

## Development of an automatic gas sampling cum injection unit and a graphical user interface of a feature extraction tool box based on PYTHON for sensor array data analysis

Debaayus Swain<sup>#,1</sup>, Kunal Gupta<sup>#,1</sup>, A. Sree Rama Murthy<sup>#,2,3,\*</sup>, V. Jayaraman<sup>2,3</sup>

<sup>1</sup>Birla Institute of Technology and Science, Pilani, India - 333 031

<sup>2</sup>Indira Gandhi Centre for Atomic Research, Kalpakkam, India - 603 102

<sup>3</sup>Homi Bhabha National Institute, Kalpakkam - 603 102

### Supplementary Information

#### S1. Creation of Data Matrices

##### a) Data matrix (Type 1)

The data matrix of type 1 that consists of all the features of a particular sensor calculated by iterating the loop for all the pois and independently calling all the feature functions and storing them in the data frame subsequently providing option to user to store in separate csv files as per the code below:

```
def matrix_type1(df,list_poi,gap,points,sensor_name):
    sensor_list=df.columns.tolist()
    sensor=sensor_list.index(sensor_name)
    num_poi=len(list_poi)
    list_poi.sort(reverse=False)
    num_rows = df.shape[0]
    points = points // gap
    poi = list_poi[0]//gap
    if num_poi==1:
        next = num_rows-((list_poi[0])//gap)
    else:
        next = (list_poi[1]-list_poi[0])//gap
    sens = find_sensitivity(sensor,poi,df,next)
    recslope = recovery_slope(sensor,poi,df,next,gap)
    resslope = response_slope(sensor,poi,df,next,gap)
```

```
restime = response_time(sensor,poi,df,next,gap)
rectime = recovery_time(sensor,poi,df,next,gap)
area = integral_area(sensor,poi,df,points,gap)
Ratio = 1-sens
sens *=100
features = [[sens,recslope,resslope,rectime,restime,area,Ratio]]
df1 = pd.DataFrame(features,columns=['Response(in %)', 'Recovery Slope', 'Response Slope', 'Recovery Time', 'Response Time', 'Integral Area', 'Ratio'])
for i in range(1,num_poi):
    poi = list_poi[i]//gap
    if i+1==num_poi:
        next = num_rows-((list_poi[i]//gap)
    else:
        next = (list_poi[i+1]-list_poi[i]//gap)
    sens = find_sensitivity(sensor,poi,df,next)
    recslope = recovery_slope(sensor,poi,df,next,gap)
    resslope = response_slope(sensor,poi,df,next,gap)
    restime = response_time(sensor,poi,df,next,gap)
    rectime = recovery_time(sensor,poi,df,next,gap)
    area = integral_area(sensor,poi,df,points,gap)
    Ratio = 1-sens
    sens *= 100
    features = [[sens,recslope,resslope,restime,rectime,area,Ratio]]
    df1 = df1.append(pd.DataFrame(features,columns=['Response(in %)', 'Recovery Slope', 'Response Slope', 'Recovery Time', 'Response Time', 'Integral Area', 'Ratio']),ignore_index=True)
    index_col = []
    for i in range (1,num_poi+1):
        index_col.append('Signal '+str(i))
    df1.insert(loc=0,column='Signal',value=index_col)
    df1.set_index("Signal",inplace=True)
return df1
```

b) *Data matrix (Type 2)*

The data matrix type 2 is the table of a chosen feature for all the sensors. The elements of the data frame are computed by iterating all the sensors followed by the signals associated with that feature.

The code for this type of matrix is given below.

```
def matrix_type2(feature,df,list_poi,gap,total_sensor, dat_col, points):  
    num_rows=df.shape[0]  
    num_poi=len(list_poi)  
    start=int(dat_col)-1  
    column_name = df.columns[start: start+total_sensor].tolist()  
    df1 = pd.DataFrame(columns=column_name)  
    if feature=='Response(in %)':  
        for i in range(num_poi):  
            poi = list_poi[i]//gap  
            if i+1==num_poi:  
                next = num_rows-((list_poi[i]//gap)  
            else:  
                next = (list_poi[i+1]-list_poi[i]//gap  
            sens=[]  
            for i in range (start,start+total_sensor):  
                sens.append(find_sensitivity(i,poi,df,next)*100)  
            df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)  
    elif feature=='Recovery Slope':  
        for i in range(num_poi):  
            poi = list_poi[i]//gap  
            if i+1==num_poi:  
                next = num_rows-((list_poi[i]//gap)  
            else:  
                next = (list_poi[i+1]-list_poi[i]//gap  
            sens=[]  
            for i in range (start,start+total_sensor):  
                sens.append(response_slope(i,poi,df,next,gap))  
            df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
```

```
elif feature=='Response Slope':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i]//gap)
        else:
            next = (list_poi[i+1]-list_poi[i]//gap)
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(recovery_slope(i,poi,df,next,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Recovery Time':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i]//gap)
        else:
            next = (list_poi[i+1]-list_poi[i]//gap)
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(response_time(i,poi,df,next,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Response Time':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i]//gap)
        else:
            next = (list_poi[i+1]-list_poi[i]//gap)
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(recovery_time(i,poi,df,next,gap))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Integral Area':
```

```
points = points // gap
for i in range(num_poi):
    poi = list_poi[i]//gap
    if i+1==num_poi:
        next = num_rows-((list_poi[i])//gap)
    else:
        next = (list_poi[i+1]-list_poi[i])//gap
    sens=[]
    for i in range (start,start+total_sensor):
        sens.append(integral_area(i,poi,df,points,gap))
    df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
elif feature=='Ratio':
    for i in range(num_poi):
        poi = list_poi[i]//gap
        if i+1==num_poi:
            next = num_rows-((list_poi[i])//gap)
        else:
            next = (list_poi[i+1]-list_poi[i])//gap
        sens=[]
        for i in range (start,start+total_sensor):
            sens.append(1-find_sensitivity(i,poi,df,next))
        df1=df1.append(pd.DataFrame([sens],columns=column_name),ignore_index=True)
index_col = []
for i in range (1,num_poi+1):
    index_col.append('Signal '+str(i))
df1.insert(loc=0,column='Signal',value=index_col)
df1.set_index("Signal",inplace=True)
return df1
```

## S2. Python codes for response and recovery time calculations

```
def response_time(sensor,poi,df,next,gap):
    ser1=df.iloc[:,sensor]
    baseline =base_line(poi,ser1)
    ser2=df.iloc[poi-1:poi+next-1,sensor]
    index_tip = tip(ser2,poi,next)
    delR = (baseline-ser2[index_tip])*0.90
    R90 = ser2[poi] -delR
    index1=poi
    index1min = abs(ser2[poi]-R90)
    for i in range(poi,index_tip+1):
        index1min = min(index1min,abs(ser2[i]-R90))
        if index1min == abs(ser2[i]-R90):
            index1 = i
    index2 = index1-1
    time1 = index1 * gap
    time2 = index2 * gap
    time = (((R90-ser2[index2])/(ser2[index1]-ser2[index2]))*(time1-time2))+time2)-(poi*gap)
    return time
```

```
def recovery_time(sensor,poi,df,next,gap):
    ser1=df.iloc[:,sensor]
    baseline =base_line(poi+next,ser1)
    ser2=df.iloc[poi-1:poi+next-1,sensor]
    index_tip = tip(ser2,poi,next)
    delR = (baseline-ser2[index_tip])*0.90
    R90 = ser2[index_tip]+delR
    index1 = index_tip
    index1min = abs(ser2[index_tip]-R90)
    for i in range(index_tip,poi+next):
        index1min = min(index1min,abs(ser2[i]-R90))
        if index1min == abs(ser2[i]-R90):
            index1 = i
    index2 = index1-1
```

```
time1 = index1 * gap
time2 = index2 * gap
time = (((R90-ser2[index2])/(ser2[index1]-ser2[index2]))*(time1-time2))+time2)-(index_tip*gap)
return time
```

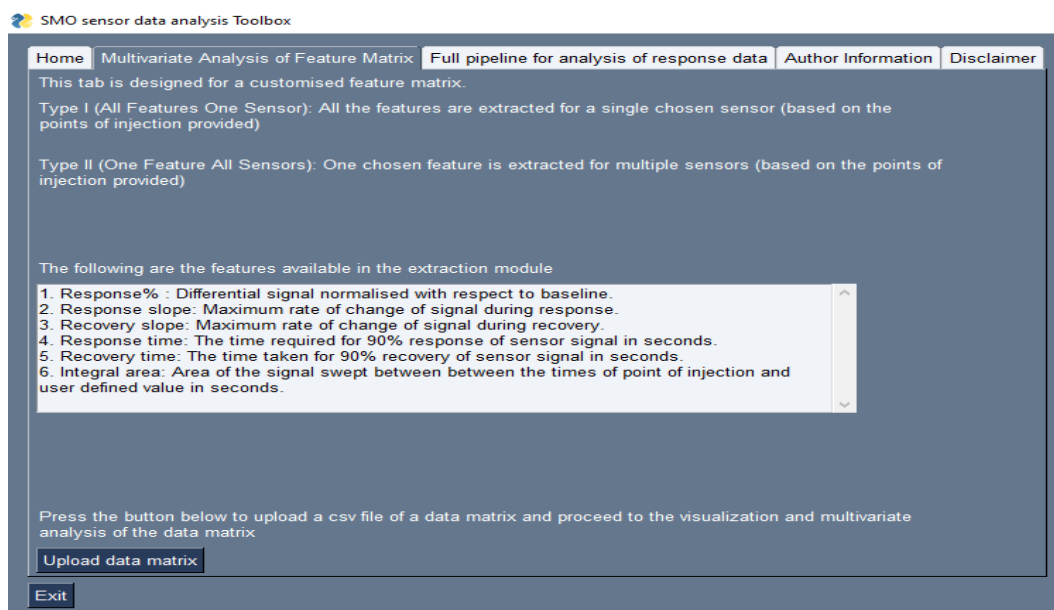


Fig. S1 The descriptors of Multivariate Analysis of Feature Matrix tab

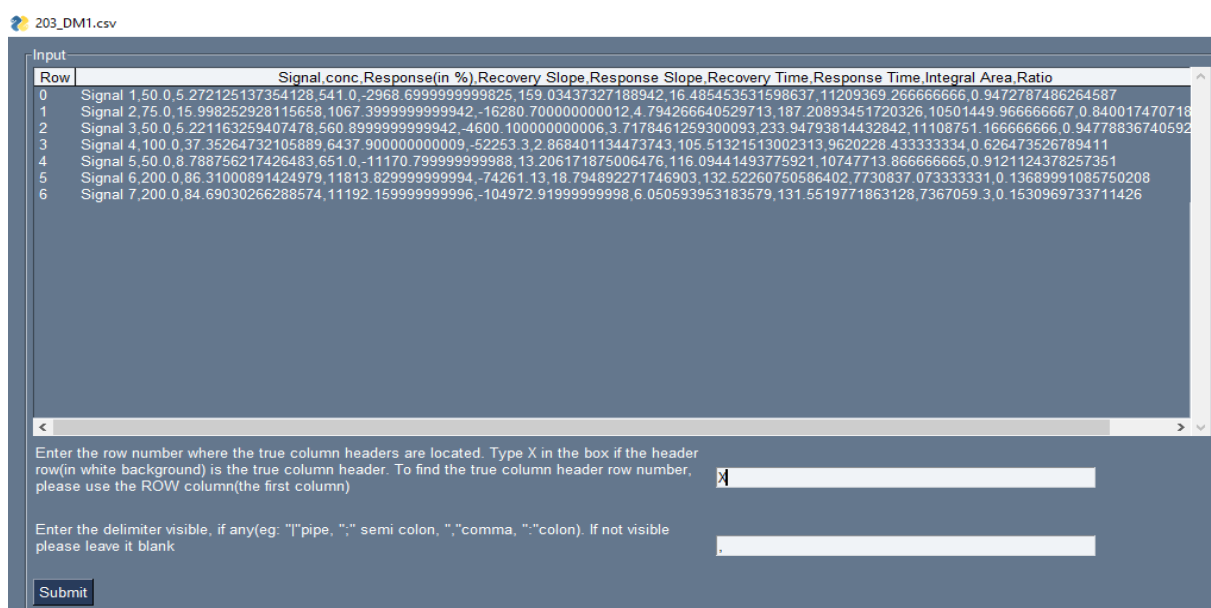


Fig. S2 Screenshot depicting the typical view of the uploaded data

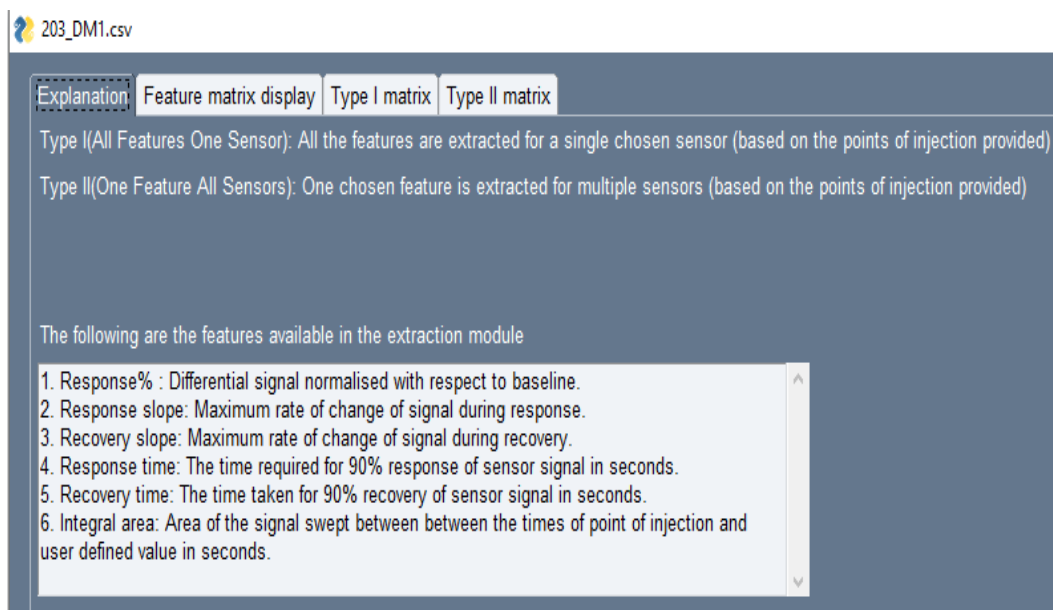


Fig. S3 Explanation tab with a short summary of the type of matrix and the features of the signals which can be visualized

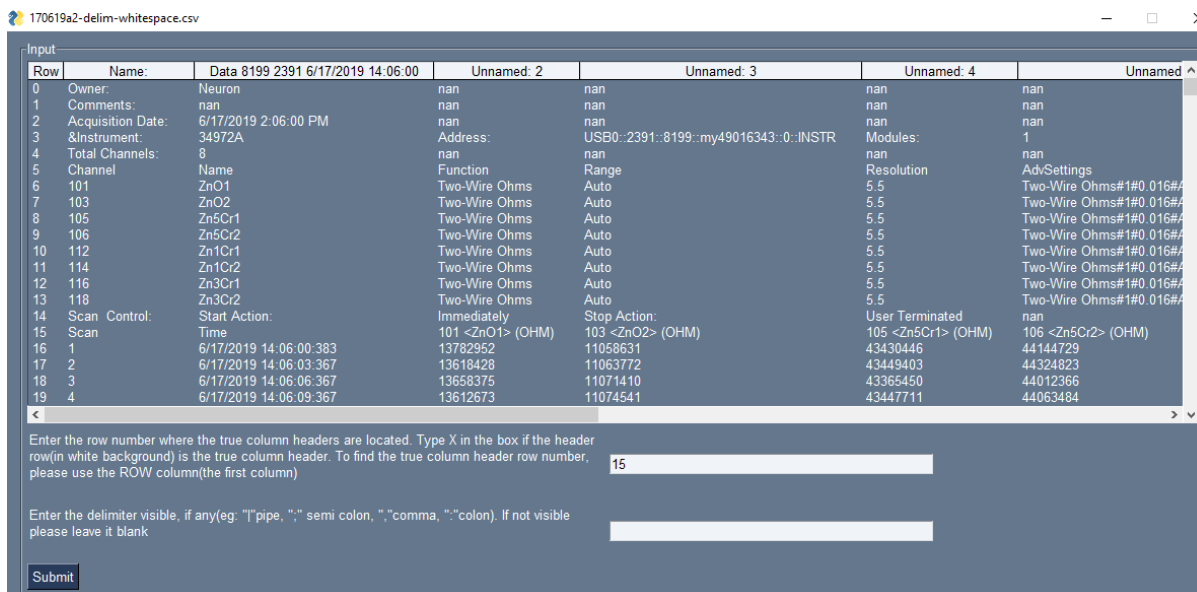


Fig. S4 In this typical example, user must enter 15 as the true headers to skip 15 rows and leave delimiter option blank for formatting the multiple sensors time series data



170619a2-delim-whitespace.csv

Input

Scan	Time	101 <ZnO1> (OHM)	103 <ZnO2> (OHM)	105 <Zn5Cr1> (OHM)	106 <Zn5Cr2> (OHM)	112 <Zn1Cr1> (OHM)	114 <Zn1Cr2> (OHM)
1	6/17/2019 14:06:00:383	13782952.0	11058631.0	43430446	44144729	31862440	31206721
2	6/17/2019 14:06:03:367	13618428.0	11063772.0	43449403	44324823	31859393	31124799
3	6/17/2019 14:06:06:367	13658375.0	11071410.0	43365450	44012366	31898662	31145110
4	6/17/2019 14:06:09:367	13612673.0	11074541.0	43447711	44063484	31919650	31220262
5	6/17/2019 14:06:12:367	13675978.0	11082518.0	43398286	44266936	31918296	31115659
6	6/17/2019 14:06:15:367	13651943.0	11082221.0	43409796	43941953	31959596	31187425
7	6/17/2019 14:06:18:367	13627906.0	11082898.0	43495442	44168087	31937253	31181671
8	6/17/2019 14:06:21:367	13675640.0	11086093.0	43402010	44203970	31970090	31161021
9	6/17/2019 14:06:24:367	13631632.0	11081058.0	43437217	43942969	32012405	31246667
10	6/17/2019 14:06:27:367	13626891.0	11072002.0	43503228	44207694	31953502	31230756
11	6/17/2019 14:06:30:367	13652282.0	11068004.0	43390500	44249332	31958580	31192842
12	6/17/2019 14:06:33:367	13612673.0	11057467.0	43436878	43943646	32016129	31287289
13	6/17/2019 14:06:36:367	13600486.0	11056542.0	43478855	44191445	31963996	31259192
14	6/17/2019 14:06:39:367	13636033.0	11058398.0	43398286	44283185	31977537	31201643
15	6/17/2019 14:06:42:367	13603194.0	11044942.0	43393886	43975467	32032717	31261223
16	6/17/2019 14:06:45:367	13581529.0	11038573.0	43468361	44076686	31985662	31241250
17	6/17/2019 14:06:48:367	13594393.0	11041895.0	43406750	44321099	31938946	31176593
18	6/17/2019 14:06:51:367	13602856.0	11040731.0	43364096	44167410	31953164	31140709
19	6/17/2019 14:06:54:367	13589992.0	11037706.0	43403703	43937552	31968059	31151204
20	6/17/2019 14:06:57:367	13585591.0	11038446.0	43470392	44081764	31925066	31142063

Parameters from visual inspection

Select YES if index column(first column with values 1, 2, 3) is visible. Select NO to allow the program to create an index column  Yes  No

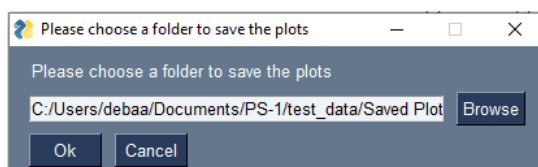
Enter the column number of the timestamp column (eg. Column number is 2 if the timestamp column is the 2nd column(if index is 1st column)). If no timestamp column, then please enter X

Enter the column number of the first sensor data. (eg. Column number is 3 if the first sensor data column is the 3rd column(if index and timestamp are the first 2 columns.))

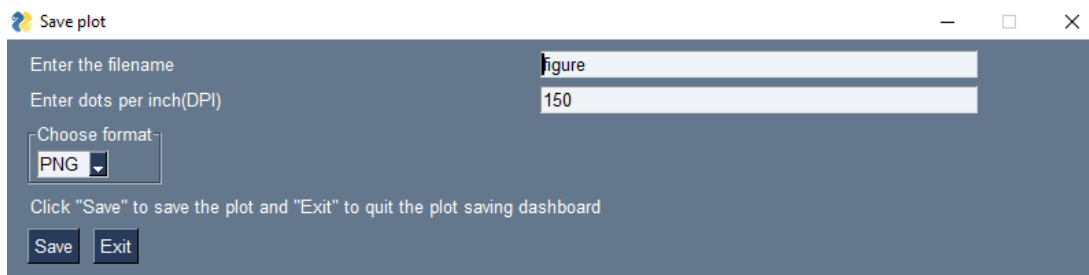
If you missed entering your delimiter, please restart the program

Press submit to confirm the above dataframe for further computation

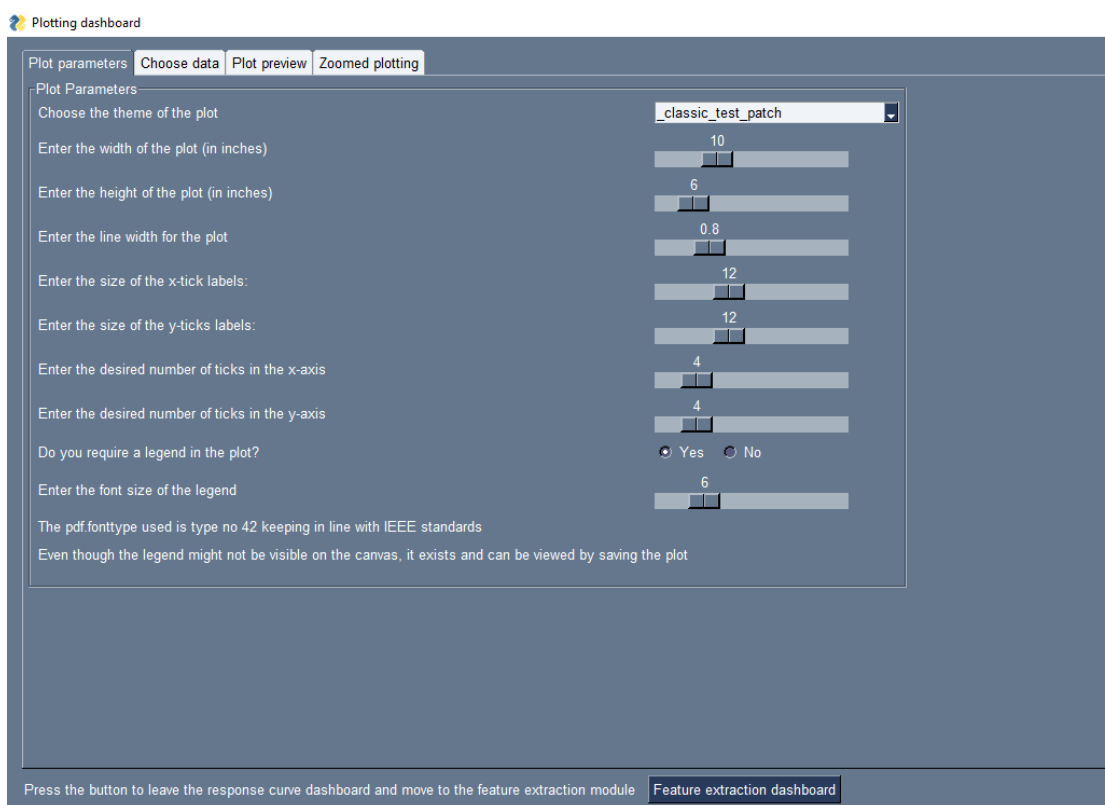
Fig. S5 Input screen for visual inspection with three entry options viz., index column, time stamp column and column number of 1<sup>st</sup> sensor's data



(a)



(b)



(c)

Fig. S6 Options for (a) saving the plot, (b) choosing the format and resolution (DPI) and (c) customized parameter selection for visualisation of the plots in plotting dashboard

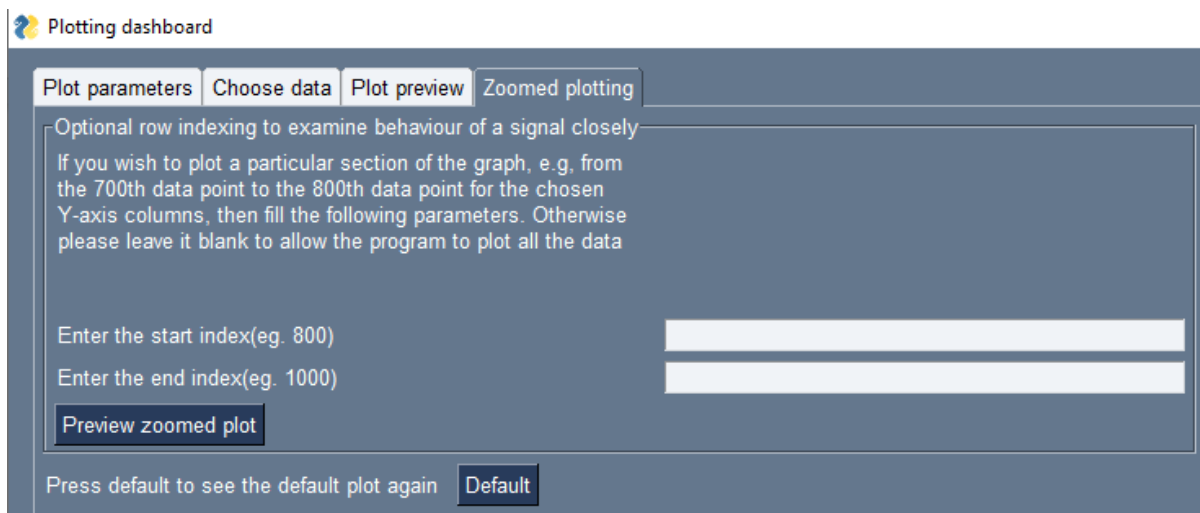


Fig. S7 Zoomed plotting option with user-selected section of the time series plot

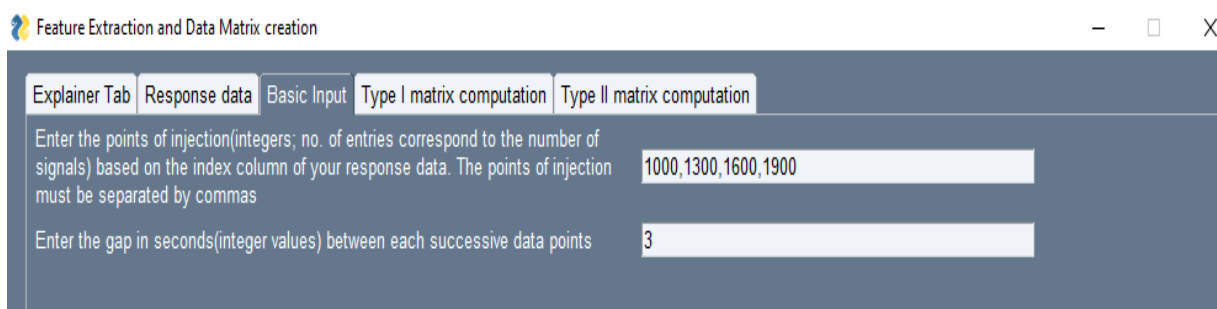


Figure S8 Basic input tab for pois and successive time gap between the time series data

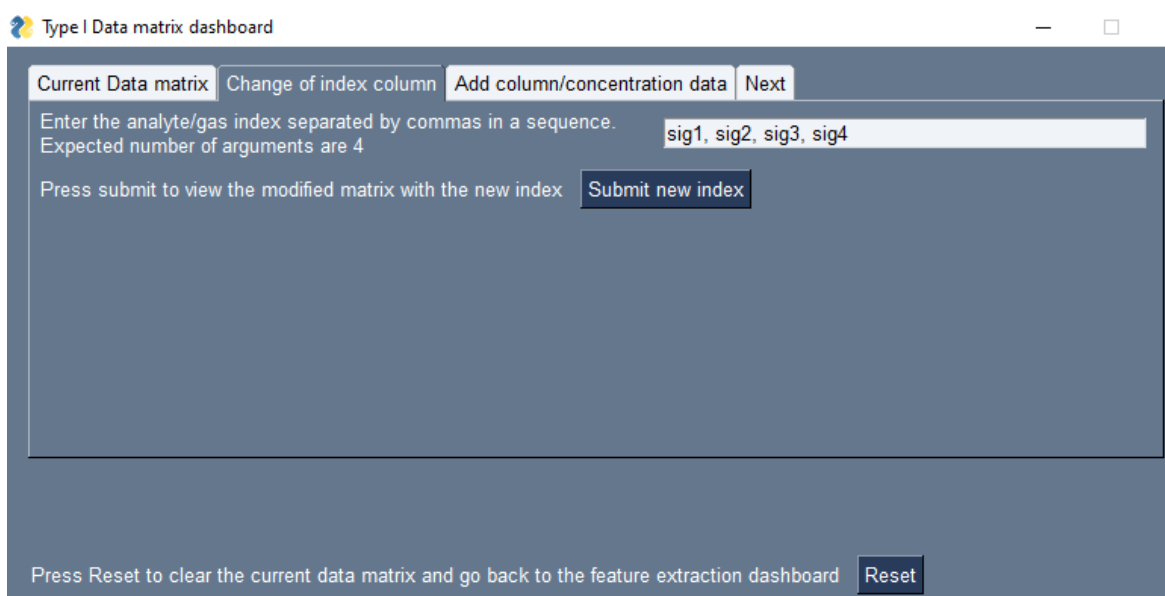


Fig. S9 Option to change the index column names as per the user's choice

Type I Data matrix dashboard

Current Data matrix | Change of index column | Add column/concentration data | Next

index	Response(in %)	Response Slope	Recovery Slope	Response Time	Recovery Time	Integral Area	Ratio
sig1	65.334303237	-1102954.3	177665.66666	12.022097543	216.71853372	334594082.19	0.3466569676
sig2	63.777747806	-1153457.1666	175282.0	14.420063434	297.76282010	340741310.2	0.3622225219
sig3	62.198414658	-1153443.2	179287.66666	26.212340965	218.26220899	354207857.40	0.3780158534
sig4	72.716842123	-1336266.9333	178103.0	1632.5618956	1115.0135582	785347537.0	0.2728315787

Save data matrix

(a)

Type II Data matrix dashboard

Current Data matrix | Change of index column | Add column/concentration data | Next

Signal	101 <ZnO1>	103 <ZnO2>	105 <Zn5Cr1>	106 <Zn5Cr2>	112 <Zn1Cr1>	114 <Zn1Cr2>	116 <Zn3Cr1>	118
Signal 1	65.334303237	55.920112190	4.1728961496	3.8648456116	9.2649890628	11.298431940	1.8120916169	1.7
Signal 2	63.777747806	54.338876076	3.7958649322	4.0430029565	7.9448179140	9.8315943263	1.8992695281	0.9
Signal 3	62.198414658	52.379817171	3.4643783156	2.7364048171	7.6227529816	9.1012951905	1.5457087531	0.9
Signal 4	72.716842123	64.645857120	6.3359718621	5.6641369301	15.696903548	18.199800303	2.6159982177	2.7

Save data matrix

(b)

Fig. S10 Tab showing the results with changed indices in (a) type I and (b) type II data matrices

Type I Data matrix dashboard

Current Data matrix | Change of index column | Add column/concentration data | Next

Add concentration column details, otherwise not present in initial data

Enter the name of the column to be inserted:

Enter the new values in sequence separated by commas. The number of values entered must match the number of rows in your data matrix. Expected number of entries are 4:

Press submit to view the modified matrix with the new column:

Fig. S11 Option to add columnar data of concentrations to the current data matrix

index	Concentration	Response(in %)	Response Slope	Recovery Slope	Response Time	Recovery Time	Integral Area	
sig1	5.0	65.334303237	-1102954.3	177665.66666	12.022097543	216.71853372	334594082.19	0.3
sig2	10.0	63.777747806	-1153457.1666	175282.0	14.420063434	297.76282010	340741310.2	0.3
sig3	15.0	62.198414658	-1153443.2	179287.66666	26.212340965	218.26220899	354207857.40	0.3
sig4	20.0	72.716842123	-1336266.9333	178103.0	1632.5618956	1115.0135582	785347537.0	0.2

Fig. S12 Visualization of the modified type 1 data matrix

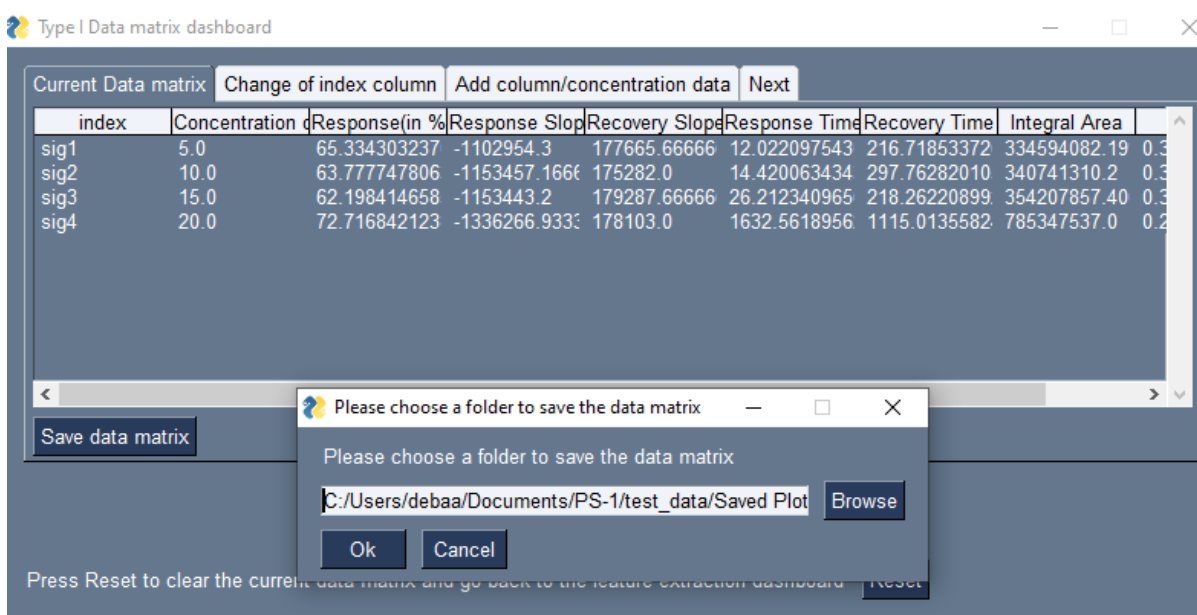


Fig. S13 Selection of a location to save the modified data matrix

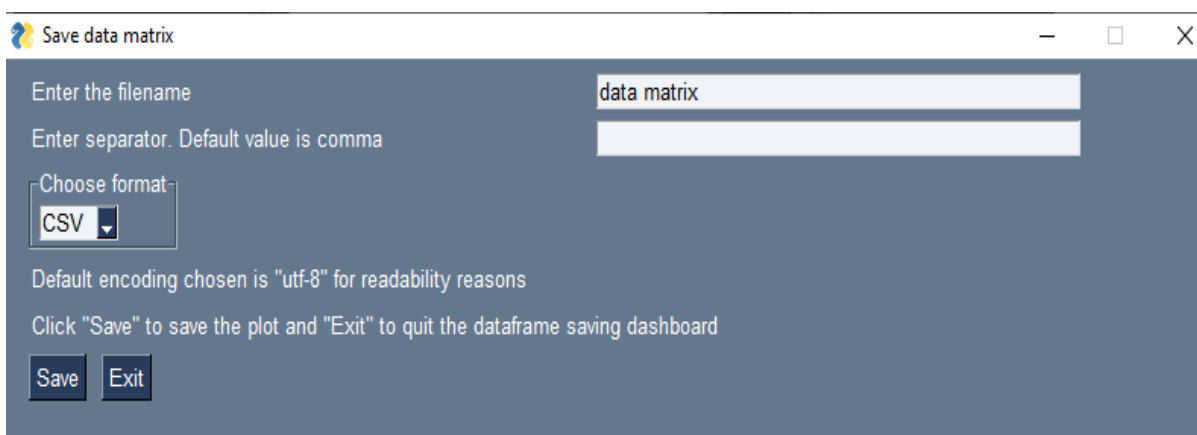


Figure S14 Screenshot of exercising the choice of format of the modified data matrix while saving

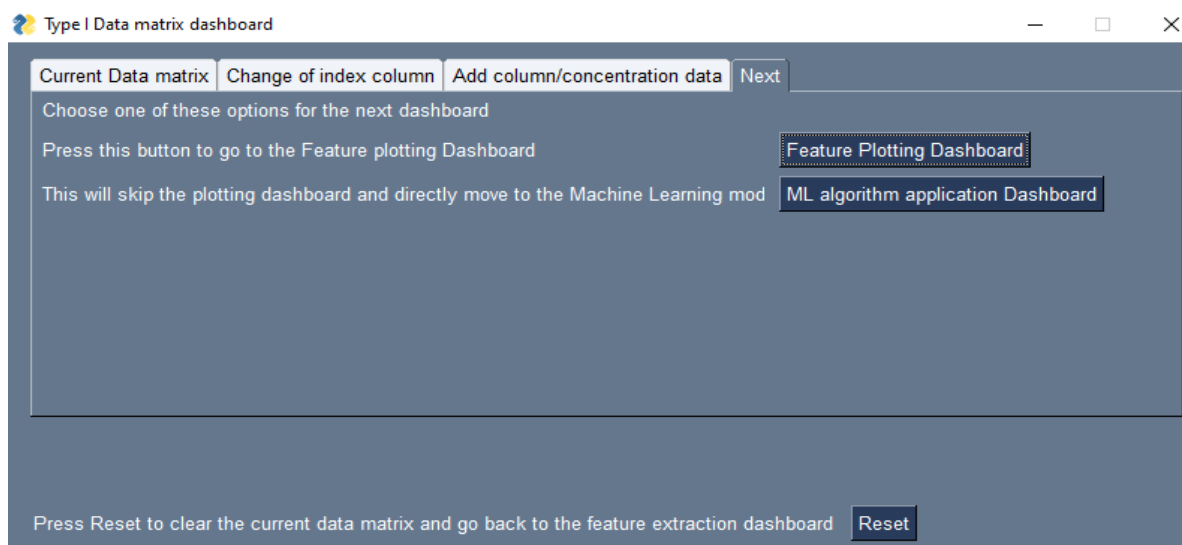


Figure S15 Screenshot of 'Next' tab to proceed either for feature visualization or for resetting the user-defined parameters

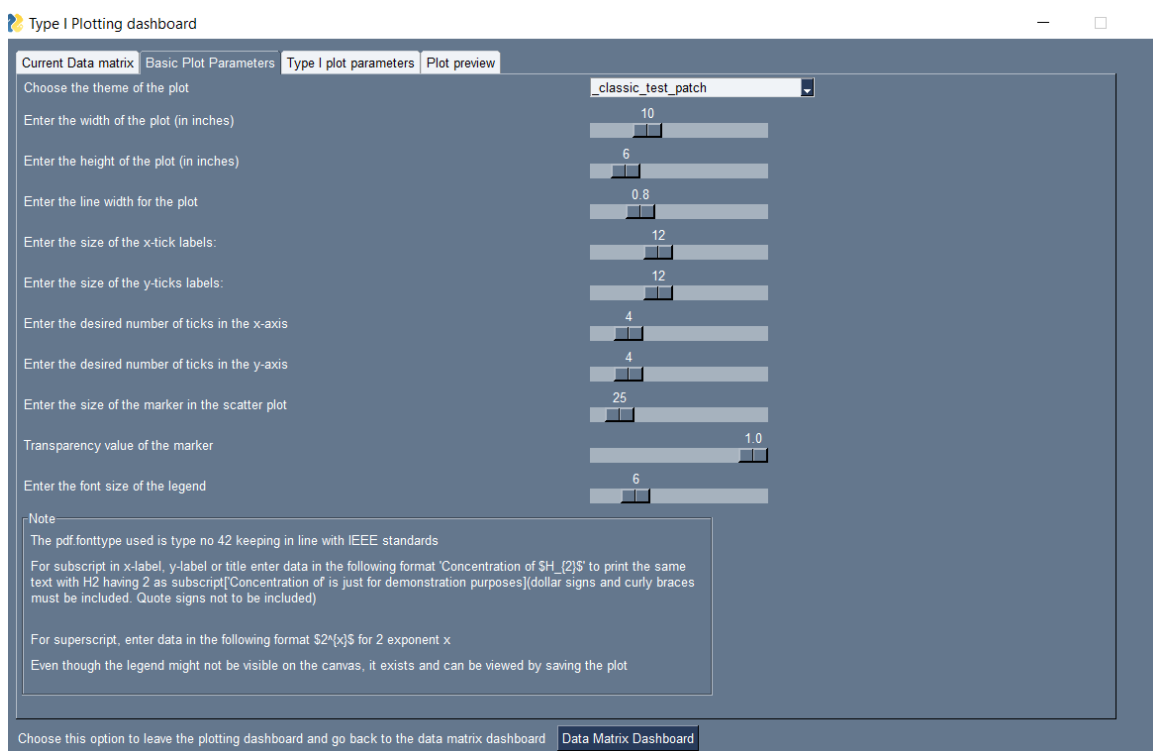


Fig. S2 Basic plot parameters tab to format the visualization details

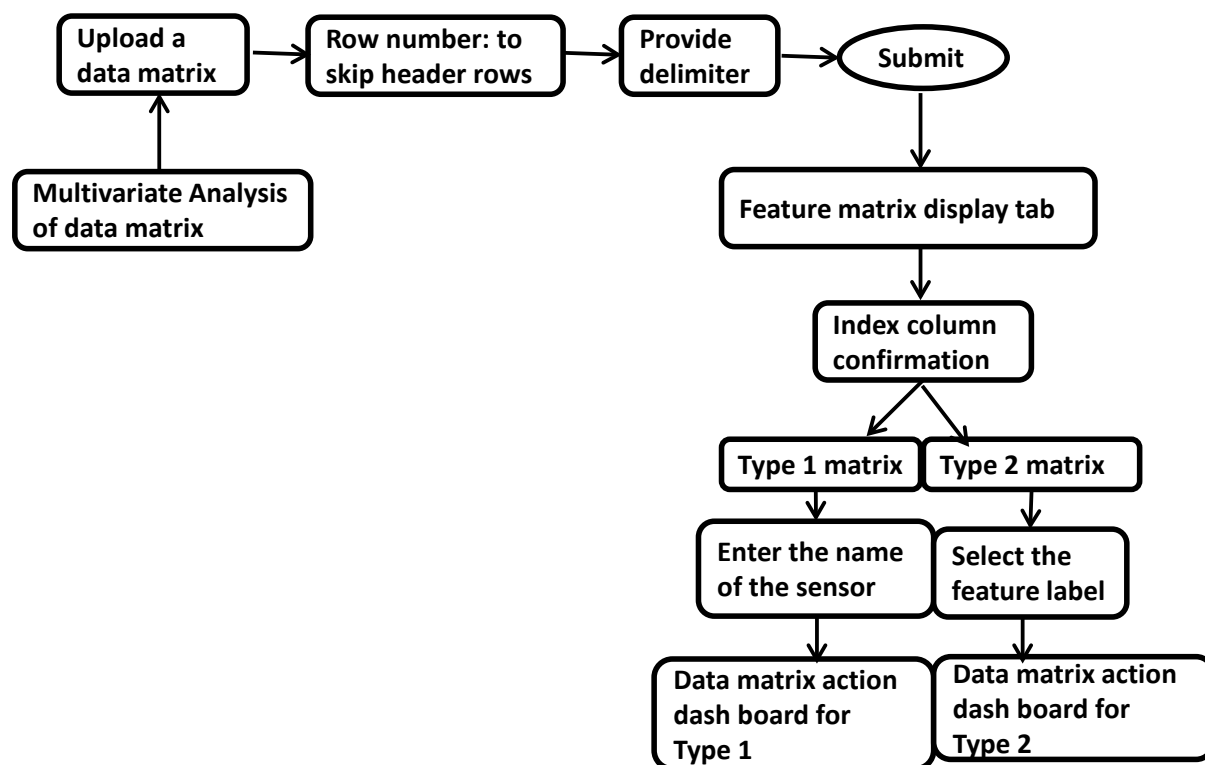
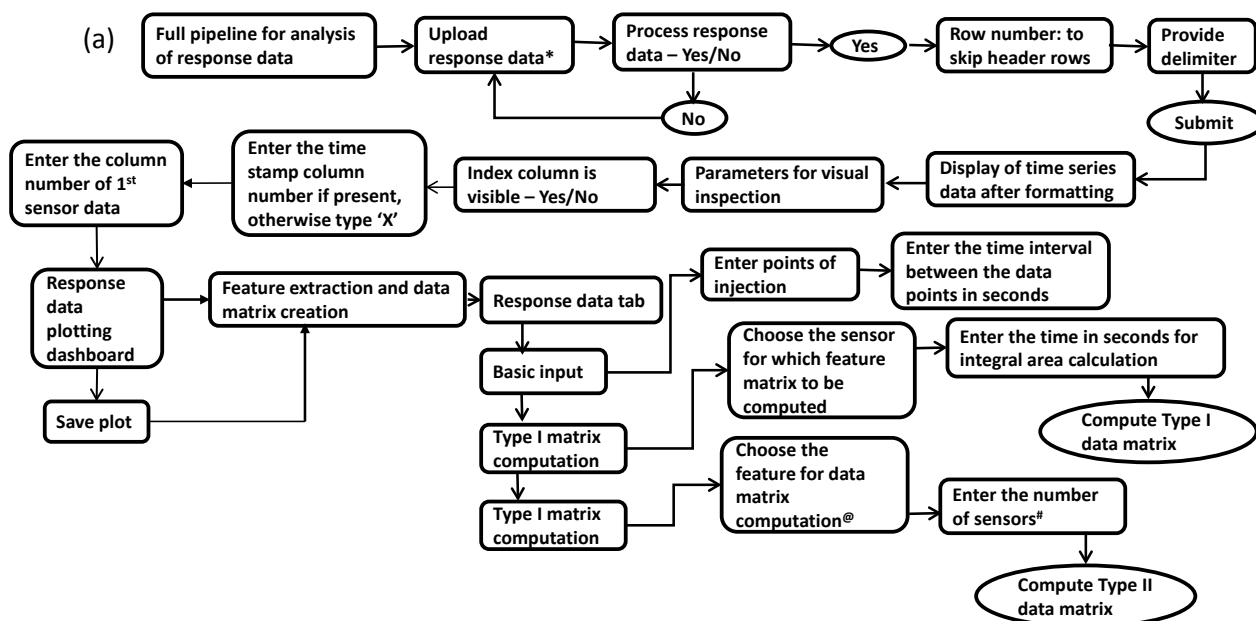
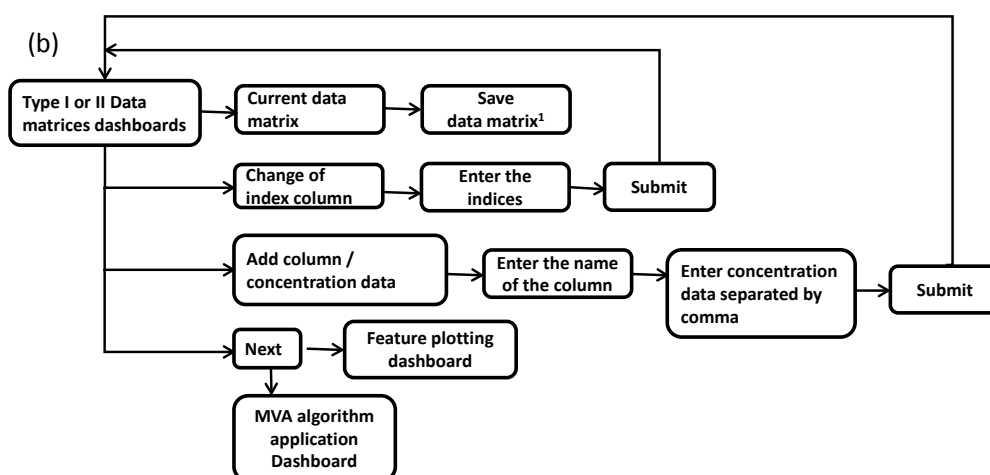


Fig. S17 A design flow sheet for multivariate analysis of data matrix



\* Response data should contain only one 'time' column with multiple sensor responses with common points of injection (POI)

@for integral area calculation provide time in seconds; #Do not consider empty columns



<sup>1</sup>One can opt \*.csv or \*.xlsx or \*.txt formats with chosen separator like , or |, etc.

Fig. S3A design flow sheet showing (a) Full pipeline for analysis of response data, (b) the process flow sheet of Type I and Type II data matrices.