y parameters to specify columns
plt.title('Raw Ethanol affect on Fire')
```

```
Text(0.5, 1.0, 'Raw Ethanol affect on Fire')
```

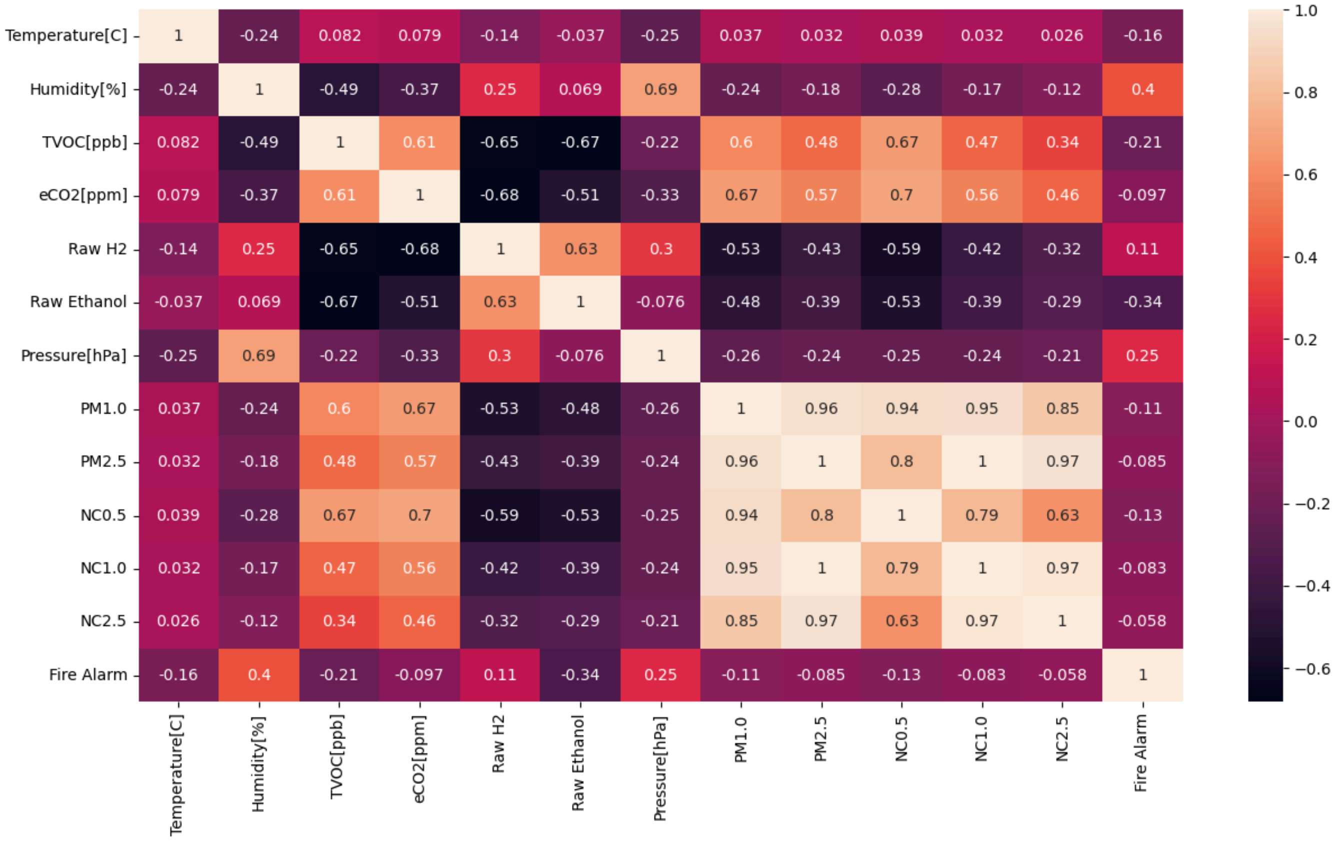


It seems that fires are higher when the Raw Ethanol value is 19438

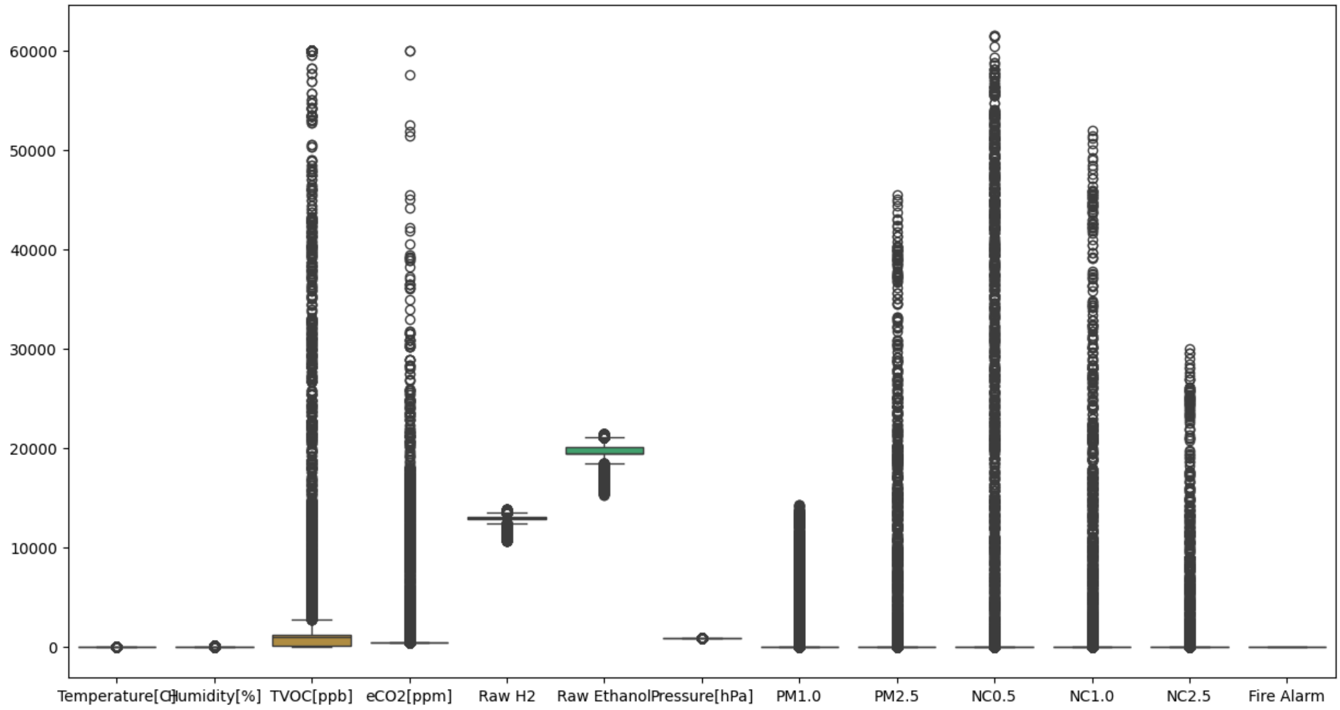
Correlation between the Parameters

```
plt.figure(figsize=[15,8])
sns.heatmap(df.corr(),annot = True)
```

```
<Axes: >
```



```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
plt.figure(figsize=[15,8])
sns.boxplot(data=df)
plt.show()
```



Handle Outliers

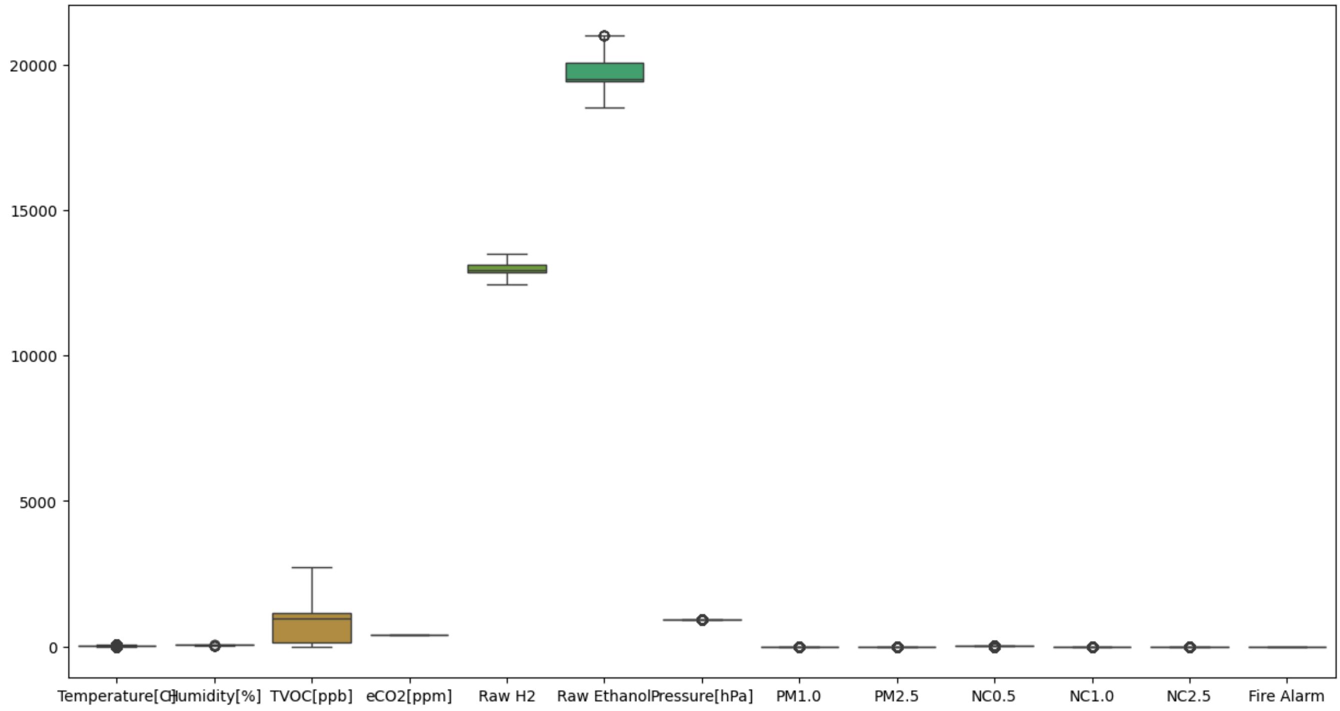
```
from datasist.structdata import detect_outliers
index = detect_outliers(df,0,['Humidity[%]','TVOC[ppb]','eCO2[ppm]', 'Raw H2', 'Raw Ethanol', 'PM1.0', 'PM2.5', 'NC0.5', 'NC1.0', 'NC2.5
len(index)
```

```
17492
```

```
for col in ['Humidity[%]', 'TVOC[ppb]', 'eCO2[ppm]', 'Raw H2', 'Raw Ethanol', 'PM1.0', 'PM2.5', 'NC0.5', 'NC1.0', 'NC2.5']:

    outliers_indices = detect_outliers(df, 0, [col])


    col_median = df[col].median()


    df.loc[outliers_indices, col] = col_median
```

```
plt.figure(figsize=[15,8])
sns.boxplot(data=df)
```

```
<Axes: >
```



Split input data and output data

```
X = df.drop(columns='Fire Alarm')
y = df['Fire Alarm']
```

Split data into Train and Test

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size= 0.2, random_state= 18, stratify= y)
```

```
y_train.value_counts()
```

```
            count
Fire Alarm
     1      35806
     0      14298

dtype: int64
```

```
y_test.value_counts()
```

```
            count
Fire Alarm
     1       8951
     0       3575

dtype: int64
```

```
x_train.duplicated().sum()
```

```
7
```

Handle Imbalance using SMOTE Over Sampling

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
```

```
y_train_smote.value_counts()
```

Fire Alarm	count
0	35806
1	35806

dtype: int64

```
x_train_smote.duplicated().sum()
```

```
11
```

```
sns.histplot(y_train_smote)
plt.xlabel('Fire Alarm', color= 'white', fontsize = 18, weight = 'bold')
plt.ylabel('Count', color= 'white', fontsize = 18, weight = 'bold')
plt.xticks(color = 'white', fontsize = 14)
plt.yticks(color = 'white', fontsize = 14)
```

```
(array([    0.,  5000., 10000., 15000., 20000., 25000., 30000., 35000.,
        40000.]),
 [Text(0, 0.0, '0'),
  Text(0, 5000.0, '5000'),
  Text(0, 10000.0, '10000'),
  Text(0, 15000.0, '15000'),
  Text(0, 20000.0, '20000'),
  Text(0, 25000.0, '25000'),
  Text(0, 30000.0, '30000'),
  Text(0, 35000.0, '35000'),
  Text(0, 40000.0, '40000')])
```