Workflow

```
Flask
  |
FlaskWTF
  |
numpy
pandas
sklearn
joblib
  |
step1
```

FLASK

In [1]:

```
from flask import Flask, jsonify, request, render_template
```

FALSK_WTF

In [2]:

```
from flask_wtf import FlaskForm
from wtforms import IntegerField
```

Primary Imports for ML Model

In [3]:

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
import joblib
```

How to use this application

- By default flask app runs on 5000 port number
- flask is web base framework which helps to build website
- These websites can execute on local as well as on server
- For local system it works on localhost or 127.0.0.1

URLS:

- Index page

  http://127.0.0.1:5000 (http://127.0.0.1:5000)
- Train ML Model Page

  http://127.0.0.1:5000/train (http://127.0.0.1:5000/train)
- Test ML Model Page

  http://127.0.0.1:5000/test (http://127.0.0.1:5000/test)
- Predict page - This will print result 0 or 1

  http://127.0.0.1:5000/predict (http://127.0.0.1:5000/predict)
- ML Model Report

  http://127.0.0.1:5000/report (http://127.0.0.1:5000/report)
- Help

  http://127.0.0.1:5000/help (http://127.0.0.1:5000/help)

In [4]:

```python
app = Flask(__name__)

app.config['SECRET_KEY'] = 'asdfasdfasdfasdfas'
# Create a class for Test the data
class FlaseAlarm(FlaskForm):
    ambient_temperature = IntegerField("Ambient Temperature")
    calibration = IntegerField("Calibration")
    unwanted_substance = IntegerField("Unwanted Substance")
    humidity = IntegerField("Humidity")
    h2s = IntegerField("H2S")
    detected_by = IntegerField("Detected By")

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/train')
def train():
    file_name = 'False Alarm Cases.xlsx'
    df_train = pd.read_excel(file_name)
    df_train = df_train.iloc[:,1:8]
    x = df_train.iloc[:,0:6]
    y = df_train['Spuriosity Index(0/1)']
    ml_model = GaussianNB()
    ml_file = 'ml_model.pkl'
    ml_model.fit(x,y)
    joblib.dump(ml_model, ml_file)
    return render_template('train.html', name=ml_file)

@app.route('/test', methods=['GET', 'POST'])
def test():
    form = FlaseAlarm()
    if request.method == 'POST':
        ml_file = 'ml_model.pkl'
        clf = joblib.load(ml_file)
        a = form.data['ambient_temperature']
        b = form.data['calibration']
        c = form.data['unwanted_substance']
        d = form.data['humidity']
        e = form.data['h2s']
        f = form.data['detected_by']

        input_values = [a,b,c,d,e,f]
        input_array = np.array(input_values)
        input_array = input_array.reshape(1,6)
        df_test = pd.DataFrame(input_array, columns=["Ambient Temperature", "Calibration","Unwanted Substance","Humidity","H2S","Dete
        y_pred = clf.predict(input_array)
        result = "No Danger"
        if y_pred == 1:
            result = 'Danger'
        return "Prediction is %s"%(result)
    else:
        return render_template('test.html', form = form)

@app.route('/predict')
def predict():
    return render_template('predict.html')

@app.route('/report')
def report():
    return render_template('report.html')

@app.route('/help')
def help():
    return render_template('help.html')

if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off

 * Running on http://127.0.0.1:5000/ (http://127.0.0.1:5000/) (Press CTRL+C to quit)
```

In [6]:

```python
file_name = 'False Alarm Cases.xlsx'
df_train = pd.read_excel(file_name)
```

In [7]:

```
df_train.head()
```

Out[7]:

	Case No.	Ambient Temperature(deg C)	Calibration(days)	Unwanted substance deposition(0/1)	Humidity(%)	H2S Content(ppm)	detected by(% of sensors)	Spuriosity Index(0/1)	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	Case # 1	-2	226	1	96	9	21	1	NaN	NaN	NaN
1	Case # 2	4	134	1	83	4	77	0	NaN	NaN	NaN
2	Case # 3	7	163	0	69	2	81	0	NaN	NaN	NaN
3	Case # 4	5	162	0	80	6	69	0	NaN	NaN	NaN
4	Case # 5	2	192	1	87	3	67	0	NaN	NaN	NaN

In [8]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1892 entries, 0 to 1891
Data columns (total 11 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Case No.                            1892 non-null   object 
 1   Ambient Temperature( deg C)         1892 non-null   int64  
 2   Calibration(days)                   1892 non-null   int64  
 3   Unwanted substance deposition(0/1)  1892 non-null   int64  
 4   Humidity(%)                         1892 non-null   int64  
 5   H2S Content(ppm)                    1892 non-null   int64  
 6   detected by(% of sensors)           1892 non-null   int64  
 7   Spuriosity Index(0/1)               1892 non-null   int64  
 8   Unnamed: 8                          0 non-null      float64
 9   Unnamed: 9                          0 non-null      float64
 10  Unnamed: 10                         1 non-null      float64
dtypes: float64(3), int64(7), object(1)
memory usage: 162.7+ KB
```

In [9]:

```
df_train.describe()
```

Out[9]:

	Ambient Temperature(deg C)	Calibration(days)	Unwanted substance deposition(0/1)	Humidity(%)	H2S Content(ppm)	detected by(% of sensors)	Spuriosity Index(0/1)	Unnamed: 8	Unnamed: 9	Unnamed: 10
count	1892.000000	1892.000000	1892.000000	1892.000000	1892.000000	1892.000000	1892.000000	0.0	0.0	1.0
mean	3.449789	131.633192	0.482030	82.513214	5.532241	71.610465	0.173890	NaN	NaN	0.0
std	3.323731	67.741005	0.499809	7.659900	2.271502	21.203802	0.379115	NaN	NaN	NaN
min	-2.000000	10.000000	0.000000	69.000000	2.000000	20.000000	0.000000	NaN	NaN	0.0
25%	1.000000	75.000000	0.000000	76.000000	4.000000	63.000000	0.000000	NaN	NaN	0.0
50%	3.000000	133.000000	0.000000	82.000000	6.000000	76.000000	0.000000	NaN	NaN	0.0
75%	6.000000	188.000000	1.000000	89.000000	8.000000	88.000000	0.000000	NaN	NaN	0.0
max	9.000000	250.000000	1.000000	96.000000	9.000000	100.000000	1.000000	NaN	NaN	0.0

In [ ]:

```

```