TITLE:
A PREDICTIVE MODELLING OF THE GAS SENSOR DATASET

PROJECT DESCRIPTION:

it’s always advisable and cool to have a system that helps to monitor access to highly restrictive areas through odor detection/sensing for proper security attention and monitoring. Therefore the aim of this project is to model a system that gives authorized personnel access to our systems by detecting the odor being emitted from them.

METHODOLOGY

In order to achieve this task,we did the following:

1. we needed to build a classifier system that identifies/ senses the odor and then successful predict the gas detected accurately.
2. We needed to build a regression system that is able to predict the concentration level of the gas detected/identified.

PROJECT APPROACH

To get this task done, we discovered that this task is a machine learning problem because it involves predictive analysis that could be learnt from a massive information of data.

The software then used in achieving this is Anaconda (which prominently uses/utilizes the universal python programming language).

BRIEF DESCRIPTION OF ANACONDA

The software package called anaconda is basically built and designed for data analysis, synthesis, predictive modeling and deployment to servers and bases.

It uses the python programming language.

It is an open source software with daily contribution from the anaconda community. It has some inbuilt packages like jupyter, spyder, Ipython, Orange e.t.c

For the sake of this project we therefore restrict our attention to the use of jupyter notebook for a chronological analysis and visualization.

Existing work

This project looks similar to a task done by a group of engineers in calibrating their sensor devices by taking records from their sensors for 21 days.

They used 5 different boards which contains 8 sensor array each (making a total of 40 sensors).

Their aim was to detect environmental effects on the sensor operation, that’s why it was taken for 21 days.

They were to analyze and also predict the concentration level of gas being detected by their sensor array and then check for variation.

Existing work

- They were to work with 4 gases CO(carbon(IV) oxide, ME(methane), ET(Ethylene), EA(Ethane).
- Each gases were passed individually on different days and different boards for 21 days and records were taken for a total of 600seconds at a sample rate of 0.01Hz, which makes a total of 60,000 data recorded by each board.
- 10 different concentration level were used for the 4 gases individually (a total of 40 concentration level for all gases considered) in 21 days.
- Each task per day for the gas being considered was repeated 4 times and were all labelled as R1, R2,R3,R4.
- The data gotten were all saved in a text file, a total of 640 unprocessed text files for all the boards, a typical text file is labelled in this format:

  B1_F020_CO_R1.txt, which implies the data in the text file was taken from board One which contains 8 different sensors on it. A gas CO was passed to this sensors at a concentration of 20ppm. R1 indicates the first repitition.

Existing work

- The aim of those who gathered the data was to calibrate their instruments, therefore they know the gas they were dealing with and the concentration level of the gas.
- To make this very simple for them for a gas they had to conduct the experiment on 5 replicate boards that have 8 different sensors on them.
- For each board 10 different concentration levels for all the 4 gases for 4 individual iteration which makes us have 640 unprocessed text files.
- For this project since we are not aiming at calibrating an instrument we will simple restrict our work to a single board, we will use Board One for our analysis.

Project analysis and results

Project analysis and results

Here is how we went about analyzing the task, we got a dataset basically for board one and then we had 160 files to process for board 1.

We took all text files of the same concentration and gas and then merged them into a single file to represent the particular gas at that concentration level.

Since we are not aiming at calibration we decided to scale down from 160 files to 64 files each containing 60000 raw datas.

That is, for each gas we took 3 different concentration levels to represent that gas. Each concentration level has 4 individual repitition.

Project analysis and result

- This sums up to 12 different concentration level and gases (3 concentration level selected for the 4 different gases). Each of this was repeated 4 different times (making a total of 64 unprocessed text files).
- All repetition of similar concentration level and same gases were all merged to make a single file by suming them up using the pandas in python.
- Before doing all this, each repetition on visualization had outliers that need to be removed or cleared so that we can have a clean data being fed into our algorithm for an optimized performance.
- So therefore all datas were cleaned before merging.
- To show this we will show what we did using a particular concentration level and a particular gas.
- The same procedure goes for all other gases and the 3 concentration attached to each of them.
- For example considering gas CO at 40ppm.

Data loading and data cleaning

In [21]:
```
import os
import sys
os.listdir()
```

Out[21]:
```
['.ipynb_checkpoints',
 'B1_GCO_F040_R1.txt',
 'B1_GCO_F040_R2.txt',
 'B1_GCO_F040_R3.txt',
 'B1_GCO_F040_R4.txt',
 'Box_plot_of_sense_eightboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_eight_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_fiveboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_five_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_fourboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_four_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_oneboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_one_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_sevenboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_seven_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_sixboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_six_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_threeboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_three_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_twoboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_sense_two_no_outliers_board_one_CO_at_40ppm.jpeg',
 'Box_plot_of_timeboard_one_CO_at_40ppm.jpeg',
 'Box_plot_of_time_no_outliers_board_one_CO_at_40ppm.jpeg',
 'merge_co_40.ipynb']
```

Data cleaning

In [22]:
```
import pandas as pd
repitition_one = pd.read_csv(r'B1_GCO_F040_R1.txt',sep = '\t',header = None)
repitition_one.columns = ['time','sense_one', 'sense_two', 'sense_three','sense_four','sense
_five',
                          'sense_six','sense_seven','sense_eight']
repitition_one.head()
```

Out[22]:

	time	sense_one	sense_two	sense_three	sense_four	sense_five	sense_six	sense_seven	sense_eight
0	0.00	39.22	18.64	21.73	5.58	73.08	45.87	55.81	7.01
1	0.01	39.22	18.61	21.73	5.58	73.30	45.87	55.81	7.01
2	0.02	39.22	18.61	21.69	5.58	73.08	45.67	55.81	7.01
3	0.03	39.22	18.61	21.73	5.58	73.08	45.67	55.81	7.01
4	0.04	39.22	18.61	21.69	5.58	73.30	45.77	55.81	7.01

In [23]:
```
repitition_two = pd.read_csv(r'B1_GCO_F040_R2.txt',sep = '\t',header = None)
repitition_two.columns = ['time','sense_one', 'sense_two', 'sense_three','sense_four','sense
_five',
                          'sense_six','sense_seven','sense_eight']
repitition_two.head()
```

Out[23]:

	time	sense_one	sense_two	sense_three	sense_four	sense_five	sense_six	sense_seven	sense_eight
0	0.00	36.43	17.09	19.84	5.81	68.66	41.78	52.64	7.03
1	0.01	36.43	17.09	19.84	5.81	68.66	41.87	52.51	7.03
2	0.02	36.50	17.09	19.84	5.81	68.86	41.87	52.76	7.03
3	0.03	36.50	17.14	19.86	5.81	68.66	41.78	52.51	7.03
4	0.04	36.43	17.11	19.84	5.81	68.86	41.78	52.64	7.03

Data cleaning

In [24]:
```
repitition_three = pd.read_csv(r'B1_GCO_F040_R3.txt',sep = '\t',header = None)
repitition_three.columns = ['time','sense_one', 'sense_two', 'sense_three','sense_four','sense_five',
                            'sense_six','sense_seven','sense_eight']
repitition_three.head()
```

Out[24]:

	time	sense_one	sense_two	sense_three	sense_four	sense_five	sense_six	sense_seven	sense_eight
0	0.00	44.11	21.83	24.66	8.04	80.85	49.28	65.38	8.53
1	0.01	44.11	21.83	24.66	8.04	80.58	49.28	65.38	8.53
2	0.02	44.11	21.80	24.66	8.04	80.58	49.16	65.19	8.53
3	0.03	44.11	21.80	24.66	8.04	80.58	49.16	65.38	8.54
4	0.04	44.11	21.83	24.66	8.04	80.58	49.16	65.19	8.53

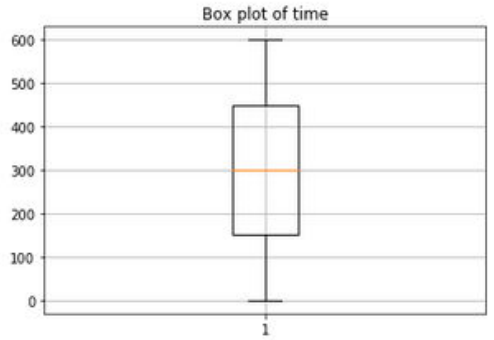
In [25]:
```
repitition_four = pd.read_csv(r'B1_GCO_F040_R4.txt',sep = '\t',header = None)
repitition_four.columns = ['time','sense_one', 'sense_two', 'sense_three','sense_four','sense_five',
                            'sense_six','sense_seven','sense_eight']
repitition_four.head()
```

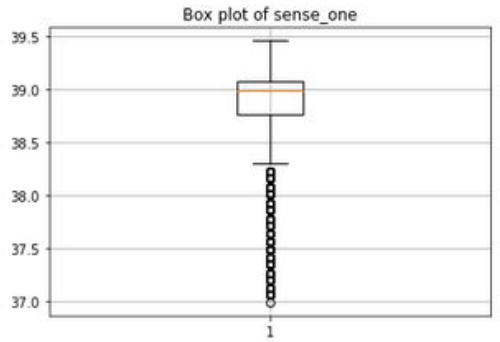
Out[25]:

	time	sense_one	sense_two	sense_three	sense_four	sense_five	sense_six	sense_seven	sense_eight
0	0.00	46.47	23.05	26.07	7.98	85.89	50.90	69.01	8.14
1	0.01	46.58	23.05	26.16	7.98	85.89	50.90	69.21	8.13
2	0.02	46.47	23.05	26.16	7.98	85.89	50.90	69.21	8.13
3	0.03	46.47	23.09	26.16	7.98	85.89	50.78	69.01	8.13
4	0.04	46.47	23.05	26.16	7.98	85.89	50.90	69.01	8.13

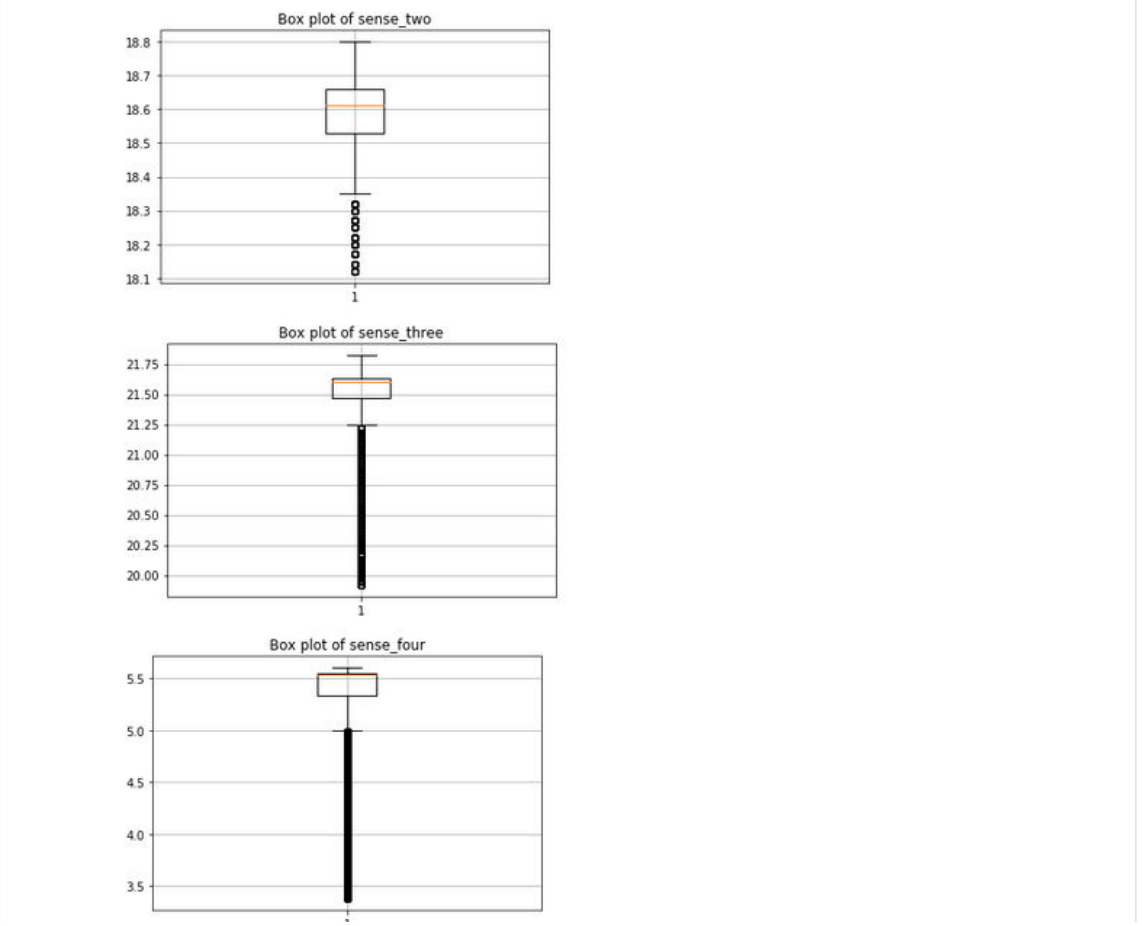
Data cleaning for repitition1, gas CO at 40ppm

In [26]:
```
import matplotlib.pyplot as plt

for key in repitition_one.keys():
    plt.boxplot(repitition_one[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ 'board_one_CO_at_40ppm.jpeg')
```

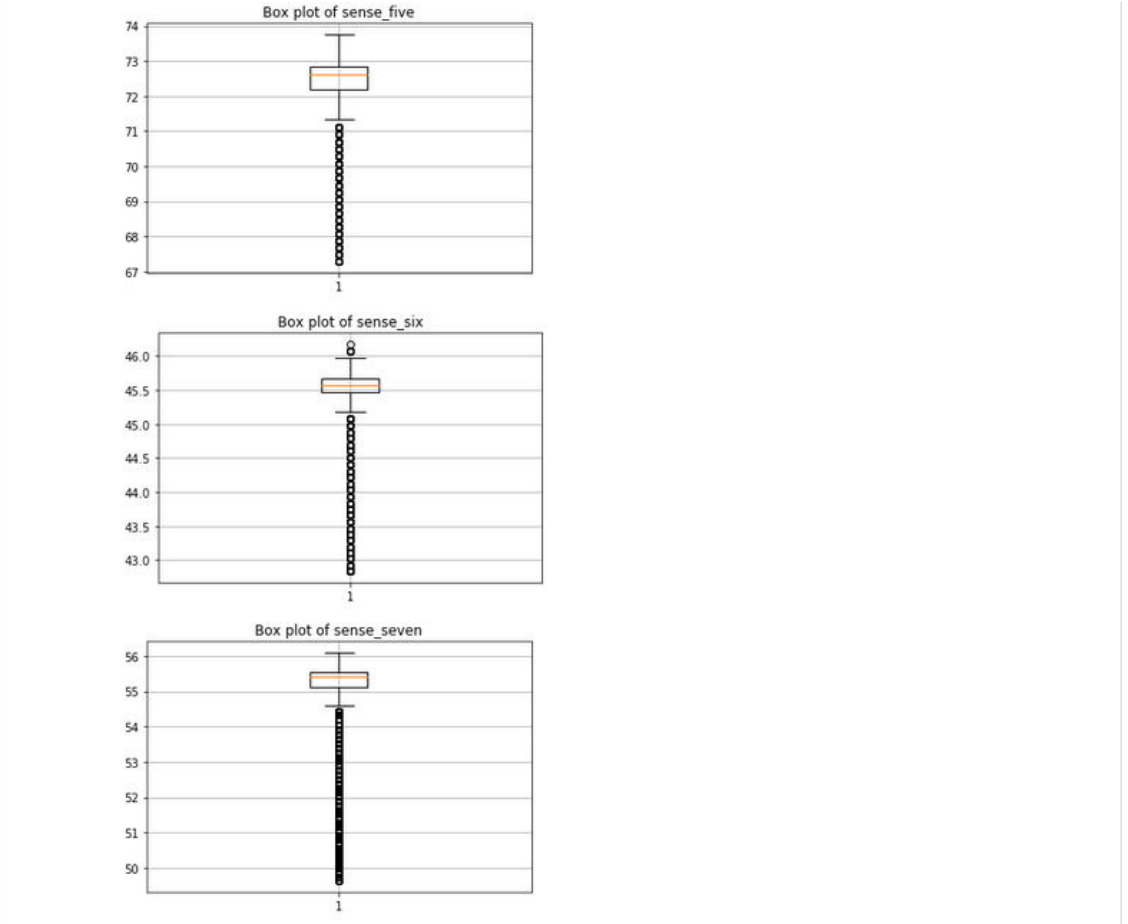




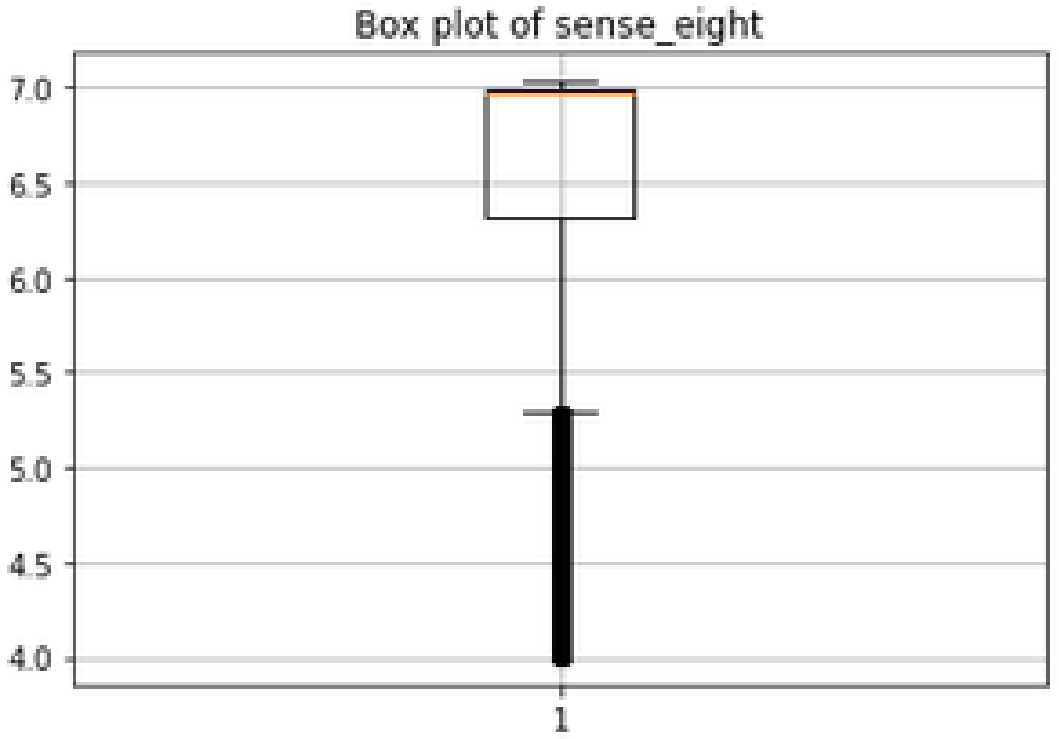
Data cleaning for repitition1, gas CO at 40ppm



Data cleaning for repitition1 gas CO at 40ppm



Data cleaning for repititon1 gas CO at 40ppm

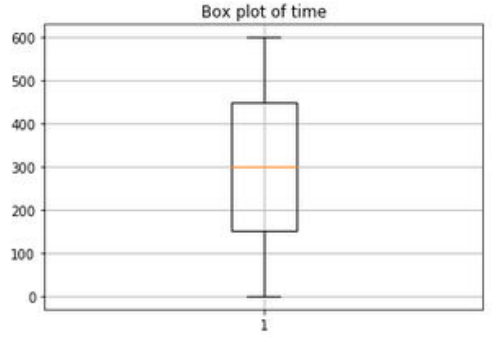


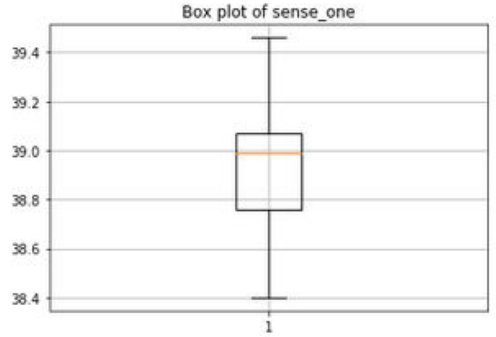
Removing outliers in repitition1

```
In [27]: repitition_one['sense_one'][repitition_one['sense_one'] < 38.4] = 38.4

In [28]: repitition_one['sense_two'][repitition_one['sense_two'] <= 18.36] = 18.36

In [29]: repitition_one['sense_three'][repitition_one['sense_three'] < 21.25] = 21.25

In [30]: repitition_one['sense_four'][repitition_one['sense_four'] <= 5.0 ]= 5.0

In [31]: repitition_one['sense_five'][repitition_one['sense_five'] <= 71.5] = 71.5

In [40]: repitition_one['sense_six'][repitition_one['sense_six'] < 45.20] = 45.20
         repitition_one['sense_six'][repitition_one['sense_six'] > 45.96] = 45.96

In [33]: repitition_one['sense_seven'][repitition_one['sense_seven'] < 54.6] = 54.6

In [34]: repitition_one['sense_eight'][repitition_one['sense_eight'] <5.3] = 5.3
```

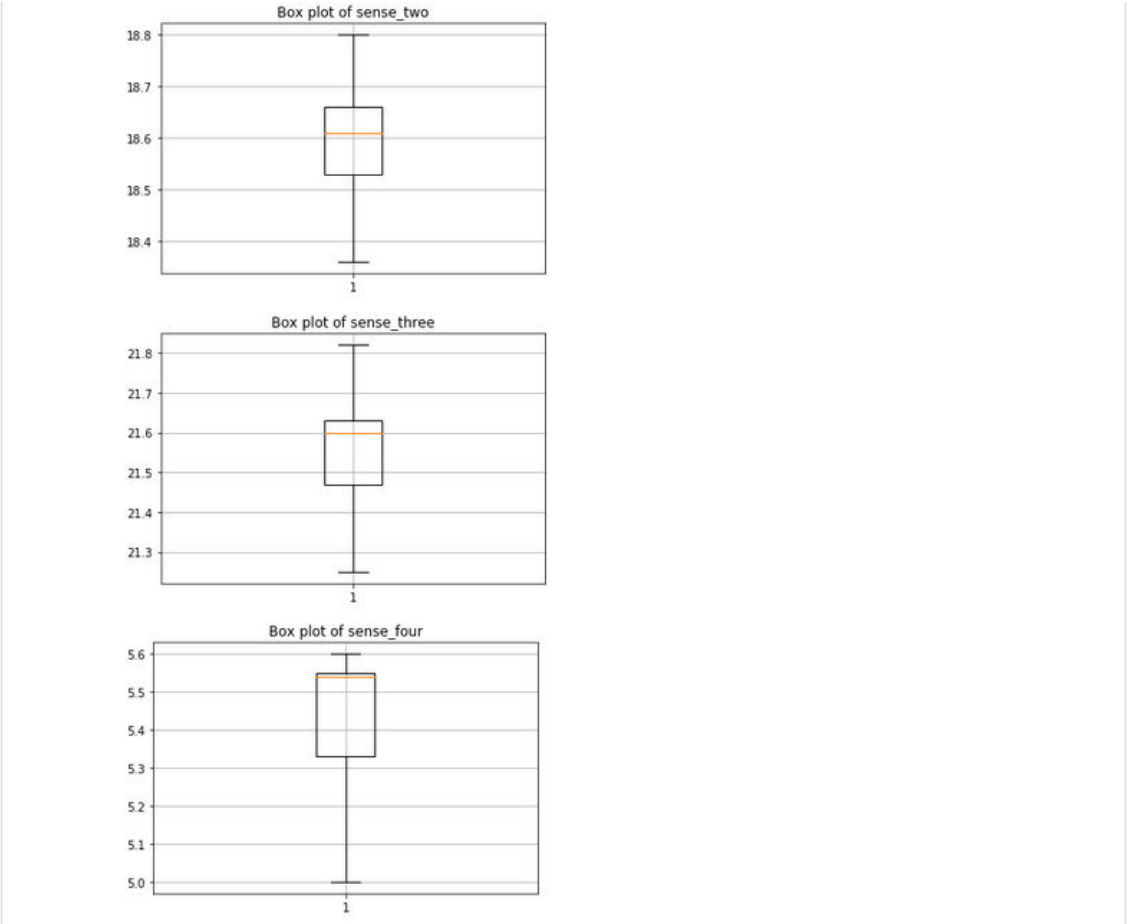
Data visualization of repitition1

In [41]:
```
import matplotlib.pyplot as plt

for key in repitition_one.keys():
    plt.boxplot(repitition_one[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ '_no_outliers_board_one_CO_at_40ppm.jpeg')
```

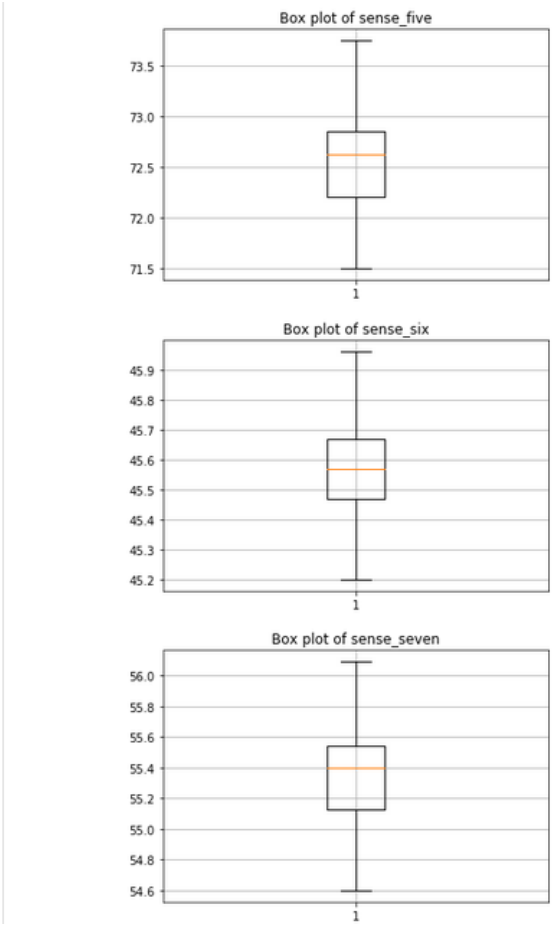




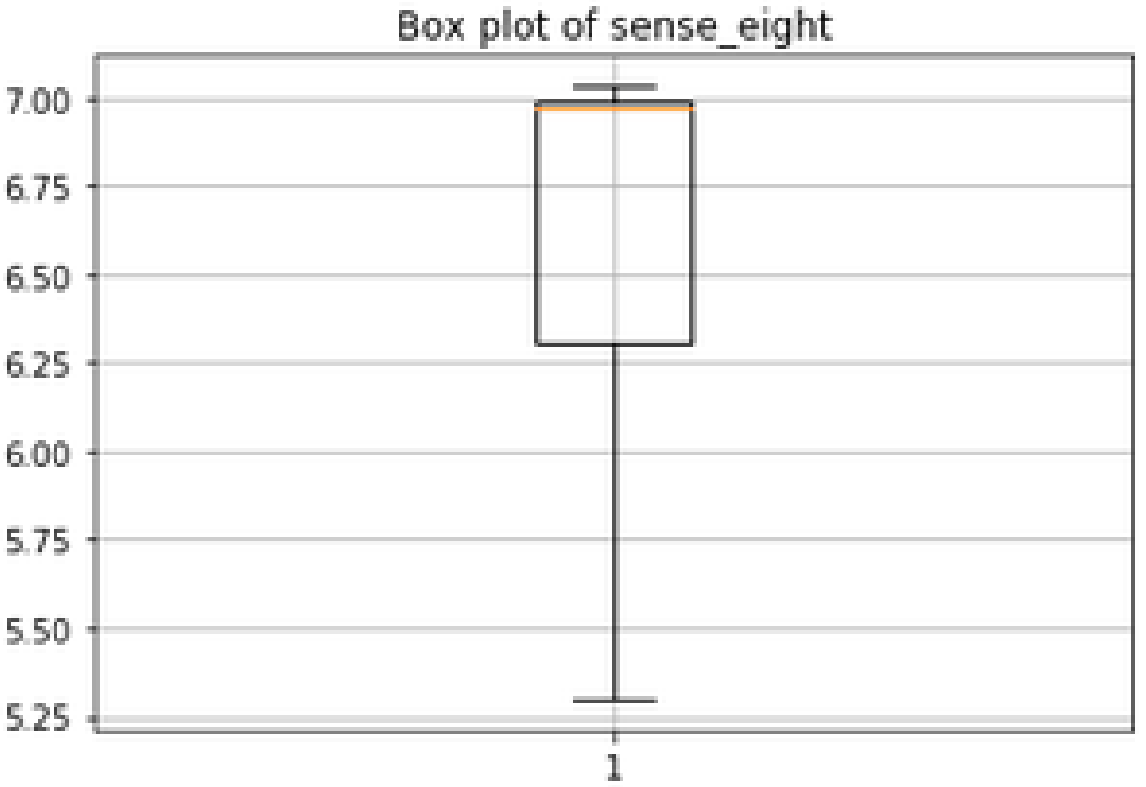
Data visualization after removing the outlier in repitition1



Data visualization after removing the outlier in repitition1

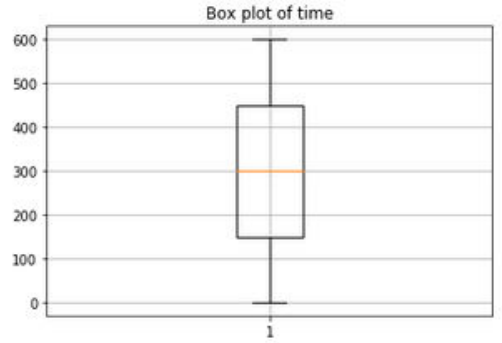


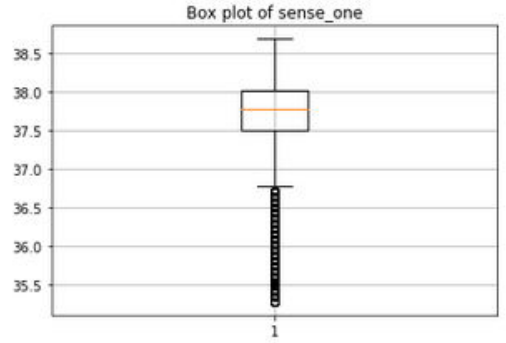
Data visualization after removing the outlier in repitition1



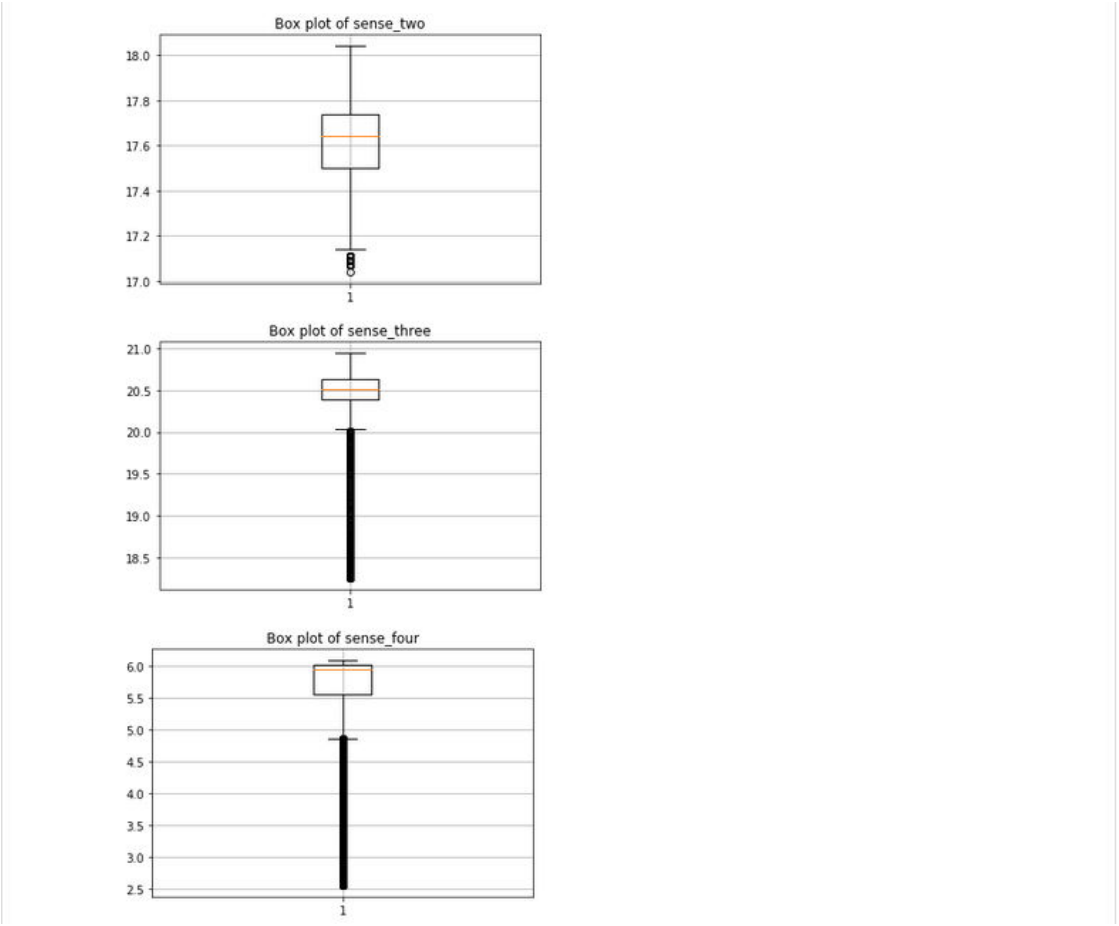
Data visualization of repitition2 with outliers

```
In [29]: import matplotlib.pyplot as plt

for key in repitition_two.keys():
    plt.boxplot(repitition_two[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ 'board_onerepiition2_CO_at_90ppm.jpeg')
```

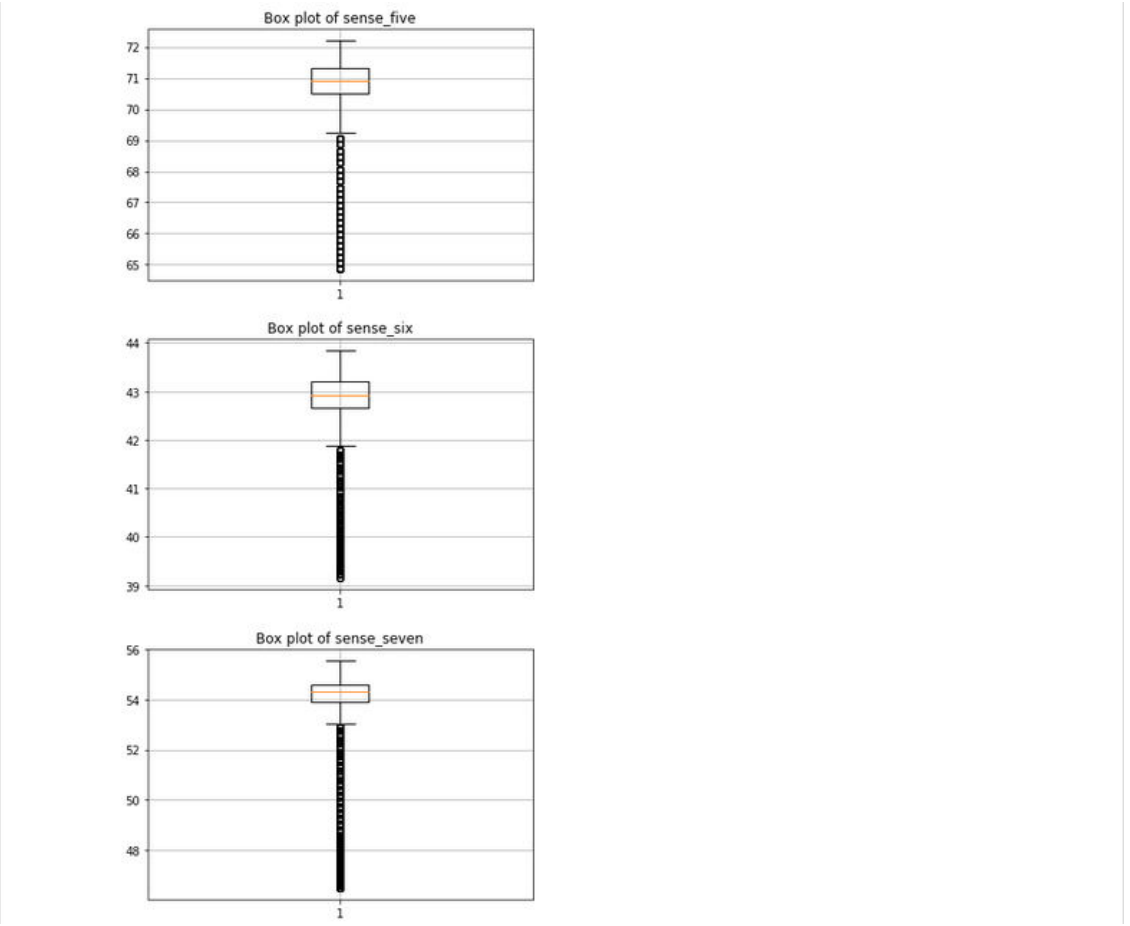




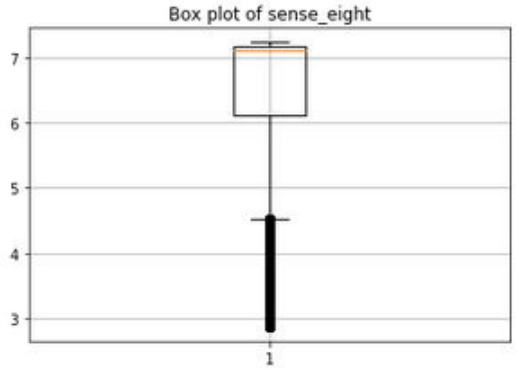
Data visualization of repitition2 with outliers



Data visualization of repitition2 with outliers



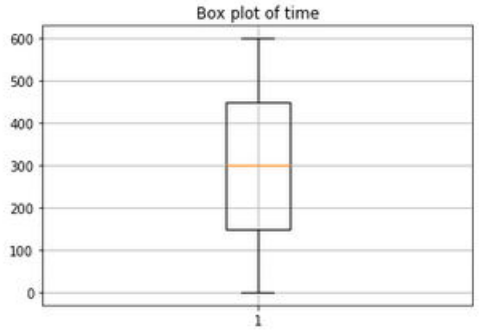
Data visualization of repitition2 with outliers

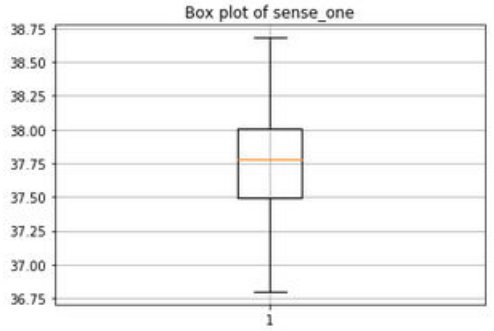


In [30]:
```
def complete_2():
    repitition_two['sense_one'][repitition_two['sense_one'] < 36.8] = 36.8

    repitition_two['sense_two'][repitition_two['sense_two'] < 17.19] = 17.19

    repitition_two['sense_three'][repitition_two['sense_three'] < 20.1] = 20.1

    repitition_two['sense_four'][repitition_two['sense_four'] < 4.9 ]= 4.9

    repitition_two['sense_five'][repitition_two['sense_five'] < 69.5] = 69.5

    repitition_two['sense_six'][repitition_two['sense_six'] < 42] = 42

    repitition_two['sense_seven'][repitition_two['sense_seven'] < 53.2]= 53.2

    repitition_two['sense_eight'][repitition_two['sense_eight'] <4.7] = 4.7

    return

complete_2()
```

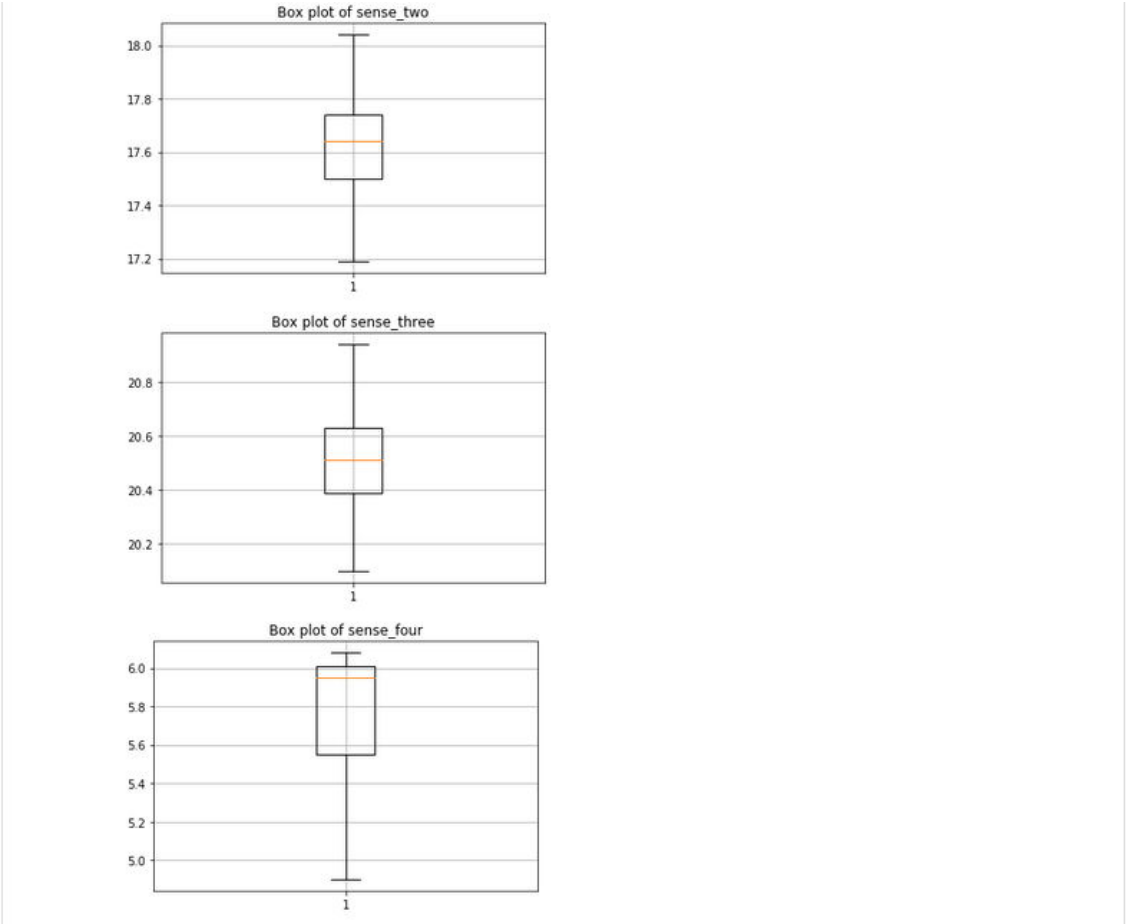
Data visualization of repitition2 without outliers

In [31]:
```
import matplotlib.pyplot as plt

for key in repitition_two.keys():
    plt.boxplot(repitition_two[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ '_no_outliers_repitition2_board_one_CO_at_90ppm.jpeg')
```

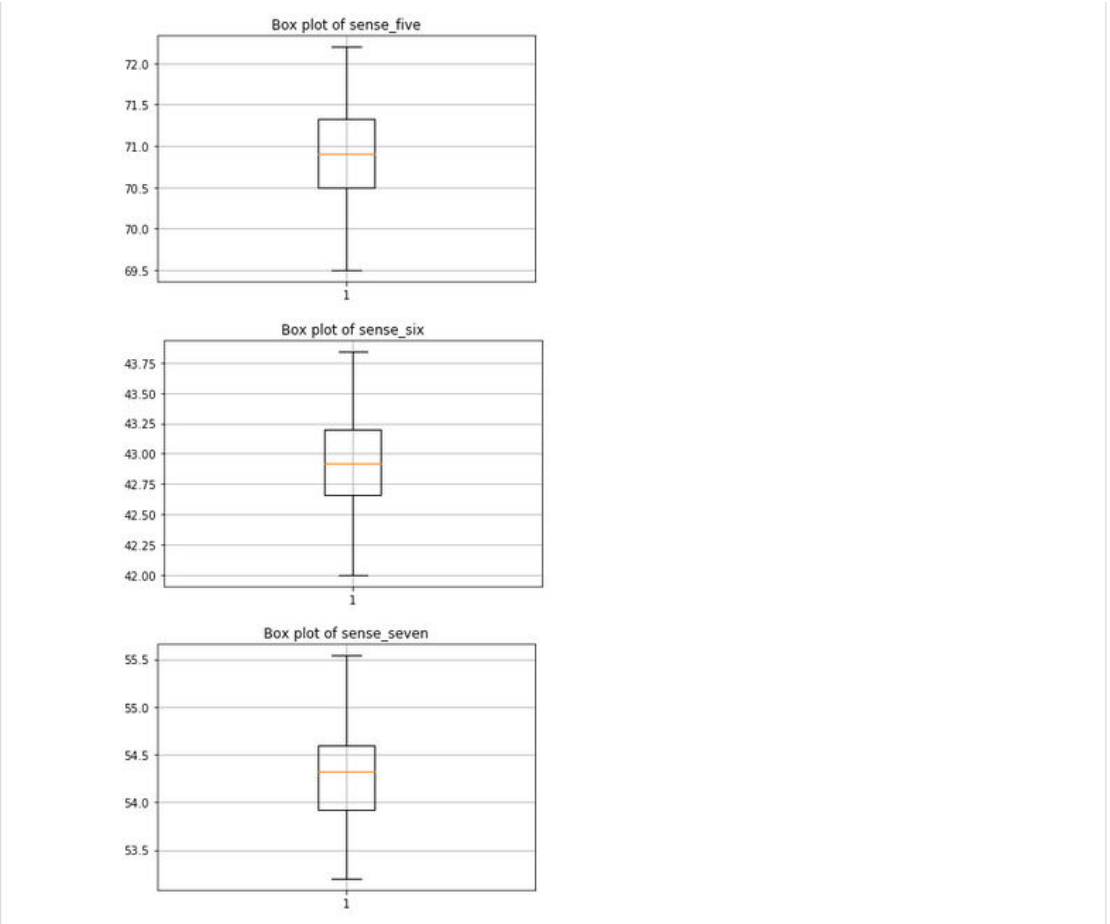




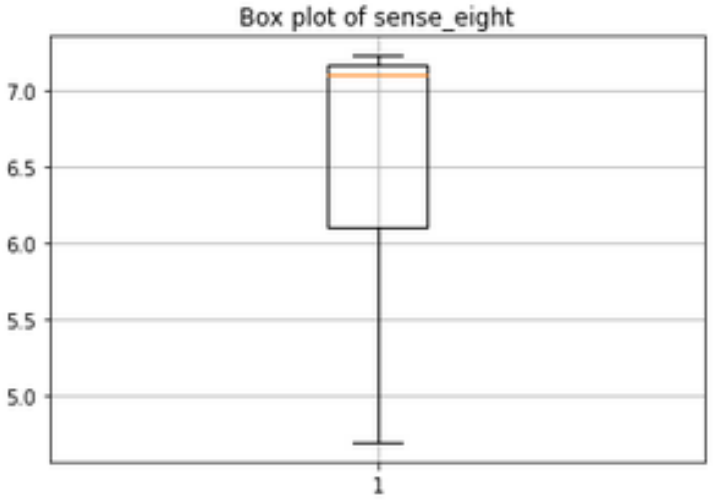
Data visualization of repitition2 without outliers



Data visualiztion of repitition2 without outliers

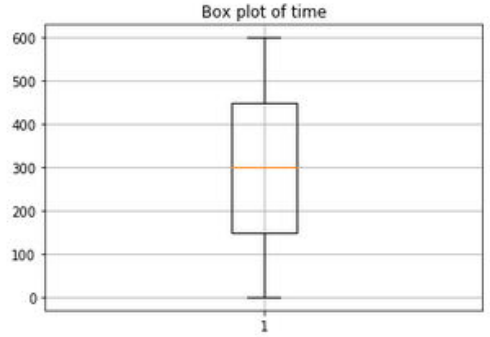


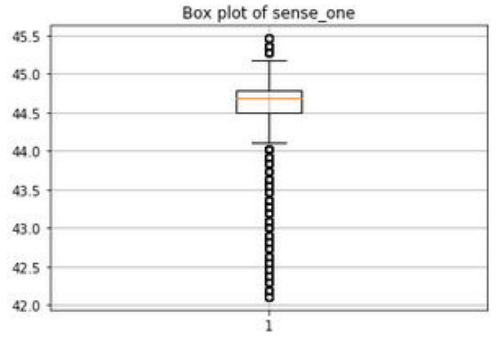
Data visualization of repitition2 without outliers



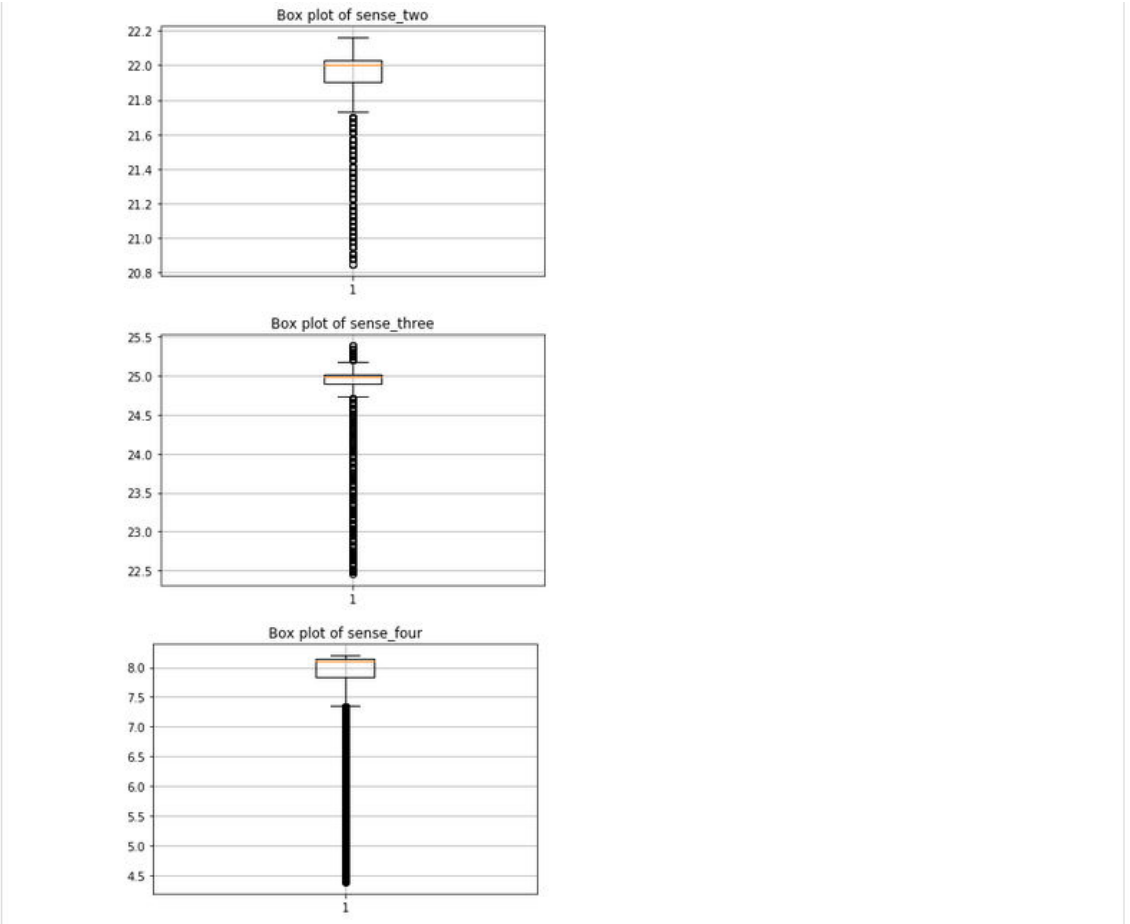
Data visualization of repitition3 with outliers

In [32]:
```
import matplotlib.pyplot as plt

for key in repitition_three.keys():
    plt.boxplot(repitition_three[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ 'board_onerepiition3_CO_at_90ppm.jpeg')
```

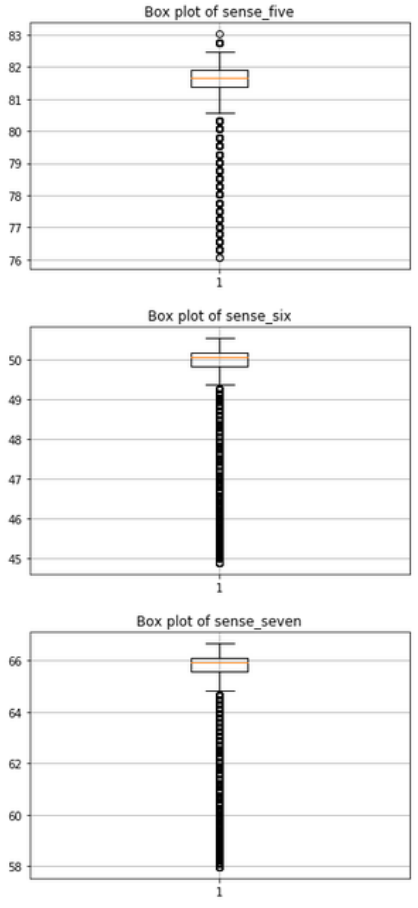




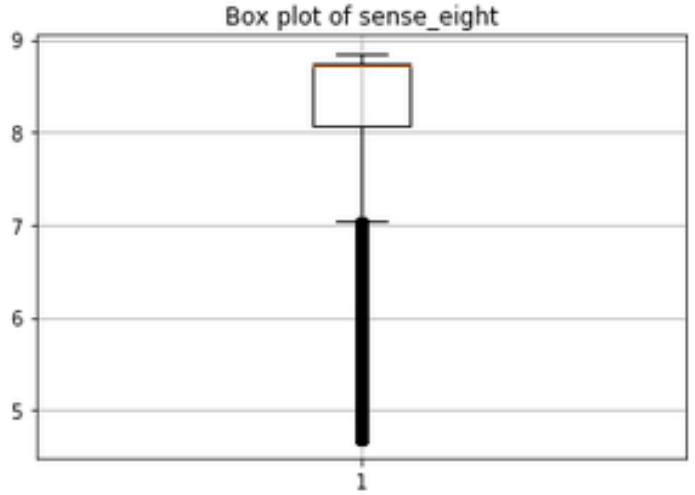
Data visualization of repitition3 with outliers



Data visualization of repitition3 with outliers



Data visualization of repitition3 with outliers

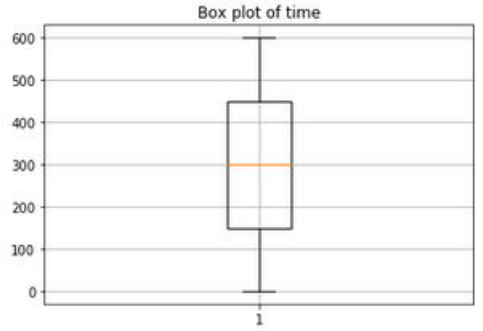


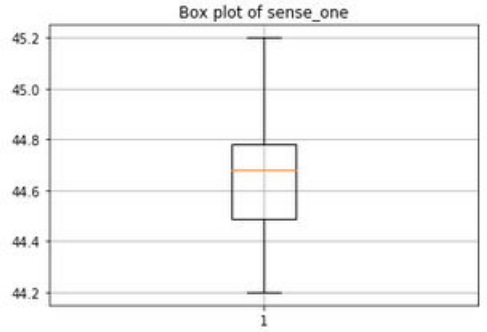
Preprocessing repitition3

In [33]:
```
def complete_3():
    repitition_three['sense_one'][repitition_three['sense_one'] < 44.2] = 44.2
    repitition_three['sense_one'][repitition_three['sense_one'] > 45.2] = 45.2

    repitition_three['sense_two'][repitition_three['sense_two'] < 21.74] = 21.74

    repitition_three['sense_three'][repitition_three['sense_three'] < 24.8] = 24.8
    repitition_three['sense_three'][repitition_three['sense_three'] > 25.1] = 25.1

    repitition_three['sense_four'][repitition_three['sense_four'] < 7.4 ]= 7.4

    repitition_three['sense_five'][repitition_three['sense_five'] < 80.7] = 80.7
    repitition_three['sense_five'][repitition_three['sense_five'] > 82.3] = 82.3

    repitition_three['sense_six'][repitition_three['sense_six'] < 49.7] = 49.7

    repitition_three['sense_seven'][repitition_three['sense_seven'] < 65]= 65

    repitition_three['sense_eight'][repitition_three['sense_eight'] <7.1] = 7.1

    return

complete_3()
```

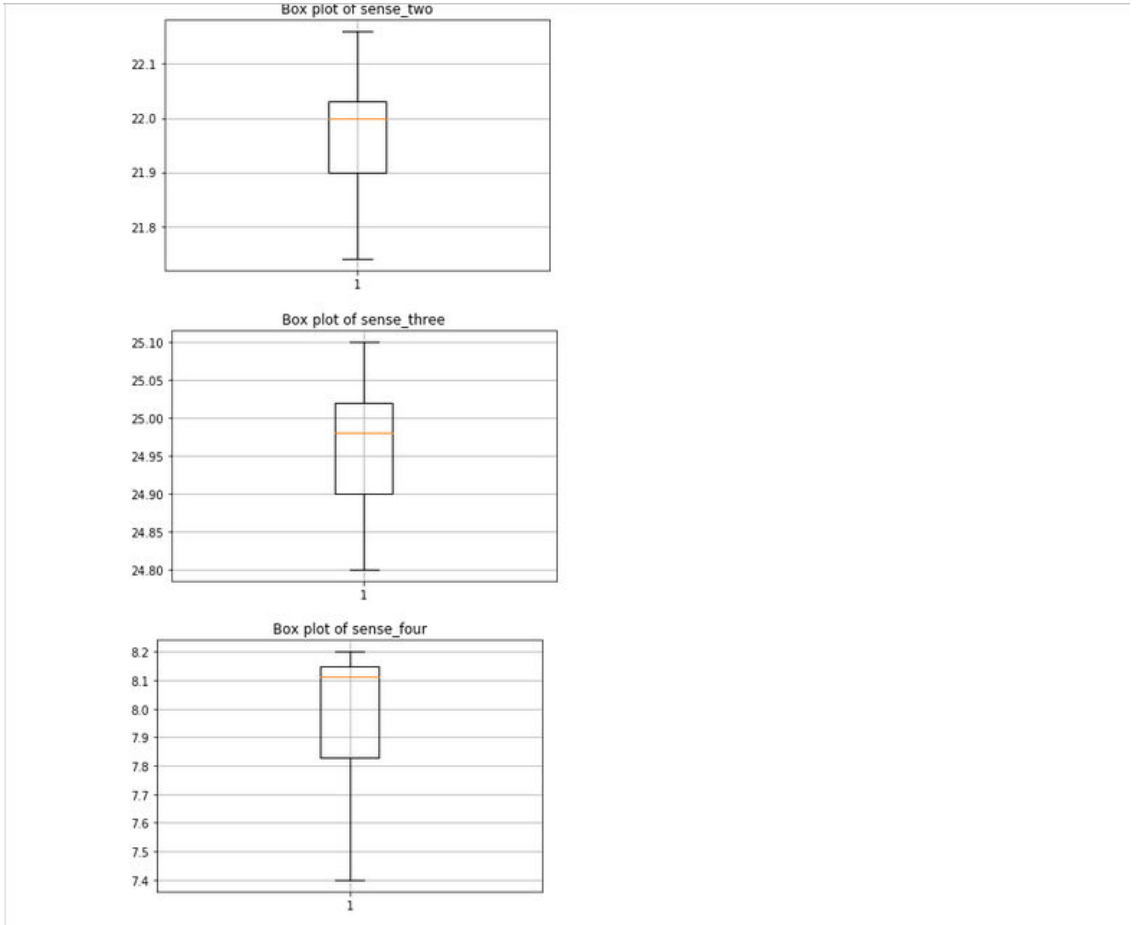
Data visualization of repitition3 without outliers

In [34]:
```
import matplotlib.pyplot as plt

for key in repitition_three.keys():
    plt.boxplot(repitition_three[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ 'board_one_no_outliers_repiition3_CO_at_90ppm.jpeg
')
```

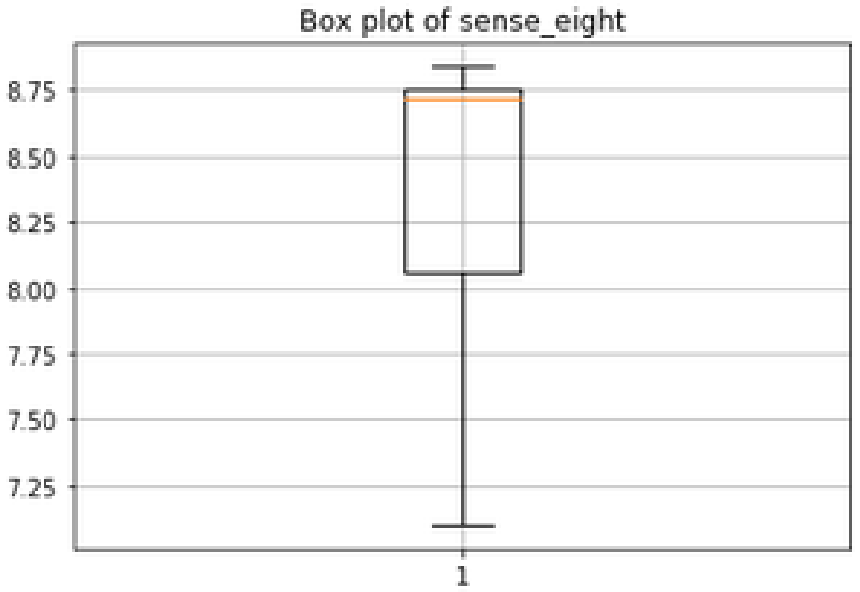




Data visualization of repitition3 without outliers

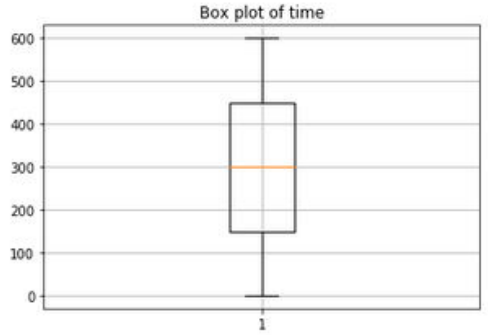


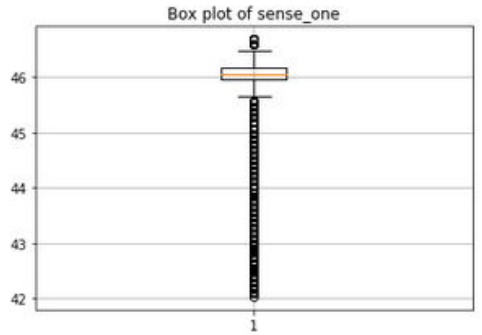
Data visualization of repitition3 without outliers



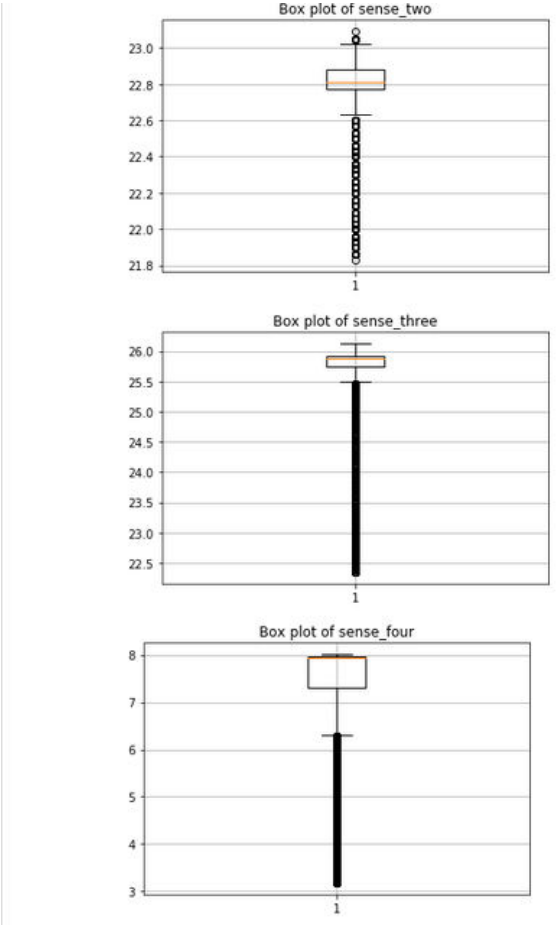
Data visualization of repitition4 with outliers

In [35]:
```
import matplotlib.pyplot as plt

for key in repitition_four.keys():
    plt.boxplot(repitition_four[key])
    plt.grid('on')
    plt.title('Box plot of '+str(key))
    plt.show()
    plt.savefig('Box_plot_of_'+ str(key)+ 'board_onerepiition4_CO_at_90ppm.jpeg')
```

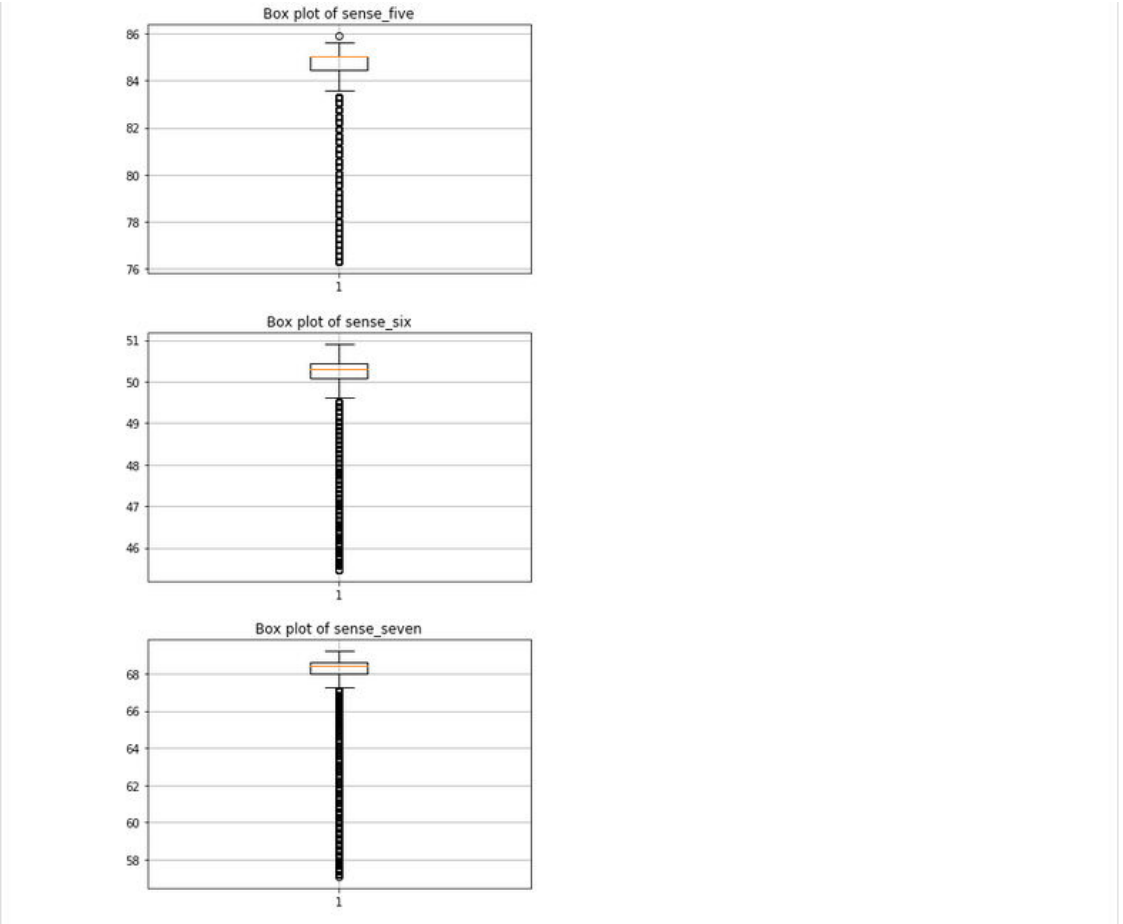




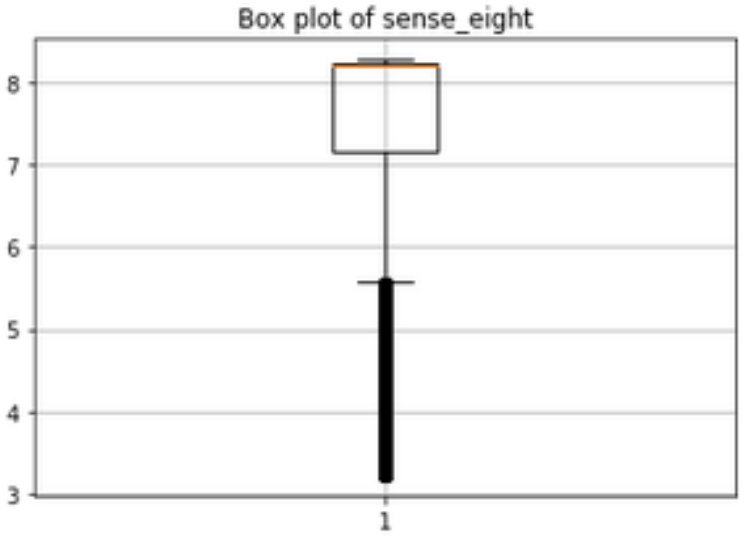
Data visualization of repitition4 with outliers



Data visualization of repitition4 with outliers



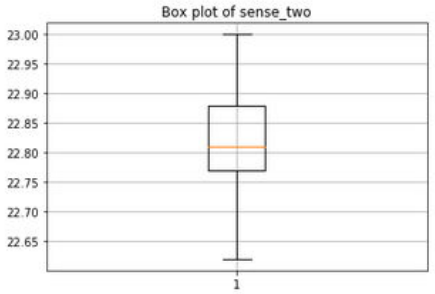
Data visualization of repitition4 with outliers

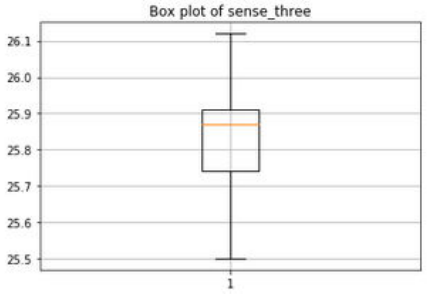


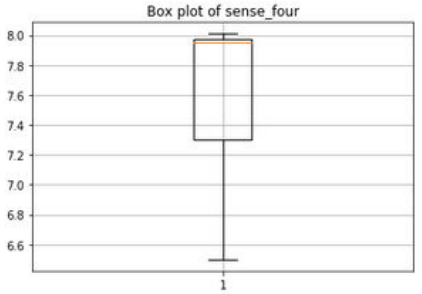
Data visualization of repitition4 preprocessing

In [36]:
```
def complete_4():
    repitition_four['sense_one'][repitition_four['sense_one'] < 45.8] = 45.8
    repitition_four['sense_one'][repitition_four['sense_one'] > 46.4] = 46.4

    repitition_four['sense_two'][repitition_four['sense_two'] < 22.62] = 22.62
    repitition_four['sense_two'][repitition_four['sense_two'] > 23.00] = 23.00

    repitition_four['sense_three'][repitition_four['sense_three'] < 25.50] = 25.50

    repitition_four['sense_four'][repitition_four['sense_four'] < 6.5 ]= 6.5

    repitition_four['sense_five'][repitition_four['sense_five'] < 83.8] = 83.8
    repitition_four['sense_five'][repitition_four['sense_five'] > 85.7] = 85.7

    repitition_four['sense_six'][repitition_four['sense_six'] < 49.7] = 49.7

    repitition_four['sense_seven'][repitition_four['sense_seven'] < 67.2]= 67.2

    repitition_four['sense_eight'][repitition_four['sense_eight'] <5.7] = 5.7

    return

complete_4()
```

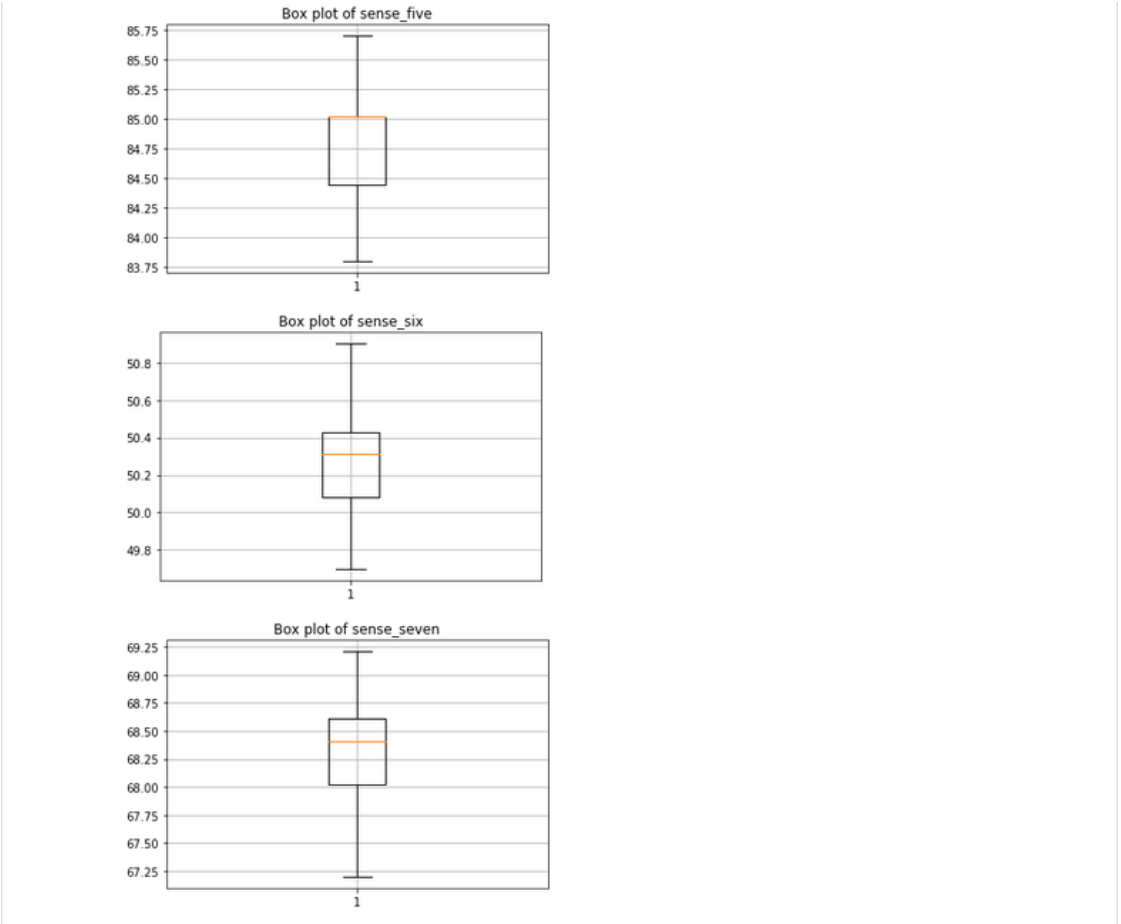
Data visualization of repitition4 without outliers



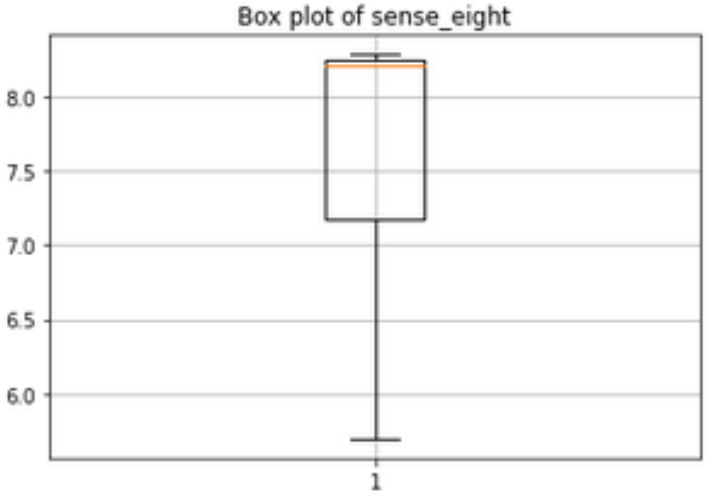




Data visualization of repitition4 without outliers



Data visualization of repitition4 sensor 8 without outliers, merging of all repitittions




```
In [38]: final_repitition = (repitition_one + repitition_two+repitition_three + repitition_four) / 4
```

```
In [39]: final_repitition.to_csv('Repitition_CO_at_90_ppm.csv')
```

Data cleaning

- This same task was performed for on all Text files involved:

B1_FCO_70_(R1, R2,R3,R4).txt - merged into merged_FCO_70.txt
B1_FCO_80_(R1,R2,R3,R4).txt - merged into merged_FCO_80.txt
B1_FCO_90_(R1,R2,R3,R4).txt – merged into merged_FCO_90.txt

B1_FEA_20_(R1,R2,R3,R4).txt – merged into merged_FEA_20.txt
B1_FEA_30_(R1,R2,R3,R4).txt – merged into merged_FEA_30.txt
B1_FEA_40_(R1,R2,R3,R4).txt – merged into merged_FEA_40.txt

B1_FEY_40_(R1,R2,R3,R4).txt – merged into merged_FEY_40.txt
B1_FEY_50_(R1,R2,R3,R4).txt – merged into merged_FEY_50.txt
B1_FEY_60_(R1,R2,R3,R4).txt – merged into merged_FEY_60.txt

B1_FME_60_(R1,R2,R3,R4).txt – merged into merged_FME_60.txt
B1_FME_70_(R1,R2,R3,R4).txt – merged into merged_FME_70.txt

Data merging

In [62]:
```
import os
os.listdir()
```

Out[62]:
```
['.ipynb_checkpoints',
 'create_dataset.ipynb',
 'data visualization.ipynb',
 'merged_preprocess_data.csv',
 'model_building.ipynb',
 'Repitition_CO_at_70_ppm.csv',
 'Repitition_CO_at_80_ppm.csv',
 'Repitition_CO_at_90_ppm.csv',
 'Repitition_EA_at_20_ppm.csv',
 'Repitition_EA_at_30_ppm.csv',
 'Repitition_EA_at_40_ppm.csv',
 'Repitition_Ey_at_40_ppm.csv',
 'Repitition_Ey_at_50_ppm.csv',
 'Repitition_Ey_at_60_ppm.csv',
 'Repitition_ME_at_60_ppm.csv',
 'Repitition_ME_at_70_ppm.csv',
 'Repitition_ME_at_80_ppm.csv']
```

Data merging

In [63]:
```
import pandas as pd
co_at_70_ppm = pd.read_csv('Repitition_CO_at_70_ppm.csv')
co_at_80_ppm = pd.read_csv('Repitition_CO_at_80_ppm.csv')
co_at_90_ppm = pd.read_csv('Repitition_CO_at_90_ppm.csv')
```

In [64]:
```
ea_at_20_ppm = pd.read_csv('Repitition_EA_at_20_ppm.csv')
ea_at_30_ppm = pd.read_csv('Repitition_EA_at_30_ppm.csv')
ea_at_40_ppm = pd.read_csv('Repitition_EA_at_40_ppm.csv')
```

In [65]:
```
ey_at_40_ppm = pd.read_csv('Repitition_EY_at_40_ppm.csv')
ey_at_50_ppm = pd.read_csv('Repitition_EY_at_50_ppm.csv')
ey_at_60_ppm = pd.read_csv('Repitition_EY_at_60_ppm.csv')
```

In [66]:
```
me_at_60_ppm = pd.read_csv('Repitition_ME_at_60_ppm.csv')
me_at_70_ppm = pd.read_csv('Repitition_ME_at_70_ppm.csv')
me_at_80_ppm = pd.read_csv('Repitition_ME_at_80_ppm.csv')
```

In [67]:
```
co_at_70_ppm['target'] = 1.0
co_at_80_ppm['target'] = 1.0
co_at_90_ppm['target'] = 1.0
```

In [68]:
```
ea_at_20_ppm['target'] = 2.0
ea_at_30_ppm['target'] = 2.0
ea_at_40_ppm['target'] = 2.0
```

In [69]:
```
ey_at_40_ppm['target'] = 3.0
ey_at_50_ppm['target'] = 3.0
ey_at_60_ppm['target'] = 3.0
```

In [70]:
```
me_at_60_ppm['target'] = 4.0
me_at_70_ppm['target'] = 4.0
me_at_80_ppm['target'] = 4.0
```

In [71]:
```
co_at_70_ppm = co_at_70_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
```

Data merging

In [72]:
```
co_at_70_ppm.head()
```

Out[72]:

	sense_one	sense_two	sense_three	sense_four	sense_five	sense_six	sense_seven	sense_eight	target
0	41.935	20.2800	23.2900	6.875	77.7575	47.1925	61.2100	7.7250	1.0
1	41.960	20.3000	23.3075	6.875	77.7075	47.2700	61.2100	7.7225	1.0
2	41.935	20.2875	23.2800	6.875	77.7025	47.1625	61.2100	7.7300	1.0
3	41.915	20.2800	23.2900	6.875	77.7025	47.2625	61.1775	7.7250	1.0
4	41.960	20.3025	23.2900	6.875	77.7575	47.2450	61.1600	7.7250	1.0

In [73]:
```
co_at_80_ppm = co_at_80_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
co_at_90_ppm = co_at_90_ppm.drop(['Unnamed: 0', 'time'], axis = 1)

ea_at_20_ppm = ea_at_20_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
ea_at_30_ppm = ea_at_30_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
ea_at_40_ppm = ea_at_40_ppm.drop(['Unnamed: 0', 'time'], axis = 1)

ey_at_40_ppm = ey_at_40_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
ey_at_50_ppm = ey_at_50_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
ey_at_60_ppm = ey_at_60_ppm.drop(['Unnamed: 0', 'time'], axis = 1)


me_at_60_ppm = me_at_60_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
me_at_70_ppm = me_at_70_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
me_at_80_ppm = me_at_80_ppm.drop(['Unnamed: 0', 'time'], axis = 1)
```

In [74]:
```
me_at_80_ppm['target'][:10]
```

Out[74]:
```
0    4.0
1    4.0
2    4.0
3    4.0
4    4.0
5    4.0
6    4.0
7    4.0
8    4.0
9    4.0
Name: target, dtype: float64
```

Data merging

In [89]:
```
def concatenate_data(key):
    value_one = np.append(co_at_80_ppm[key],co_at_90_ppm[key])
    value_one_one = np.append(ea_at_30_ppm[key], ea_at_40_ppm[key])
    value_two = np.append(ea_at_20_ppm[key], value_one_one)
    value_two_two = np.append(ey_at_40_ppm[key], ey_at_50_ppm[key])
    value_three = np.append(ey_at_60_ppm[key],value_two_two)
    value_three_three = np.append(me_at_70_ppm[key],me_at_80_ppm[key])
    value_four = np.append(me_at_60_ppm[key],value_three_three)

    val_1 = np.append(value_two, value_three)
    value = np.append(value_one,val_1)
    final = np.append(value, value_four)
    return final

sense_one = np.append(co_at_70_ppm['sense_one'],concatenate_data(key = 'sense_one'))
print(sense_one.shape)
sense_two = np.append(co_at_70_ppm['sense_two'],concatenate_data(key = 'sense_two'))
print(sense_two.shape)
sense_three = np.append(co_at_70_ppm['sense_three'],concatenate_data(key = 'sense_three'))
print(sense_three.shape)
sense_four = np.append(co_at_70_ppm['sense_four'],concatenate_data(key = 'sense_four'))
print(sense_four.shape)
sense_five = np.append(co_at_70_ppm['sense_five'],concatenate_data(key = 'sense_five'))
print(sense_five.shape)
sense_six= np.append(co_at_70_ppm['sense_six'],concatenate_data(key = 'sense_six'))
print(sense_six.shape)
sense_seven = np.append(co_at_70_ppm['sense_seven'],concatenate_data(key = 'sense_seven'))
print(sense_seven.shape)
sense_eight = np.append(co_at_70_ppm['sense_eight'],concatenate_data(key = 'sense_eight'))
print(sense_eight.shape)
target = np.append(co_at_70_ppm['target'],concatenate_data(key = 'target'))
print(target.shape)
```

```
(719951,)
(719951,)
(719951,)
(719951,)
(719951,)
(719951,)
(719951,)
(719951,)
(719951,)
```

Data merging

In [90]:
```
target[:10]
```

Out[90]:
```
array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

In [91]:
```
new_dataframe = pd.DataFrame({'sense_one':sense_one,'sense_two':sense_two, 'sense_three':sen
se_three,'sense_four':sense_four,'sense_five':sense_five,
                              'sense_six': sense_six, 'sense_seven': sense_seven,'sense_eight
':sense_eight,'target':target})
new_dataframe.head()
```

Out[91]:

	sense_eight	sense_five	sense_four	sense_one	sense_seven	sense_six	sense_three	sense_two	target
0	7.7250	77.7575	6.875	41.935	61.2100	47.1925	23.2900	20.2800	1.0
1	7.7225	77.7075	6.875	41.960	61.2100	47.2700	23.3075	20.3000	1.0
2	7.7300	77.7025	6.875	41.935	61.2100	47.1625	23.2800	20.2875	1.0
3	7.7250	77.7025	6.875	41.915	61.1775	47.2625	23.2900	20.2800	1.0
4	7.7250	77.7575	6.875	41.960	61.1600	47.2450	23.2900	20.3025	1.0

In [92]:
```
new_dataframe.shape
```

Out[92]:
```
(719951, 9)
```

In [93]:
```
new_dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 719951 entries, 0 to 719950
Data columns (total 9 columns):
sense_eight    698601 non-null float64
sense_five     698601 non-null float64
sense_four     698601 non-null float64
sense_one      698601 non-null float64
sense_seven    698601 non-null float64
sense_six      698601 non-null float64
sense_three    698601 non-null float64
sense_two      698601 non-null float64
target         719951 non-null float64
dtypes: float64(9)
memory usage: 49.4 MB
```

Data merging

In [94]:
```
new_dataframe.isnull().sum()
```

Out[94]:
```
sense_eight    21350
sense_five     21350
sense_four     21350
sense_one      21350
sense_seven    21350
sense_six      21350
sense_three    21350
sense_two      21350
target             0
dtype: int64
```

In [96]:
```
new_data = new_dataframe.dropna(axis = 0, how = 'any')
new_data.shape
```

Out[96]:
```
(698601, 9)
```

In [97]:
```
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 698601 entries, 0 to 719613
Data columns (total 9 columns):
sense_eight    698601 non-null float64
sense_five     698601 non-null float64
sense_four     698601 non-null float64
sense_one      698601 non-null float64
sense_seven    698601 non-null float64
sense_six      698601 non-null float64
sense_three    698601 non-null float64
sense_two      698601 non-null float64
target         698601 non-null float64
dtypes: float64(9)
memory usage: 53.3 MB
```

In [98]:
```
new_data.to_csv('merged_preprocess_data.csv')
```

Model building

- The algorithm used in building our machine learning model are KnearestNeighbor classifier (using 4 numbers of neighbors since we have 4 classes of gases to be classified) and the support vector machine classifier (which we used two types of kernel = ‘rbf’, and ‘linear’).
- All machine learning algorithm used outputed an accuracy of 100 on both train and test set.
- I never expected this high accuracy but this implies that our datas are linearly separable as we can see in the visualization chart and it also a simple data coupled with the fact that it was well preprocessed without outliers.

Model building

- Also before building the model, our merged data containing over 700,000 instances and 8 sensing columns and a target column (a total of 9), we had to scale and normalize or data.
- The scaler used was based on standard deviation of each row involved.
- Our data is also normalized between the value of 0 and 1.

Model building

In [80]:
```
import pandas as pd
read_data = pd.read_csv('merged_preprocess_data.csv')
```

In [81]:
```
read_data.shape
```

Out[81]:
```
(698601, 10)
```

In [82]:
```
read_data.head()
```

Out[82]:

	Unnamed: 0	sense_eight	sense_five	sense_four	sense_one	sense_seven	sense_six	sense_three	sense_two	target
0	0	7.7250	77.7575	6.875	41.935	61.2100	47.1925	23.2900	20.2800	1.0
1	1	7.7225	77.7075	6.875	41.960	61.2100	47.2700	23.3075	20.3000	1.0
2	2	7.7300	77.7025	6.875	41.935	61.2100	47.1625	23.2800	20.2875	1.0
3	3	7.7250	77.7025	6.875	41.915	61.1775	47.2625	23.2900	20.2800	1.0
4	4	7.7250	77.7575	6.875	41.960	61.1600	47.2450	23.2900	20.3025	1.0

In [83]:
```
read_data = read_data.drop('Unnamed: 0', axis = 1)
```

In [84]:
```
read_data.info()
```

Model building

In [84]:
```
read_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 698601 entries, 0 to 698600
Data columns (total 9 columns):
sense_eight    698601 non-null float64
sense_five     698601 non-null float64
sense_four     698601 non-null float64
sense_one      698601 non-null float64
sense_seven    698601 non-null float64
sense_six      698601 non-null float64
sense_three    698601 non-null float64
sense_two      698601 non-null float64
target         698601 non-null float64
dtypes: float64(9)
memory usage: 48.0 MB
```

In [85]:
```
read_data.isnull().sum()
```

Out[85]:
```
sense_eight    0
sense_five     0
sense_four     0
sense_one      0
sense_seven    0
sense_six      0
sense_three    0
sense_two      0
target         0
dtype: int64
```

In [86]:
```
final_data = read_data.dropna(axis = 0, how = 'any')
final_data.head()
```

Out[86]:

	sense_eight	sense_five	sense_four	sense_one	sense_seven	sense_six	sense_three	sense_two	target
0	7.7250	77.7575	6.875	41.935	61.2100	47.1925	23.2900	20.2800	1.0
1	7.7225	77.7075	6.875	41.960	61.2100	47.2700	23.3075	20.3000	1.0
2	7.7300	77.7025	6.875	41.935	61.2100	47.1625	23.2800	20.2875	1.0
3	7.7250	77.7025	6.875	41.915	61.1775	47.2625	23.2900	20.2800	1.0
4	7.7250	77.7575	6.875	41.960	61.1600	47.2450	23.2900	20.3025	1.0

Model building

In [87]:
```
final_data.isnull().sum()
```

Out[87]:
```
sense_eight    0
sense_five     0
sense_four     0
sense_one      0
sense_seven    0
sense_six      0
sense_three    0
sense_two      0
target         0
dtype: int64
```

In [88]:
```
final_data.shape
```

Out[88]:
```
(698601, 9)
```

In [89]:
```
this = final_data.iloc[2][:]
True in [isinstance(k, str) for k in this.values]
```

Out[89]:
```
False
```

In [90]:
```
X = final_data.drop('target', axis = 1)
```

In [91]:
```
y = final_data['target']
```

In [92]:
```
import sklearn as sk
```

In [93]:
```
from sklearn.model_selection import train_test_split
```

In [94]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size = 0.80, random_state =
500)
```

In [95]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[95]:
```
((558880, 8), (139721, 8), (558880,), (139721,))
```

In [96]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
```

In [97]:
```
std_scaler = StandardScaler()
```

Model building

In [96]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
```

In [97]:
```
std_scaler = StandardScaler()
max_scaler = MinMaxScaler()
norm = Normalizer()

x_train_scale = std_scaler.fit_transform(x_train)
x_train_max = max_scaler.fit_transform(x_train_scale)
x_train_norm = norm.fit_transform(x_train_max)

x_test_scale = std_scaler.fit_transform(x_test)
x_test_max = max_scaler.fit_transform(x_test_scale)
x_test_norm = norm.fit_transform(x_test_max)
```

In [99]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 4)
knn.fit(x_train_scale, y_train)
knn.score(x_train_scale, y_train)
```

Out[99]:
```
1.0
```

In [100]:
```
test_predict = knn.predict(x_test_scale)
from sklearn.metrics import accuracy_score
accuracy_score(test_predict, y_test)
```

Out[100]:
```
1.0
```

In [103]:
```
test_predict[:20]
```

Out[103]:
```
array([ 4.,  1.,  1.,  3.,  3.,  4.,  4.,  3.,  3.,  4.,  4.,  1.,  3.,
        3.,  4.,  3.,  2.,  1.,  4.,  2.])
```

Model building

In [104]:
```
y_test[:20]
```

Out[104]:
```
571029    4.0
27645     1.0
105523    1.0
399997    3.0
412972    3.0
572680    4.0
520009    4.0
373509    3.0
371632    3.0
663688    4.0
584700    4.0
11809     1.0
349598    3.0
468174    3.0
679064    4.0
490029    3.0
273269    2.0
107658    1.0
574216    4.0
284046    2.0
Name: target, dtype: float64
```

In [105]:
```
from sklearn.svm import SVC
svc = SVC(C = 1.0, kernel = 'linear')
svc.fit(x_train_scale, y_train)
svc.score(x_train_scale, y_train)
```

Out[105]:
```
1.0
```

In [106]:
```
test_predict_svc = svc.predict(x_test_scale)
accuracy_score(test_predict_svc, y_test)
```

Out[106]:
```
1.0
```

In [108]:
```
from sklearn.svm import SVC
svc = SVC(C = 1.0, kernel = 'rbf')
svc.fit(x_train_scale, y_train)
svc.score(x_train_scale, y_train)
```

Out[108]:
```
1.0
```

In [ ]:
```
from sklearn.svm import SVC
svc = SVC(C = 1.0, kernel = 'sigmoid')
svc.fit(x_train_scale, y_train)
svc.score(x_train_scale, y_train)
```

Data visualization and data insight

- We want to visualize the relationship between our sensors to see if they work in like manners or not.

Data visualization and data insight

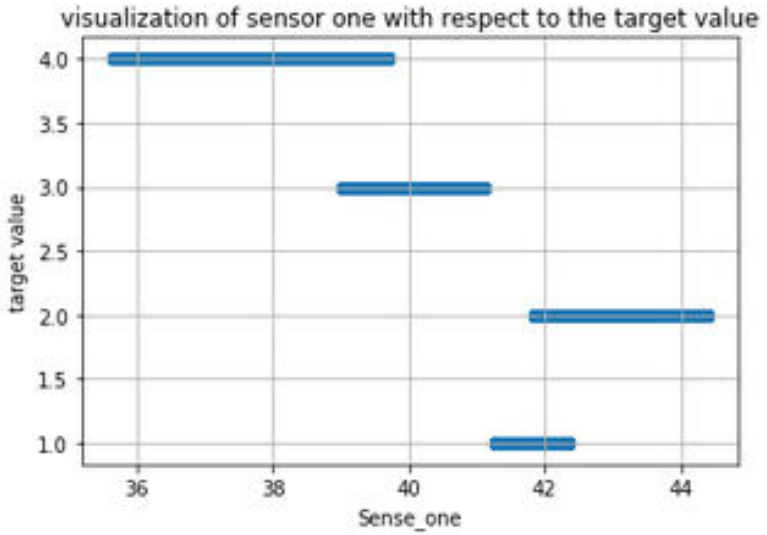
```
In [28]: import pandas as pd
         read_data = pd.read_csv('merged_preprocess_data.csv')

In [29]: read_data = read_data.drop('Unnamed: 0', axis = 1)

In [30]: type(read_data['target'][3])
Out[30]: numpy.float64

In [31]: read_data.head()
```

Out[31]:

	sense_eight	sense_five	sense_four	sense_one	sense_seven	sense_six	sense_three	sense_two	target
0	7.7250	77.7575	6.875	41.935	61.2100	47.1925	23.2900	20.2800	1.0
1	7.7225	77.7075	6.875	41.960	61.2100	47.2700	23.3075	20.3000	1.0
2	7.7300	77.7025	6.875	41.935	61.2100	47.1625	23.2800	20.2875	1.0
3	7.7250	77.7025	6.875	41.915	61.1775	47.2625	23.2900	20.2800	1.0
4	7.7250	77.7575	6.875	41.960	61.1600	47.2450	23.2900	20.3025	1.0

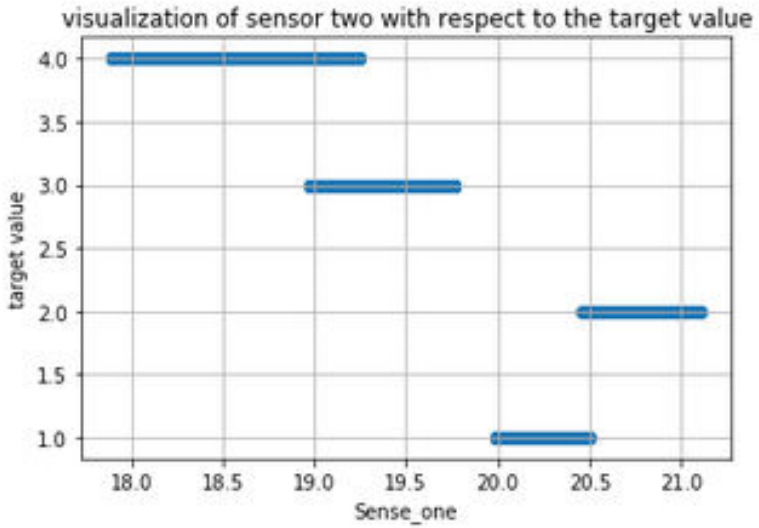
Data visualization and data insight

In [35]:
```
import matplotlib.pyplot as plt
plt.scatter(read_data['sense_one'], read_data['target'])
plt.grid('on')
plt.xlabel('Sense_one')
plt.ylabel('target value')
plt.title('visualization of sensor one with respect to the target value')
plt.show()
plt.savefig('visualization of sensor one with respect to the target value.png')
```



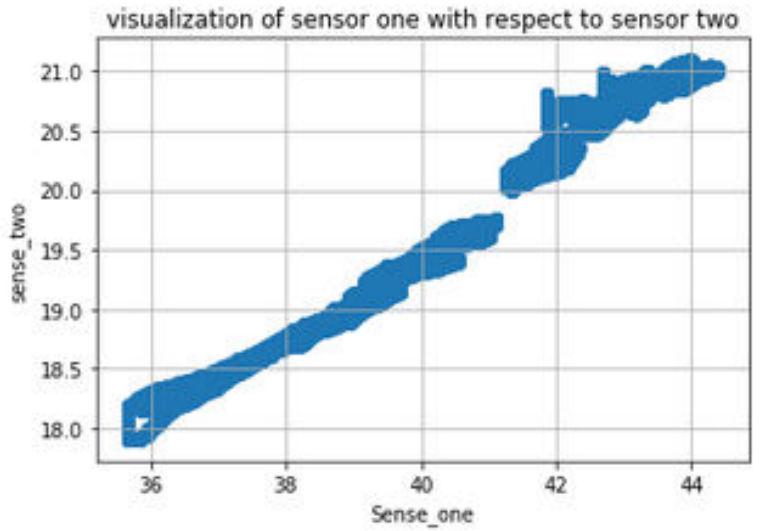
Data visualization and data insight

In [37]:
```
import matplotlib.pyplot as plt
plt.scatter(read_data['sense_two'], read_data['target'])
plt.grid('on')
plt.xlabel('Sense_one')
plt.ylabel('target value')
plt.title('visualization of sensor two with respect to the target value')
plt.show()
plt.savefig('visualization of sensor two with respect to the target value.png')
```



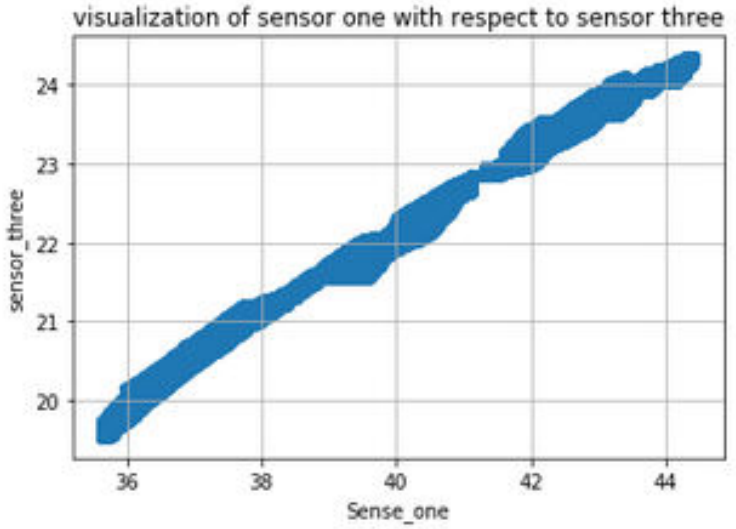
Data visualization and data insight

In [38]:
```
import matplotlib.pyplot as plt
plt.scatter(read_data['sense_one'], read_data['sense_two'])
plt.grid('on')
plt.xlabel('Sense_one')
plt.ylabel('sense_two')
plt.title('visualization of sensor one with respect to sensor two')
plt.show()
plt.savefig('visualization of sensor one with respect to sensor two value.png')
```



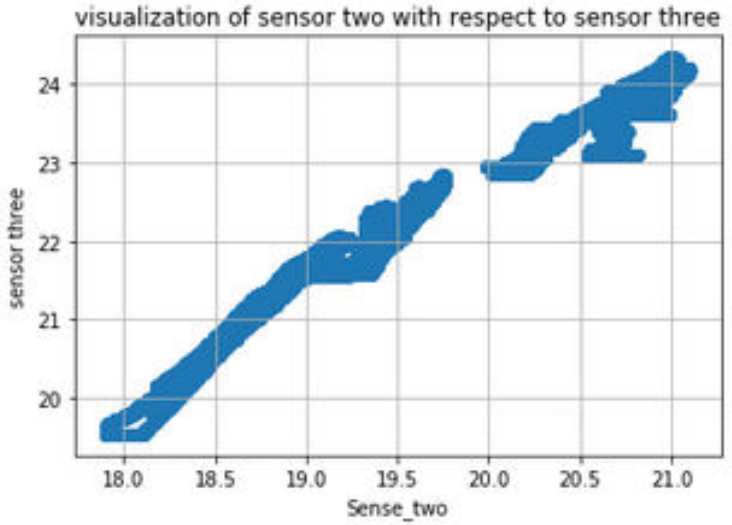
Data visualization and data insight

In [39]:
```
import matplotlib.pyplot as plt
plt.scatter(read_data['sense_one'], read_data['sense_three'])
plt.grid('on')
plt.xlabel('Sense_one')
plt.ylabel('sensor_three')
plt.title('visualization of sensor one with respect to sensor three')
plt.show()
plt.savefig('visualization of sensor one with respect to sensor three value.png')
```



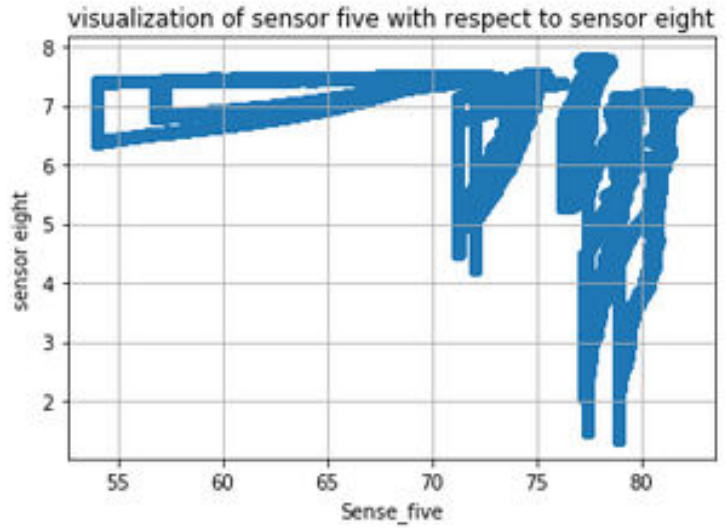
Data visualization and data insight

In [40]:
```
import matplotlib.pyplot as plt
plt.scatter(read_data['sense_two'], read_data['sense_three'])
plt.grid('on')
plt.xlabel('Sense_two')
plt.ylabel('sensor three')
plt.title('visualization of sensor two with respect to sensor three')
plt.show()
plt.savefig('visualization of sensor two with respect to sensor three value.png')
```



Data visualization and data insight

In [41]:
```
import matplotlib.pyplot as plt
plt.scatter(read_data['sense_five'], read_data['sense_eight'])
plt.grid('on')
plt.xlabel('Sense_five')
plt.ylabel('sensor eight')
plt.title('visualization of sensor five with respect to sensor eight')
plt.show()
plt.savefig('visualization of sensor five with respect to sensor eight value.png')
```



Conclusion

- I could vividly note that there was a corresponding correlation amongst the sensors and some are not linearly related.
- We were able to build an efficient model which is uniquely able to predict the type of gas intercepted with our sensors with an accuracy of 100.
- The algorithms used are Support vector machine(kernels = ‘rbf’, ‘linear’), KnearestNeighbors classifier.
- I was efficiently able to build a model that can successfully detect gases and tell us the type of gas being detected.