1.Try to Define Business Problem and Metrics then Machine Learning Metrics

Business Problem:

Significant Customer Complaints towards low performances of Conventional Smoke Detector in detect the true fire trigger Sales softening trend for Months.

Business Metrics:

Customer Satisfaction Survey Score towards performances of Smoke Detector Product

AI Solution:

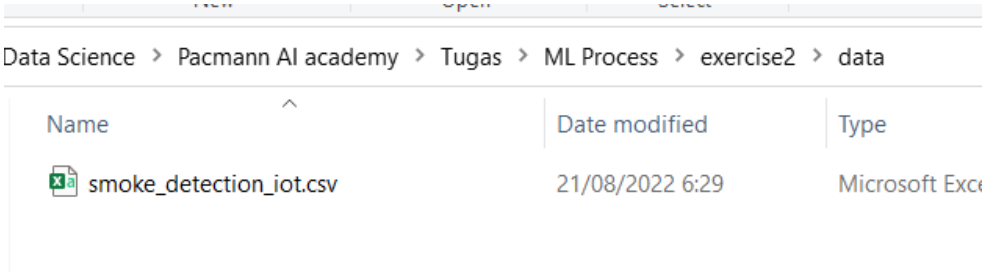
AI based Smoke Detector

Machine Learning Metrics

F1 Score of AI based Smoke Detector

2. Try Data pipelines , Especially definition; Validation;Defense;Splitting; and Serialization

Data Collection:



Data Definition:

[18]:
```
# Read Data
DATA_PATH = "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/

TARGET_COLUMN = "Fire Alarm"
INDEX_COLUMN = "Unnamed: 0"
TEST_SIZE = 0.2

data_house = read_data(DATA_PATH, set_index = INDEX_COLUMN)

data_house.head()
```

[18]:

)C[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	NC0.5	NC1.0	NC2.5	CNT	Fire Alarm
0	400	12306	18520	939.735	0.0	0.0	0.0	0.0	0.0	0	0
0	400	12345	18651	939.744	0.0	0.0	0.0	0.0	0.0	1	0
0	400	12374	18764	939.738	0.0	0.0	0.0	0.0	0.0	2	0
0	400	12390	18849	939.736	0.0	0.0	0.0	0.0	0.0	3	0
0	400	12403	18921	939.744	0.0	0.0	0.0	0.0	0.0	4	0

datadescription.txt - Notepad

File Edit Format View Help

```
UTC
Timestamp UTC seconds

Temperature[C]
Air Temperature

Humidity[%]
Air Humidity

TVOC[ppb]
otal 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)
```

Data Validation

All data is valid and No Null identified

[23]:
```
#Data Validation

print(data_house.info())
print(f'data shape:{data_house.shape}')
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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
None
data shape:(62630, 15)
```

[27]:
```
data_house.describe()
```

[27]:

	UTC	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Eth
count	6.263000e+04	62630.000000	62630.000000	62630.000000	62630.000000	62630.000000	62630.000
mean	1.654792e+09	15.970424	48.539499	1942.057528	670.021044	12942.453936	19754.25
std	1.100025e+05	14.359576	8.865367	7811.589055	1905.885439	272.464305	609.51
min	1.654712e+09	-22.010000	10.740000	0.000000	400.000000	10668.000000	15317.000
25%	1.654743e+09	10.994250	47.530000	130.000000	400.000000	12830.000000	19435.000
50%	1.654762e+09	20.130000	50.150000	981.000000	400.000000	12924.000000	19501.000
75%	1.654778e+09	25.409500	53.240000	1189.000000	438.000000	13109.000000	20078.000
max	1.655130e+09	59.930000	75.200000	60000.000000	60000.000000	13803.000000	21410.000

Data Defense

[36]:
```
#Data Defense

params = {
 'float_columns': [ 'Temperature[C]','Humidity[%]','Pressure[hPa]', 'PM1.0', 'PM2.5', 'N
 'int32_columns': ['UTC','TVOC[ppb]', 'eCO2[ppm]','Raw H2', 'Raw Ethanol', 'CNT', 'Fire
 }

print(params)
```

```
{'float_columns': ['Temperature[C]', 'Humidity[%]', 'Pressure[hPa]', 'PM1.0', 'PM2.5',
'NC0.5', 'NC1.0', 'NC2.5'], 'int32_columns': ['UTC', 'TVOC[ppb]', 'eCO2[ppm]', 'Raw H
2', 'Raw Ethanol', 'CNT', 'Fire Alarm']}
```

[34]:
```
def check_data(input_data, params):
    # check data types
    assert input_data.select_dtypes("float").columns.to_list() == params["float_columns"
    assert input_data.select_dtypes("int").columns.to_list() == params["int32_columns"].
```

[35]:
```
# jika tidak ada error berarti data sudah sesuai dengan desain kita
check_data(data_house, params)
```

Data Splitting and Data Serialization

[59]:
```
#Data Splitting

# pisahkan data x dan y (x adalah fitur, y adalah label)
x = data_house[params["predictors"]].copy()
y = data_house['Fire Alarm'].copy()
```

[60]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42, stratify = y)
x_valid, x_test, y_valid, y_test = train_test_split(x_test, y_test, test_size = 0.5, random_state = 42, strat
```

[61]:
```
#Data Serialization
joblib.dump(x_train, "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/e
joblib.dump(y_train, "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/e
joblib.dump(x_valid, "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/e
joblib.dump(y_valid, "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/e
joblib.dump(x_test, "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/ex
joblib.dump(y_test, "C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/ex
```

[61]:
```
['C:/Users/HP/Documents/Jeffri Ivander/Data Science/Pacmann AI academy/Tugas/ML Process/exercise2/processed/
y_test.pkl']
```