CHEM LEARNING CHALLENGE

- House Pegasus

Team Members:

1) Sriram Pingali - 8332970018
2) Akshat Porwal - 8318360861
3) Parth Kanani - 9820381305

Introduction:

The problem statement for the Inter Collegiate Technical Championship competition Chem Learning Challenge was to deploy a multi-class classification technique for predicting varying levels of carbon monoxide poisoning.

Description

This is a multiclass classification problem where we are required to classify the CO level in the atmosphere as Very High, High, Moderate, Low and Very Low according to the 13 given parameters or features namely:

The given attributes:

1. True hourly averaged concentration CO in mg/m3= CO(GT);

2. PT08.S1 (tin oxide) hourly averaged sensor response= PT08.S1(CO);

3. True hourly averaged concentration of non methane hydrocarbons in mg/m3 = NMHHC_GT 4. True hourly averaged Benzene concentration in microg/m3 =C6H6(GT);

5. PT08.S2 (titania) hourly averaged sensor response(NMHC=Non methane hydrocarbons) = PT08.S2(NMHC);

6. True hourly averaged NOx concentration in ppb = NOx(GT);

7. PT08.S3 (tungsten oxide) hourly averaged sensor response = PT08.S3(NOx);

8. True hourly averaged NO2 concentration in microg/m3 = NO2(GT);

9. PT08.S4 (tungsten oxide) hourly averaged sensor response = PT08.S4(NO2);

10. PT08.S5 (indium oxide) hourly averaged sensor response = PT08.S5(O3);

11. Temperature in °C = T;

12. Relative Humidity (%) = RH;

13. Absolute Humidity = AH.

Presentation of a sensor fusion algorithm (more specifically, a neural calibration method) for extrapolation of Carbon Monoxide concentration values as given by the dataset (generated from solid-state, multisensory device) for urban pollution monitoring.

As supporting research in the field suggests, the deployment of ad-hoc sensor fusion algorithm to obtain multivariate calibration have shown interesting capabilities for concentration estimation problems in seemingly intractable atmospheric pollutants.

Often, the selected calibration models have been artificially generated deep feedforward neural networks, particularly due to their capability to exploit partial selectivity of sensors as an advantage that allows them to model these complex frameworks to a high degree of accuracy, with a reasonable expectation to be limited by inherent stochastic noise in the data. A further development of the systems neatly estimates the analytes for which no particular sensor was developed before.

Classification Approach:

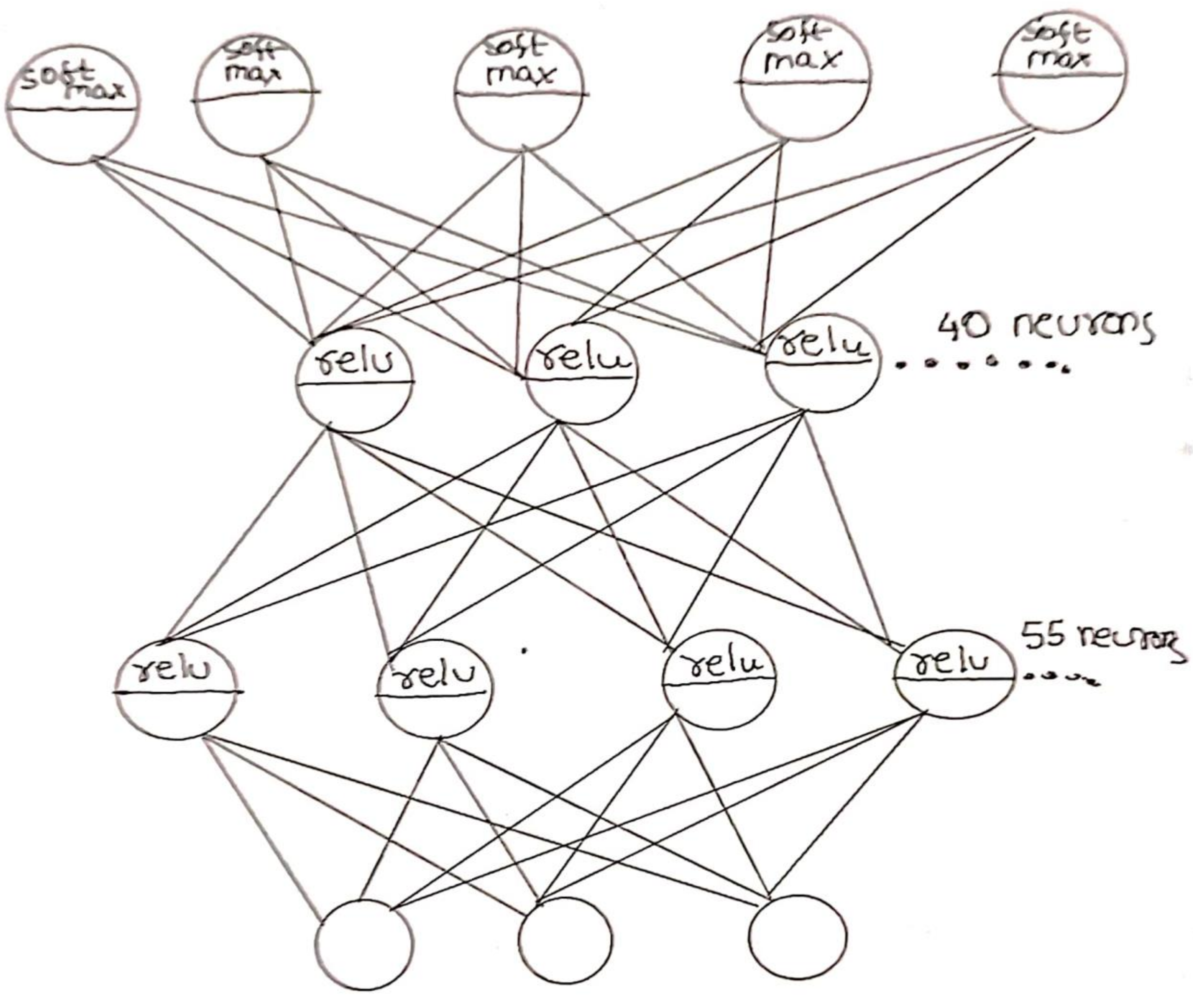
Our approach to predict the pollution levels in the air is through a feed forward neural network. We preferred deployment of a multi-layered feedforward neural network over straight forward implementation of a Machine Learning algorithm (like a random forest, SVM or decision tree) due to increased performance bartered by the multiple layers over greater model complexity. The accuracy over validation data from ML algorithms seemed to saturate at ~70 - 75%. Whereas Neural Network enable us to alter this scenario.

Proposed Approach:

A 4-layered network was observed to fare better on cross-validation set than a single layered machine learning algorithm. Careful design of parameters and hyper parameter tuning of each layer heightens the accuracy obtained. We experimented with the number of neurons of each layer and activation functions and found the below depicted model to perform the best for the given data.

4-Layer FF Network

Layer sizes: 3 (Input), 55, 40, 5 (Output)

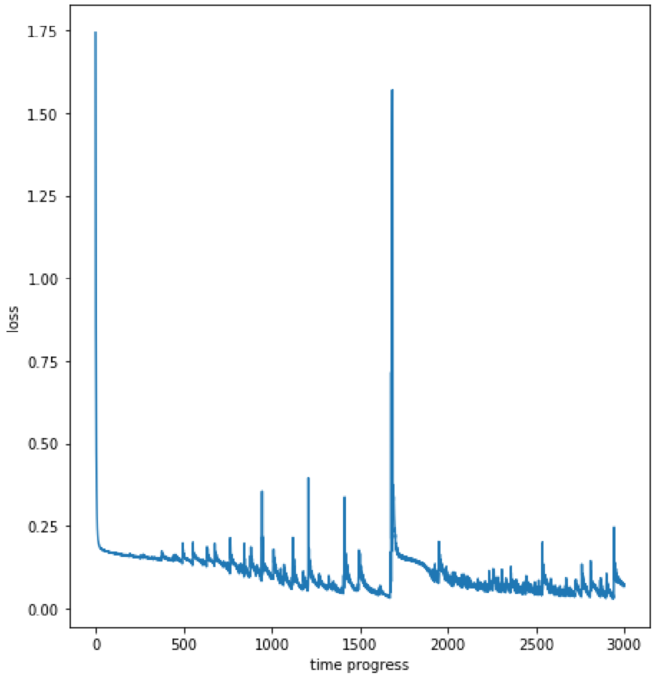


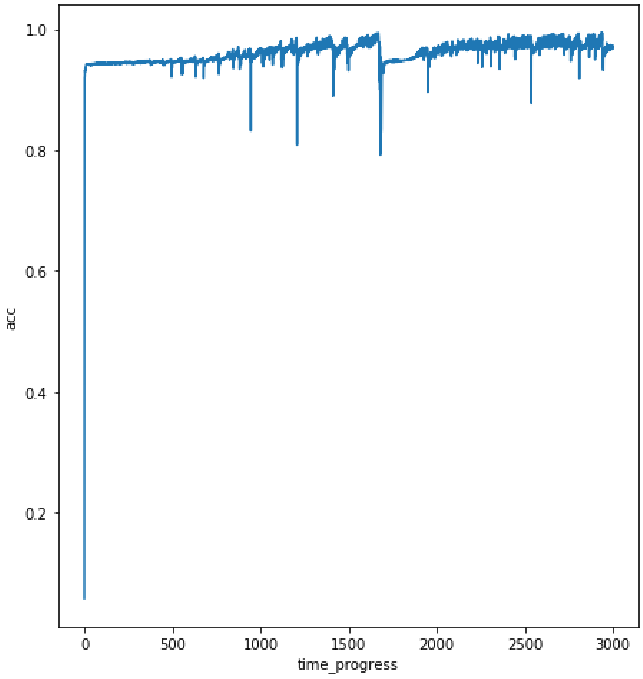
The model reportedly saturated in accuracy after 4 layers of parameters. Batch normalization of data is done as the parameters show high variance where some vary in range of thousands and some in a few decimals. To gain the best out of Batch Normalization, Mini batch gradient is opted, with a batch size of 2048 which seems a fairly good choice. ReLU activation is used as it is a widely popular choice for Feed forward neural networks, and it is quite promising in our case as well.

The labels are clearly categorical type. Therefore, we chose to use One Hot Encoding over the label data to establish the “true” distribution which the network has to learn. Under the frequentist perspective, the "true" distribution is assumed to lie in the hypothesis class, and the Adam optimization rule on a Cross Entropy Loss function decides on a functional mapping for a near-correct estimation of the true underlying distribution. We chose Adam optimizer over Stochastic Gradient Descent as the latter seems to saturate at low accuracy, probably due to plateau regions in the loss function. Learning rate is set to 0.02 which avoids jumping and also to avoid slow convergence. 1500 Epochs seemed to be good for the mini batch descent after which the loss is saturating.

Loss

Accuracy

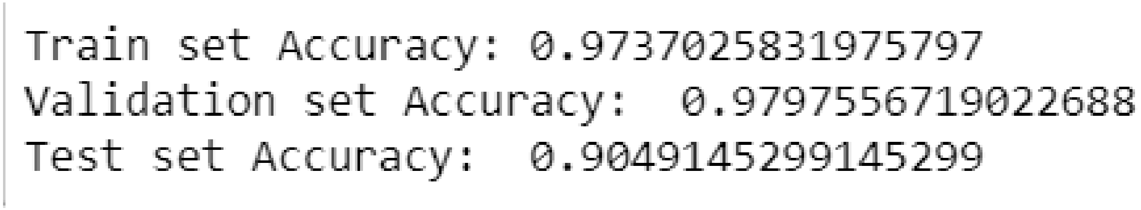




Feature correlation between the different columns of the data has been excluded on deliberation since it is not Big Data handling.

The learning objective is to find those probabilities which match the distribution to a maximum possible accuracy. The model generalizes fairly well with~90% accuracy on test data and ~98% accuracy on training and cross validation data, which confirms prior belief that a neural network can learn behavior of the novel pollution sensing device.

Resulting accuracies from model



```
Train set Accuracy: 0.9737025831975797
Validation set Accuracy:  0.9797556719022688
Test set Accuracy:  0.9049145299145299
```

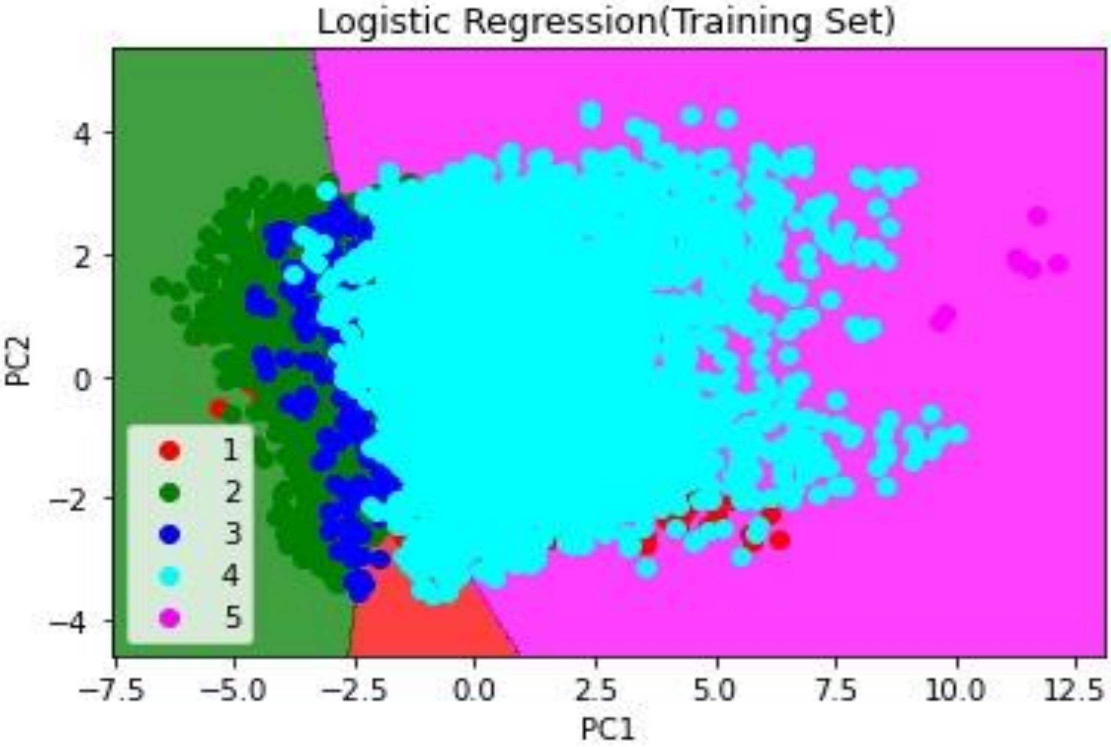
Motivation:

As per the exploratory data analysis done by us:

For dimensionality reduction and finding the relation among the independent variables we performed Principal Component Analysis using Logistic Regression. From the m independent variables of your dataset PCA extracts p<=m new independent variables that explain the most the variance of the dataset regardless of dependent variable.

So, we chose to take p = 2 while m = 13(given) and observed the graph as follows:

1-Very Low; 2-Low; 3-Moderate; 4-High; 5-Very High



The example variance matrix of the 13 new variables for the above PCA looks like below:

1.	0.511834
2.	0.187199

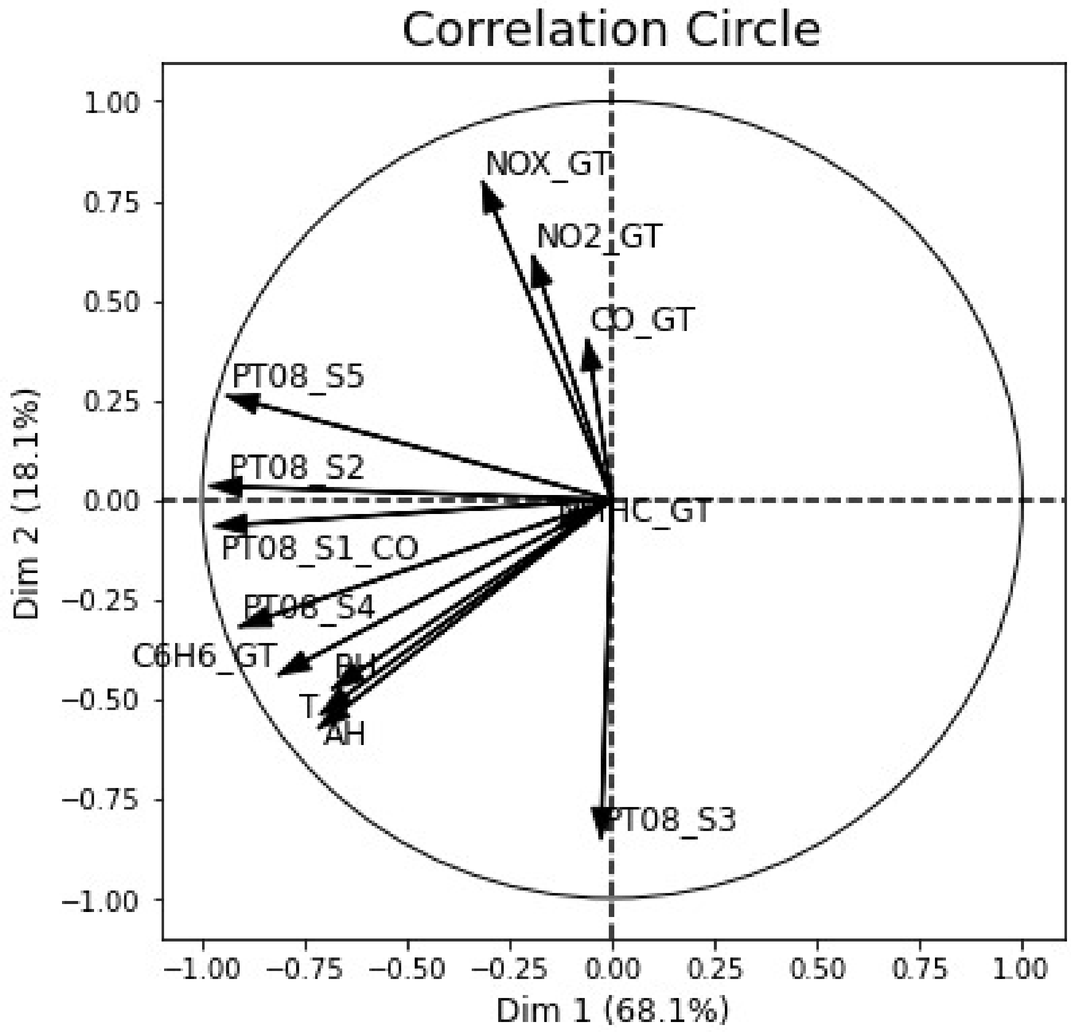
3.	0.104888
4.	0.0706184
5.	0.0442369
6.	0.0328409
7.	0.0155125
8.	0.0104086
9.	0.00910732
10.	0.00652626
11.	0.00339577
12.	0.00268964
13.	0.000742604

We take the two highest variance variables to plot our graph because that contribute the most to it.

Using PCA we also tried to make a correlation circle in which the correlations between the original dataset features and the principal component(s) are shown via coordinates.

- Features with a positive correlation will be grouped together.
- Totally uncorrelated features are orthogonal to each other.
- Features with a negative correlation will be plotted on the opposing quadrants of this plot.
- The components which are closer to axis have a significant contribution towards the PCA variables.

- The leftward pointing features favour the x-component while the downward pointing features favour the y-component.



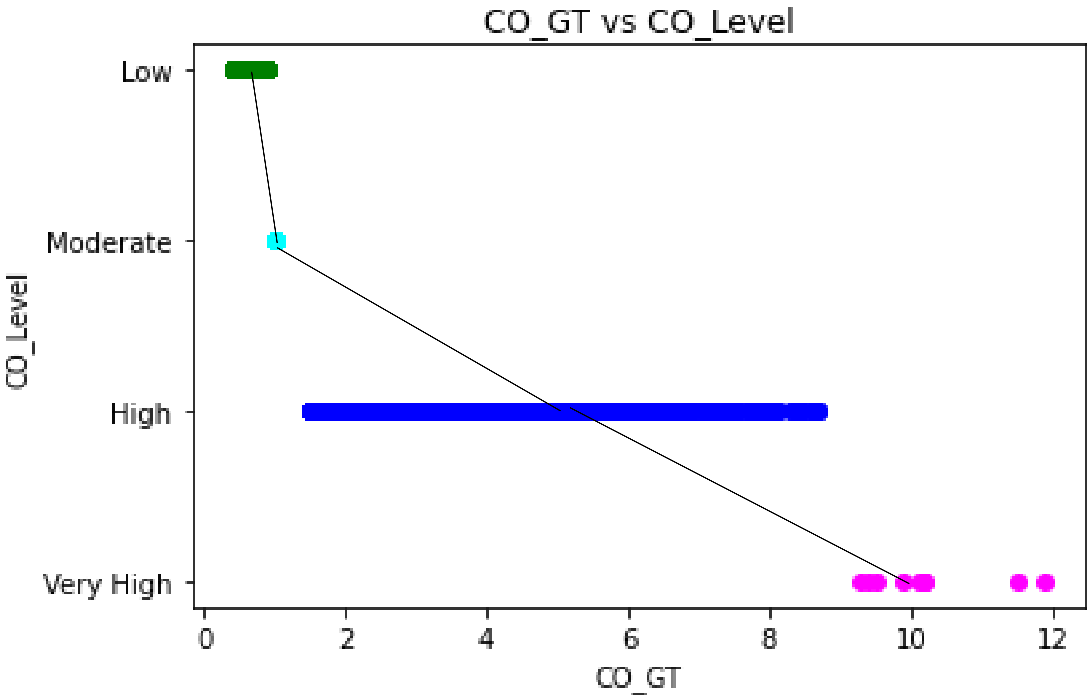
Note that the percentage values shown on the x and y-axis denote how much of the variance in the original dataset is explained by each principal component axis, i.e. If PC1 lists 68.1% and PC2 lists 18.1% as shown above then combined, the 2 principal components explain 86.2% of the total variance.

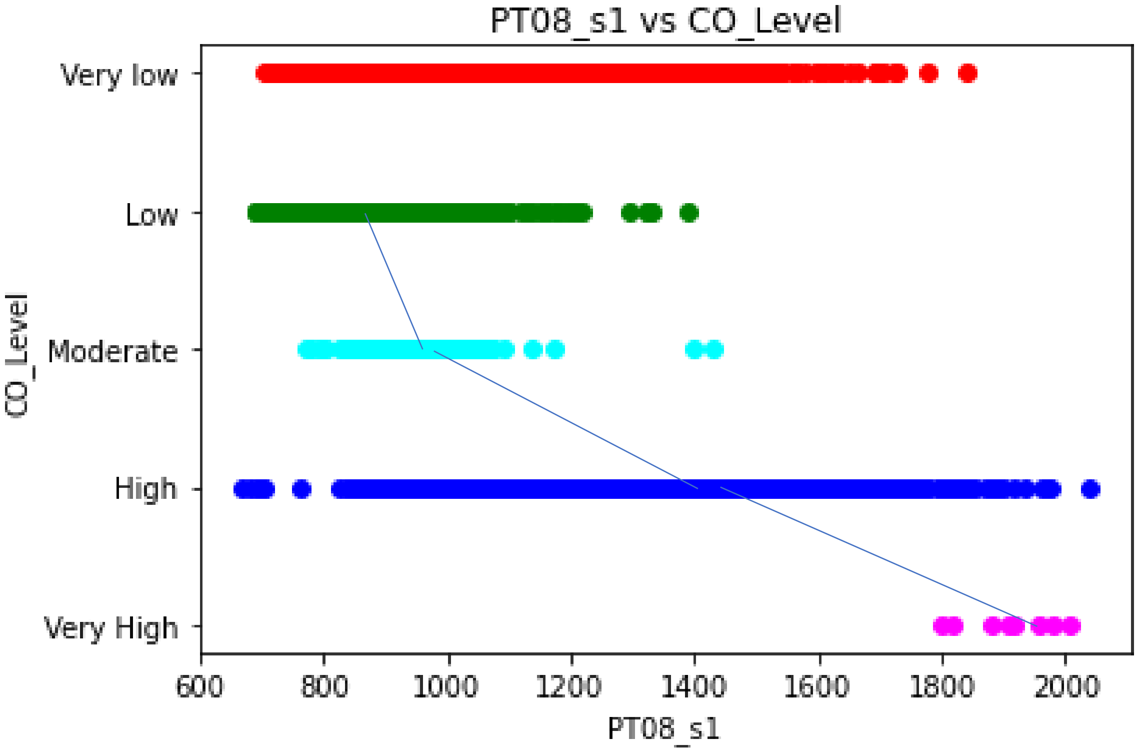
So as shown in the above correlation circle PT08_S1(CO), PT08_S2(NMHC) and CO_GT features are the most important features for this classification.

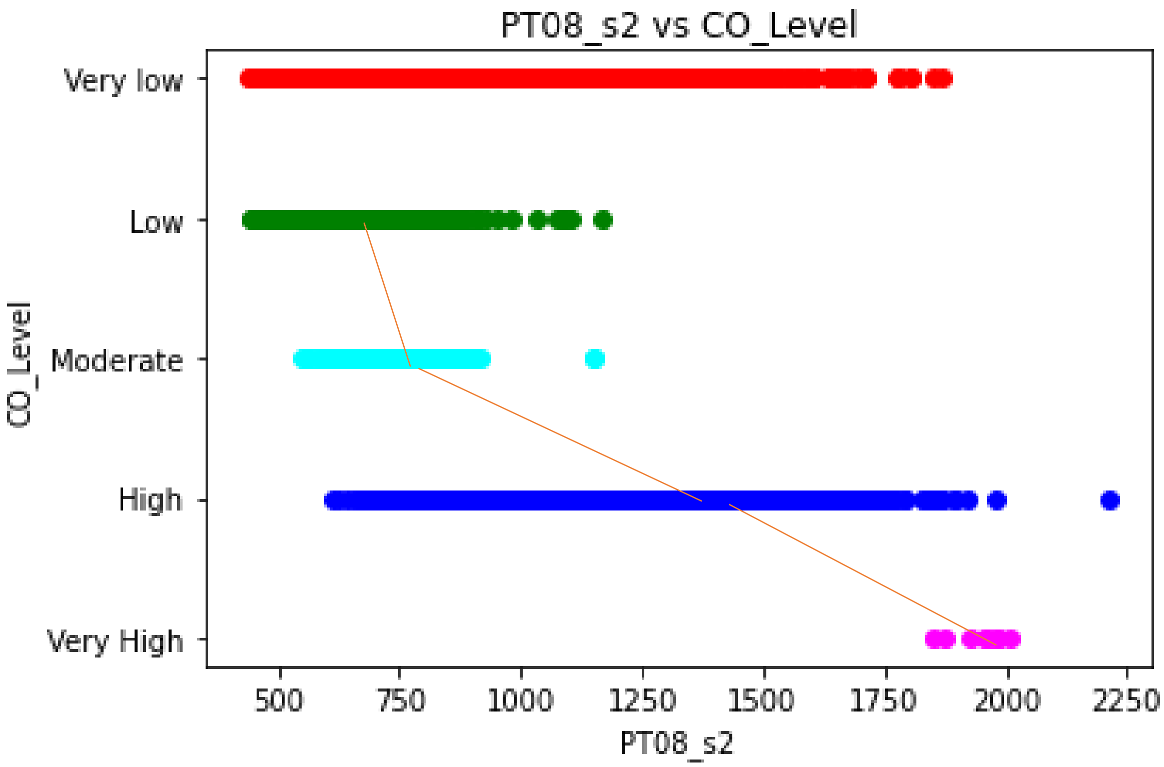
To verify the fact, we perform data analysis over those attributes. We generated plots to depict the relation between the three attributes with target labels. The plots of PT08_S1(CO) against CO level and PT08_S2(NMHC) against CO level are shown below which give an intuitive feel of the linear relation between attributes and labels(with the exception of Very Low values which is mostly due to the missing or insignificant amounts of CO_GT values). Furthermore, a direct linear relation is observed between class label for CO concentration averaged per hour and the corresponding class label which confirms our reasoning.

Evident ranges in train data through visual verification

CO_GT	CO_level
-200	Very Low
<1	Low
=1	Moderate
1-9	High
>9	Very High


CO_GT vs CO_Level
Low
Moderate
High
Very High
CO_Level
0
2
4
6
8
10
12
CO_GT


PT08_s1 vs CO_Level
Very low
Low
Moderate
High
Very High
CO_Level
600
800
1000
1200
1400
1600
1800
2000
PT08_s1



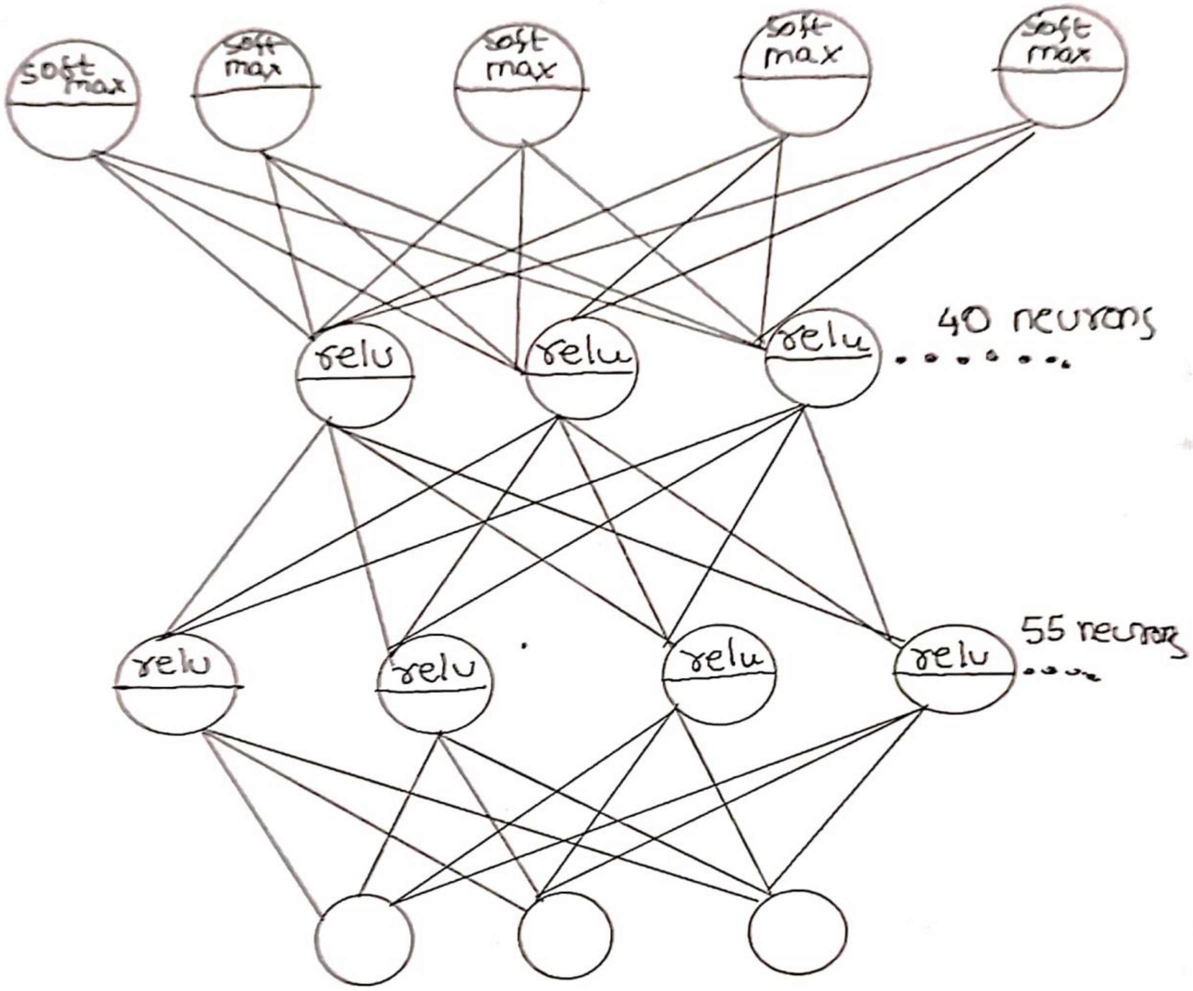
Implementation:

Our framework of choice was Pytorch, due to its simplicity in training and simple in-built methods and functions.

First, we read the CLC_train.csv into a pandas data frame using the read_csv function. From here, the CO_level column is converted to a numpy array named data. And the columns “CO_GT”, “PT08_S1_CO”, ”PT08_S2_NMHC” are fed into a different numpy array named labels.

We use One Hot Encoder from sklearn.preprocessing to encode the categorical label data. We then split the train, validation data in 0.6 : 0.4 ratio.

Neural Network Architecture



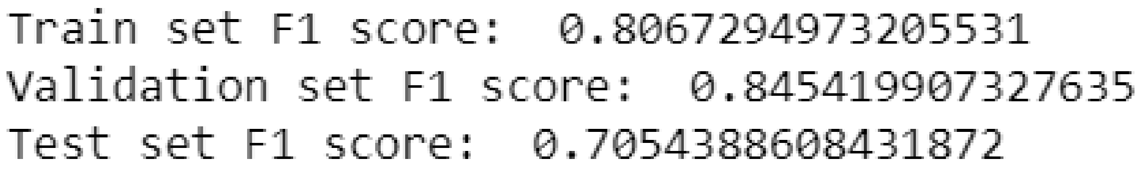
From here, the training process occurs with the following parameters to the above-mentioned neural network.

Hyper Parameter	Value
Learning Rate	0.02
Epochs	1500
Batch Size	2048
Train test split	0.6: 0.4
Layer sizes	3, 55, 40, 5

We use Adam optimizer with default parameters, and Cross Entropy Loss function for the training process.

Results:

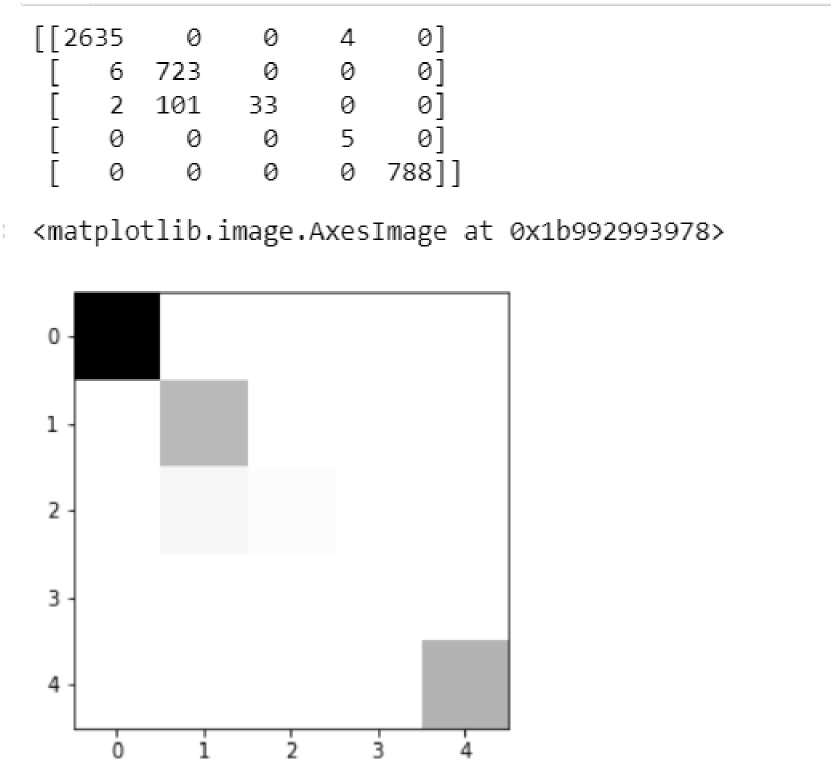
F1 Score



```
Train set F1 score:  0.8067294973205531
Validation set F1 score:  0.845419907327635
Test set F1 score:  0.7054388608431872
```

Confusion Matrix

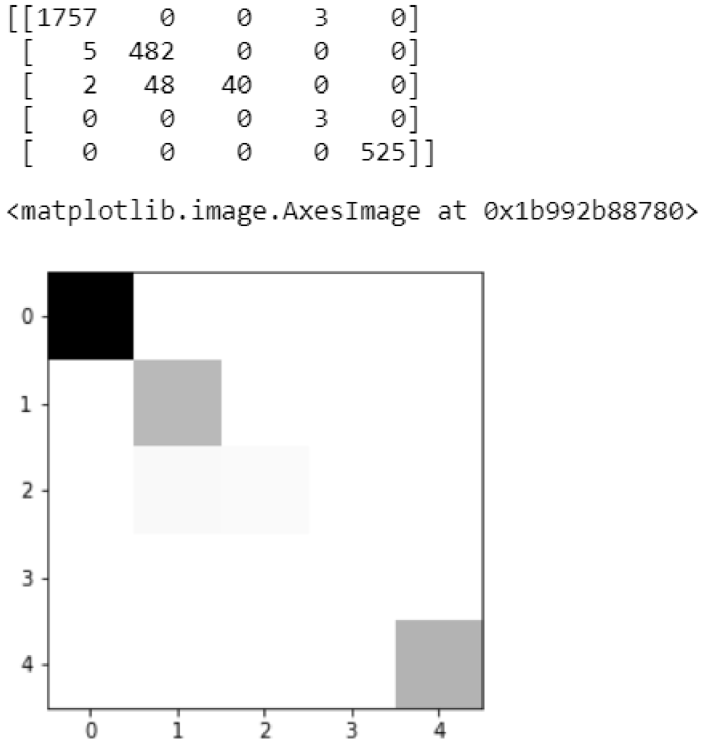
Training set



```
[[2635    0    0    4    0]
 [   6  723    0    0    0]
 [   2  101   33    0    0]
 [   0    0    0    5    0]
 [   0    0    0    0  788]]
```

```
<matplotlib.image.AxesImage at 0x1b992993978>
```

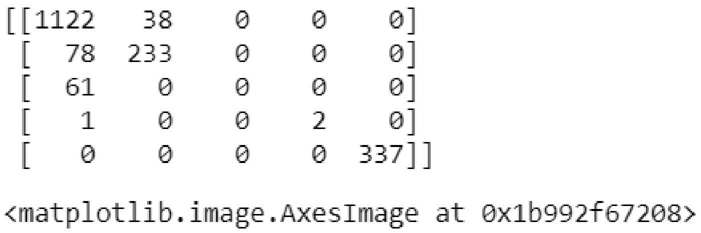
Validation Set



```
[[1757    0    0    3    0]
 [   5  482    0    0    0]
 [   2   48   40    0    0]
 [   0    0    0    3    0]
 [   0    0    0    0  525]]
```

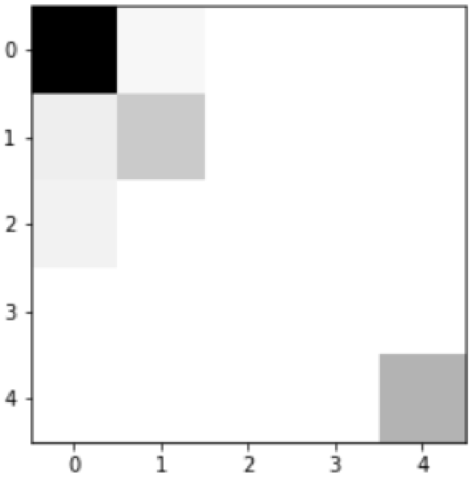
```
<matplotlib.image.AxesImage at 0x1b992b88780>
```

Test Set

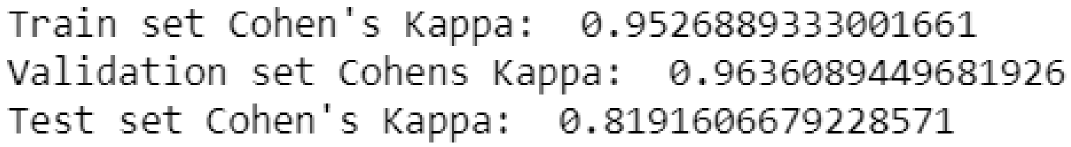


```
[[1122   38    0    0    0]
 [  78  233    0    0    0]
 [  61    0    0    0    0]
 [   1    0    0    2    0]
 [   0    0    0    0  337]]

<matplotlib.image.AxesImage at 0x1b992f67208>
```

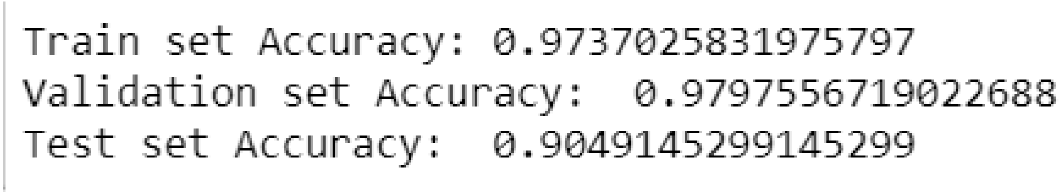


Cohen's Kappa Coefficient



```
Train set Cohen's Kappa:  0.9526889333001661
Validation set Cohens Kappa:  0.9636089449681926
Test set Cohen's Kappa:  0.8191606679228571
```

Overall Accuracy



```
Train set Accuracy: 0.9737025831975797
Validation set Accuracy:  0.9797556719022688
Test set Accuracy:  0.9049145299145299
```

Conclusion:

Hence, we conclude our effort, with this novel method to predict pollution levels. We are grateful for this opportunity to explore the chemical domain involved in the pollution levels in the modern world. We realize the importance of collaborative work. Most importantly we realized that even in this strange period of global pandemics and lockdowns, we are capable of working together and collaboratively learn new things. After all, that’s how we survive!