Smoke Detection

Avinash Chaluvadi - CS5300

May 1, 2023

Contents

1 Abstract

What is a smoke detector? A smoke detector is a device designed to detect smoke, which is commonly an indication of a fire. These detectors usually have sensors that are triggered when smoke is detected, and an alarm system that alerts people to evacuate. There are different types of smoke detectors, including ionization and photoelectric ones. Recently, advances in technology have led to the development of smoke detectors that use IOT devices and AI models to increase their efficiency and accuracy in detecting fires.

How smoke detectors are essential to AI ? Smoke detectors are significant components in AI-based fire detection systems. These systems use advanced algorithms to analyze data gathered from smoke detectors and other sensors to identify smoke patterns and locate the source of the fire. The smoke detectors are crucial in providing important information to these systems, helping them make accurate predictions and take immediate measures to minimize the risk of fires. As a result, smoke detectors play an essential role in enhancing the accuracy and reliability of AI-based fire detection systems.

Motivation The motivation behind this project is to use artificial intelligence to detect smoke, which can be a time-consuming task. The goal is to develop supervised learning algorithms that use binary classification to accurately determine whether smoke is present or not. Binary classification involves categorizing elements as true or false, or 1 or 0. By developing these algorithms, the process of detecting smoke can be made more efficient and reliable.
In conclusion, this paper presents a smoke detection dataset and demonstrates the effectiveness of neural network models, especially ANNs, in detecting smoke. The dataset and the models can be used as a benchmark for developing more advanced smoke detection systems, which can potentially save lives and properties.

2 Dataset

The **"Smoke Detection Dataset"** was obtained from the Kaggle Data Science. The smoke data were obtained with the help of IOT devices. Many different environments and fire sources have to be sampled to ensure a good dataset for training. A short list of different scenarios which are captured:

- Normal indoor
- Normal outdoor
- Indoor wood fire, firefighter training area
- Indoor gas fire, firefighter training area
- Outdoor wood, coal, and gas grill
- Outdoor high humidity

The dataset for smoke detection comprises 44757 positive samples indicating fire alarm and a corresponding number of negative samples, totaling 17871 samples in the dataset. These samples have been manually verified by annotators, and each row contains 12 continuous variables as well as a class label at the end, with two possible values - 0 (negative) and 1 (positive). The input features pertain to certain fields.

- Temperature[C], Humidity[%], TVOC[ppb], eCO2[ppm], Raw H2, Raw Ethanol, Pressure[hPa], PM1.0, PM2.5, NC0.5, NC2.5, Fire Alarm.

2.1 Visualization of the distribution of each input features

The histogram plot before normalization of each info highlights indicating their most extreme and least value as well as how they are distributed can be found in the pictures given underneath.

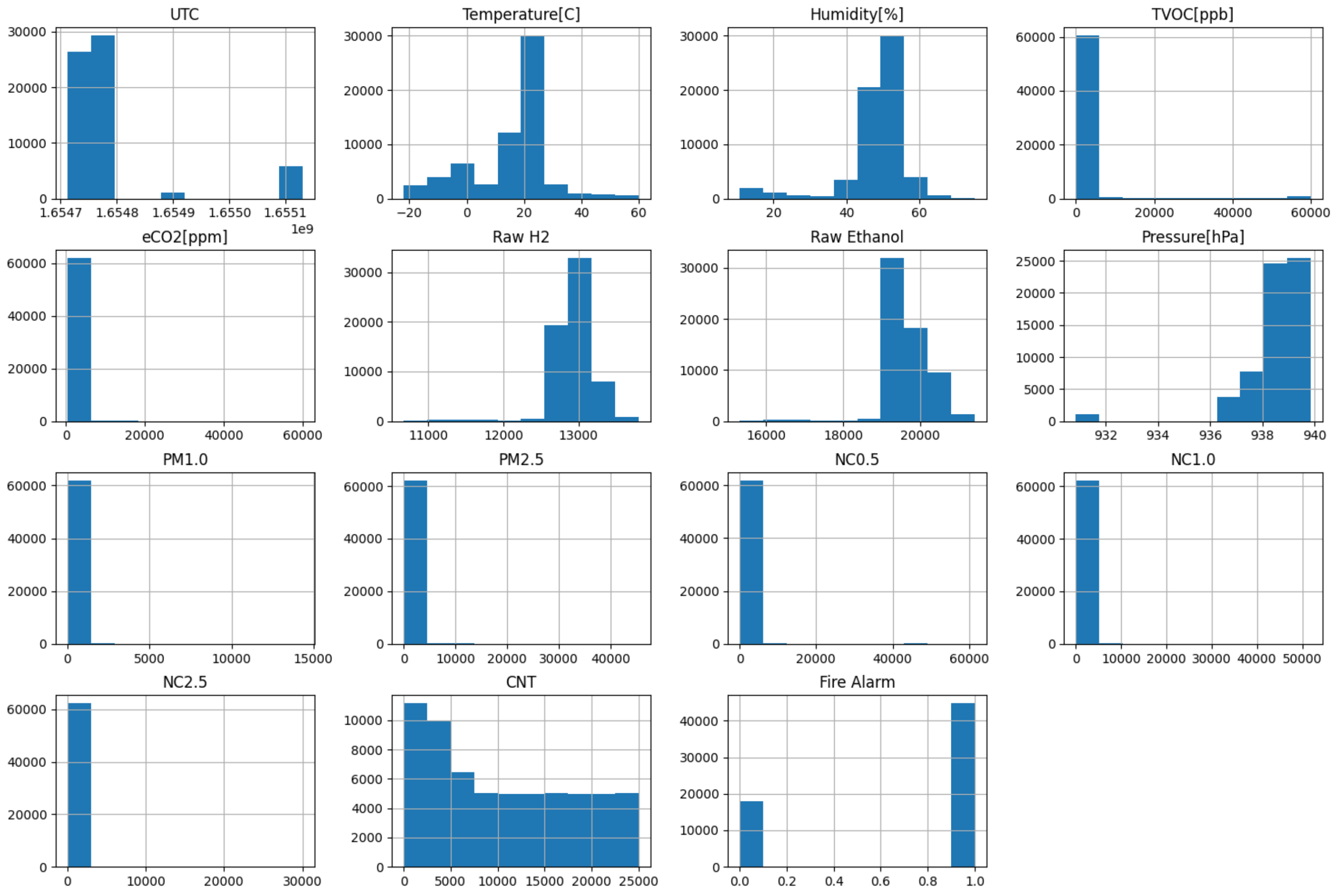


Figure 1: Input Data Distribution Histograms - Before Normalization

2.2 Distribution of the output labels

Notice that the data is imbalanced and may need to be resampled (such as oversampling, undersampling, or generate syn- thetic samples).

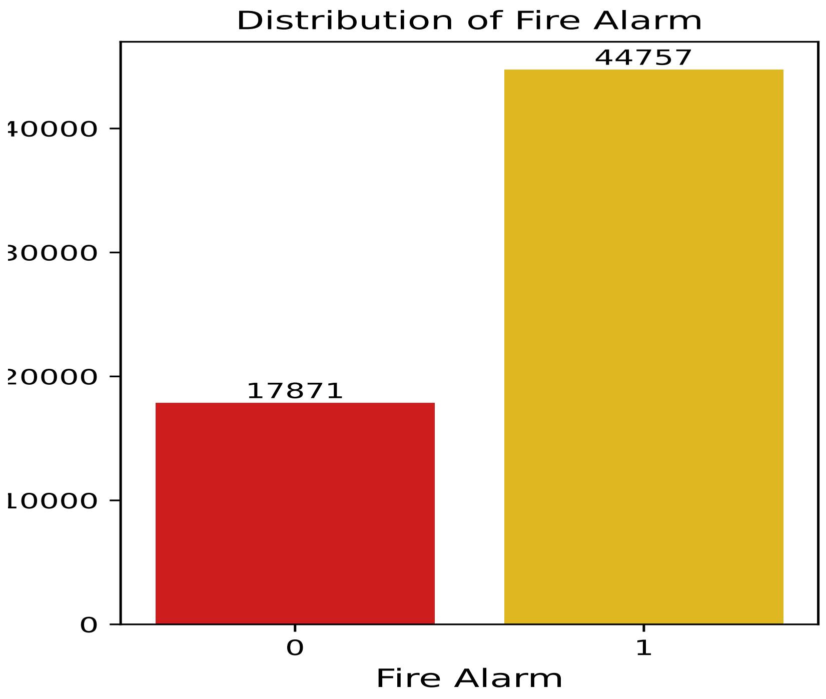


Figure 2: Output Data Distribution - Before Normalization

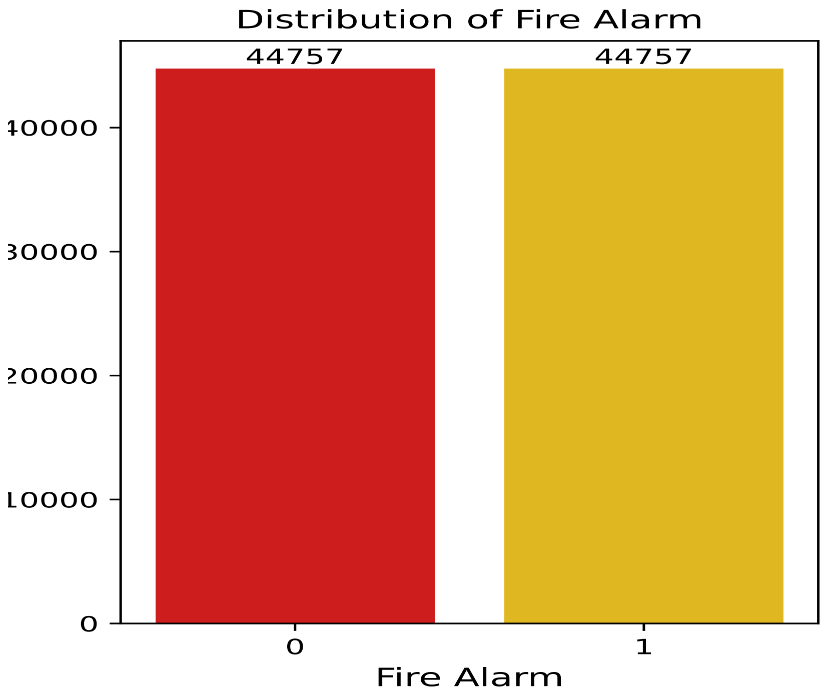


Figure 3: Output Data Distribution - After Normalization

3 Data Processing

3.1 Data Splitting

The data was randomly shuffled and then the dataset was split into training and validation, where 80% of the dataset was allocated for training and 20% was allocated for validation.

3.2 Data Normalization

Data preprocessing is essential before data mining to address the non-uniform distribution of data. To achieve this, normalization techniques are used to make the optimization problem more numerically stable and improve training. Normalization helps to ensure all values lie between 0 and 1 and outliers are visible within the normalized data. There are two normalization techniques available, each with its own consequences, but either technique can be used for now.

Mean Normalization Formula

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Z-Score Normalization

$$X_{normalized} = \frac{X - X_{mean}}{X_{standard_deviation}}$$

Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]
6.262800e+04	6.262800e+04	62628.000000	6.262800e+04	6.262800e+04	6.262800e+04	6.262800e+04
-2.904435e-17	1.016552e-16	0.000000	5.445815e-17	-6.244535e-16	7.696752e-16	-4.559963e-14
1.000000e+00	1.000000e+00	1.000000	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
-2.644913e+00	-4.263684e+00	-0.248616	-1.416796e-01	-8.347612e+00	-7.280139e+00	-5.840443e+00
-3.465310e-01	-1.138819e-01	-0.231975	-1.416796e-01	-4.127355e-01	-5.237536e-01	5.431846e-02
2.896953e-01	1.816463e-01	-0.123036	-1.416796e-01	-6.774094e-02	-4.154676e-01	1.414480e-01
6.572003e-01	5.301890e-01	-0.096409	-1.217416e-01	6.112378e-01	5.312139e-01	5.936200e-01
3.061347e+00	3.007211e+00	7.432164	3.112938e+01	3.158326e+00	2.716621e+00	9.263646e-01

Input Feature Statistics before Normalization

Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]
0.280619	0.994939	-0.248612	-0.141677	-2.335917	-2.024990	0.831754
0.281664	0.917108	-0.248612	-0.141677	-2.192779	-1.810064	0.838514
0.282639	0.837021	-0.248612	-0.141677	-2.086343	-1.624670	0.834008
0.283684	0.760318	-0.248612	-0.141677	-2.027619	-1.485215	0.832505
0.284728	0.693767	-0.248612	-0.141677	-1.979907	-1.367088	0.838514

Input Feature Statistics after Normalization

3.3 Normalized Data

The histogram plot after normalization of each info highlights indicating their most extreme and least value as well as how they are distributed can be found in the pictures given underneath.

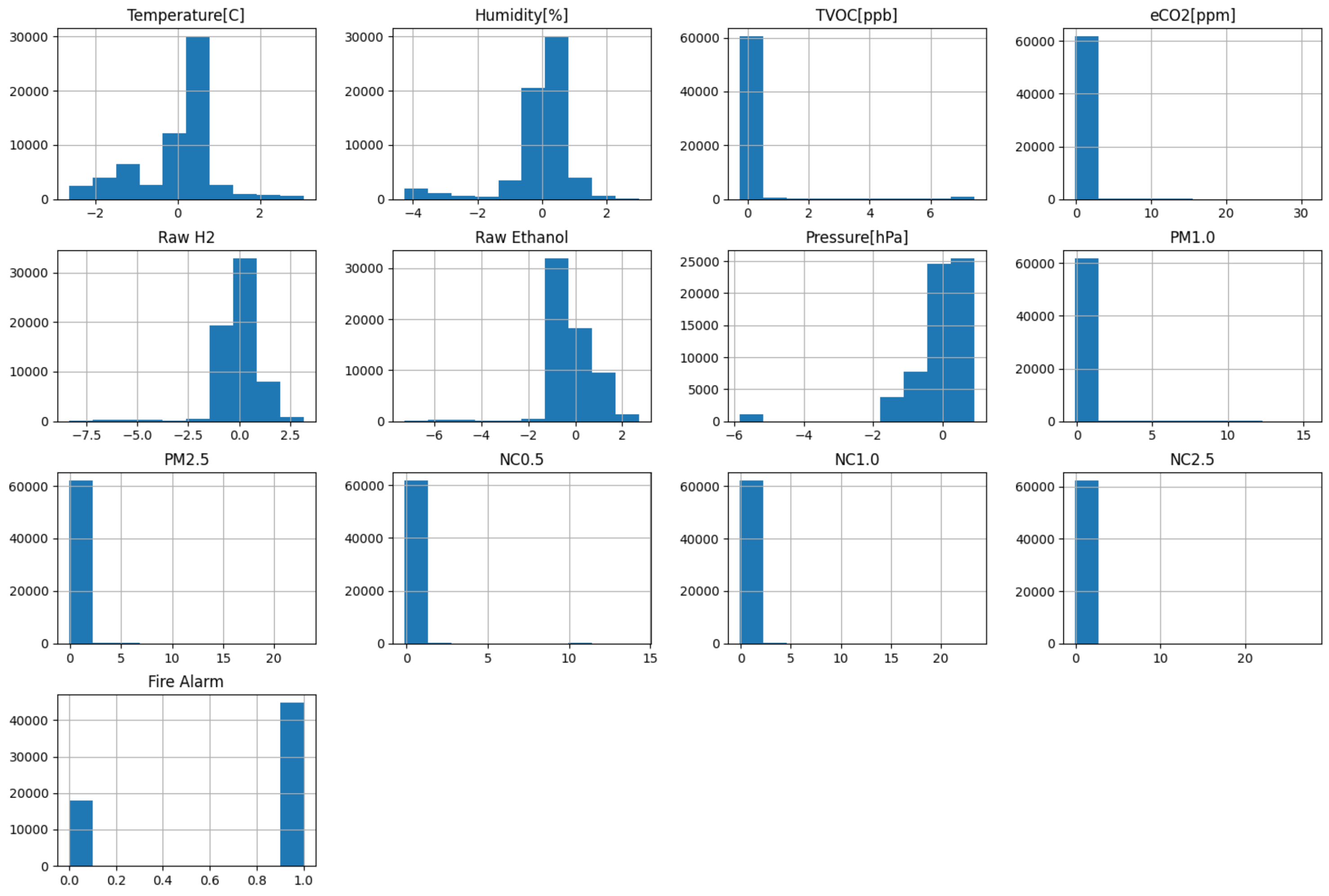


Figure 4: Input Data Distribution Histograms - Before Normalization

4 Modelling

A feed forward artificial neural network architectures was used to create the model.

NOTE: Data is shuffle. Thus, the result will vary every time. All models were compiled and fit on May 1, 2022.

4.1 Selected Neural Network Architecture

The first model is a baseline model (act as a control) that has a basic architecture of one input and one output layer. It will then gradually increase by one hidden layer and up to 5 hidden layers at the end. An early stopping technique was used during the training.

4.1.1 Model Performance

Hidden	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0 Layer	5.4794	0.8349	1.9126	0.8410
1 Layer	0.1685	0.9365	0.1262	0.9748
2 Layer	0.0311	0.9884	0.0297	0.9898
3 Layer	0.0456	0.9818	0.0387	0.9847
4 Layer	0.0061	0.9982	0.0221	0.9927

Table 1: Performance comparison for different hidden layers

As can be seen from the above table, all the basic architecture(1, 2, 3) hidden layers somewhat perform relatively the same, and 4 hidden layer architecture just happen to perform better during this iteration. In the end, 4 hidden layer will be applied to other architecture.

4.2 Learning Curve of Selected Neural Network Architecture

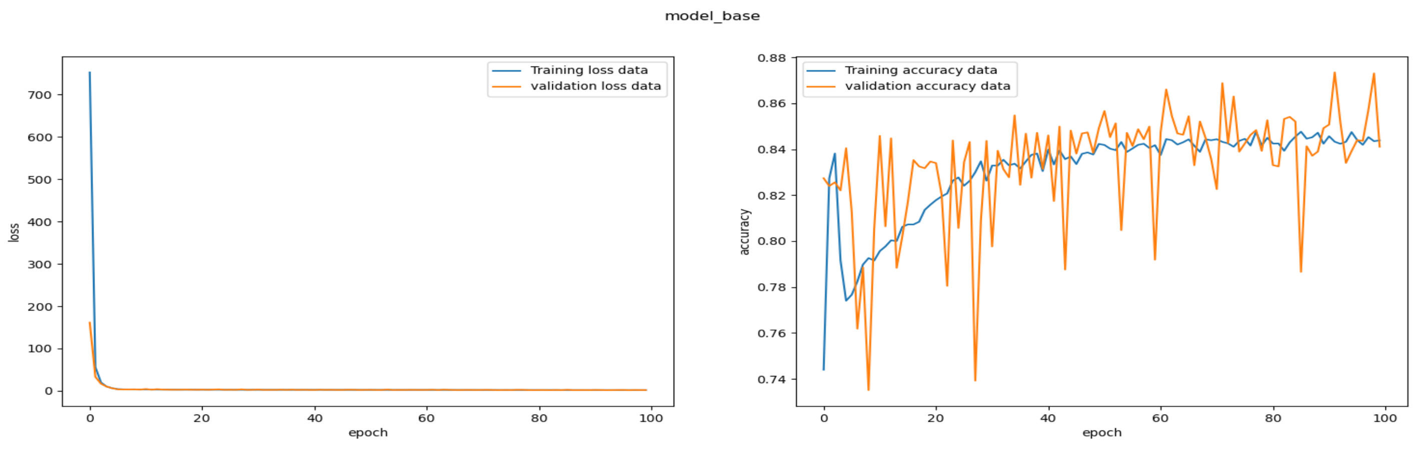


Figure 5: curve showing change in loss/accuracy vs epoch

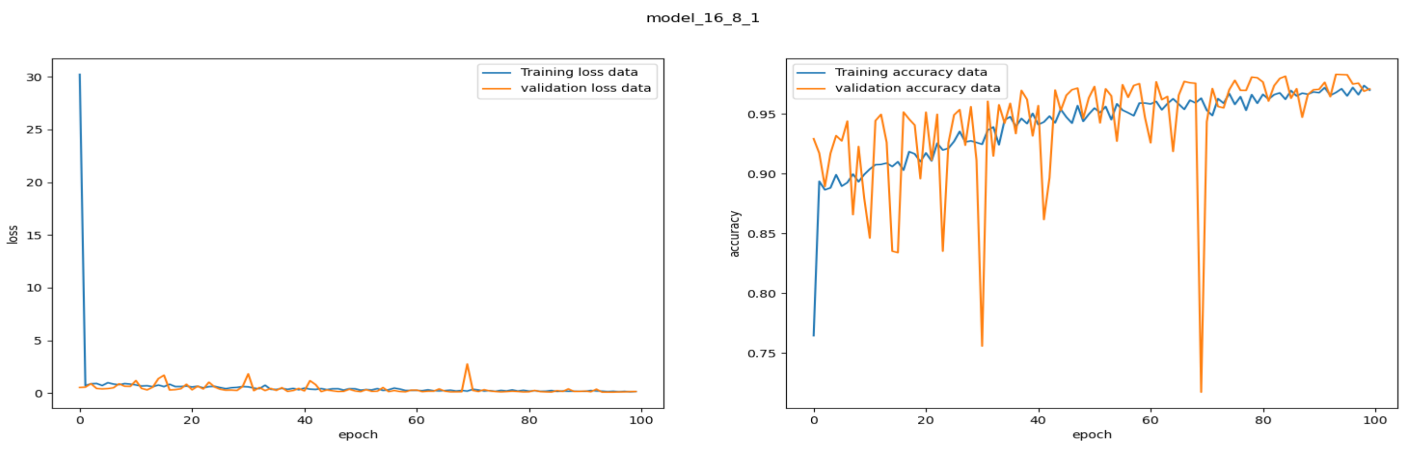


Figure 6: curve showing change in loss/accuracy vs epoch

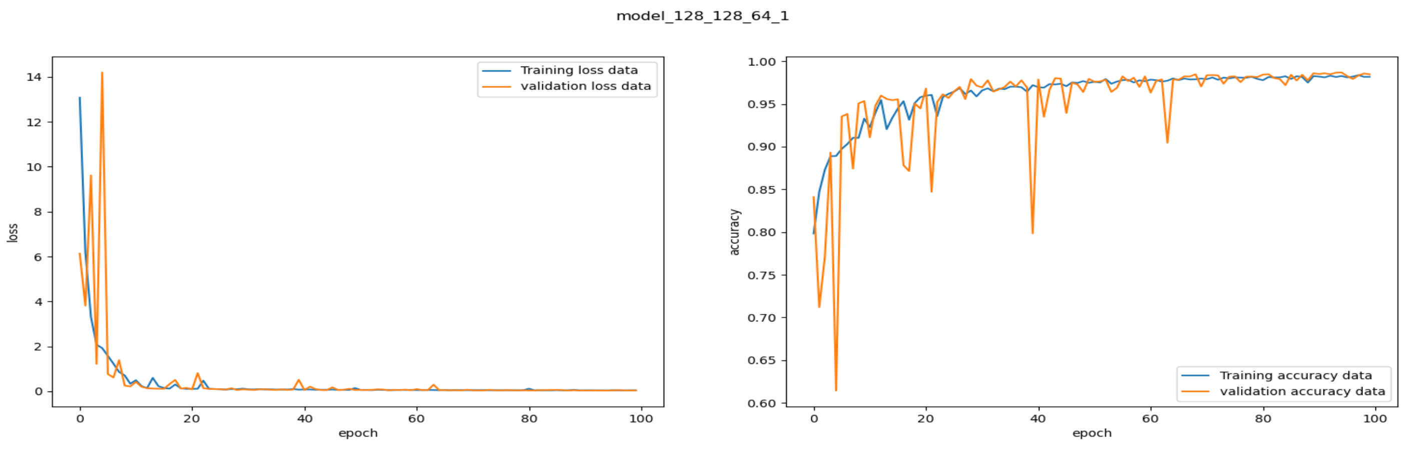


Figure 7: curve showing change in loss/accuracy vs epoch

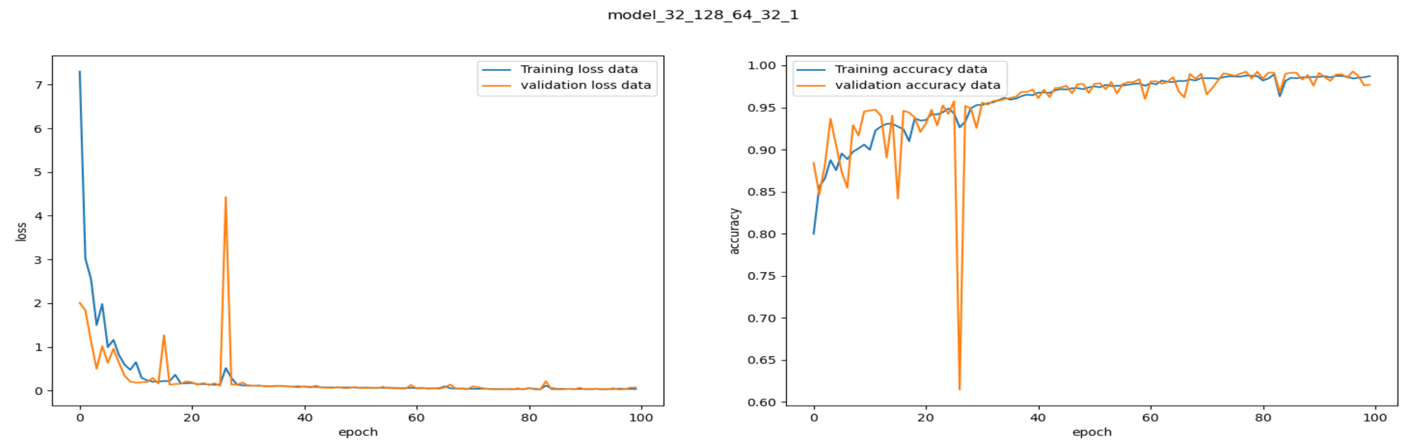


Figure 8: curve showing change in loss/accuracy vs epoch

5 Model Evaluation

Three essential classification model metrics to evaluate. The table given below shows the precision, recall and f1 score for the neural network model.

1. Precision: what proportion of positive identifications was actually correct?
2. Recall: what proportion of actual positives was identified correctly?
3. F1-Score: evaluation metric for classification algorithms, where the best value is at 1 and the worst is at 0.

Model	Precision	Recall	F1-Score
Baseline Model	81.20	89.42	0.85
1 Layer	97.26	97.34	0.97
2 Layer	99.67	98.71	0.99
3 Layer	98.98	98.38	0.98
4 Layer	98.63	98.14	0.98
Overfitting	99.45	99.10	0.99

A useful tool when predicting the probability of a binary outcome is the Receiver Operating Characteristic curve or ROC curve. The area covered by the curve is the area between the red line and the axis. This area covered is AUC. The bigger the area covered, the better the machine learning models are at distinguishing the given classes. In other words, the AUC can be used as a summary of the model skill. The ideal value for AUC is 1.

5.1 Test Accuracy

The closer the graph is to the top and left-hand borders, the more accurate the test. Likewise, the closer the graph to the diagonal, the less accurate the test. In a perfect test, it would go straight from zero up the top-left corner and then straight across the horizontal. The figure is given below shows the receiver operating characteristic curve for the different neural network architectures.

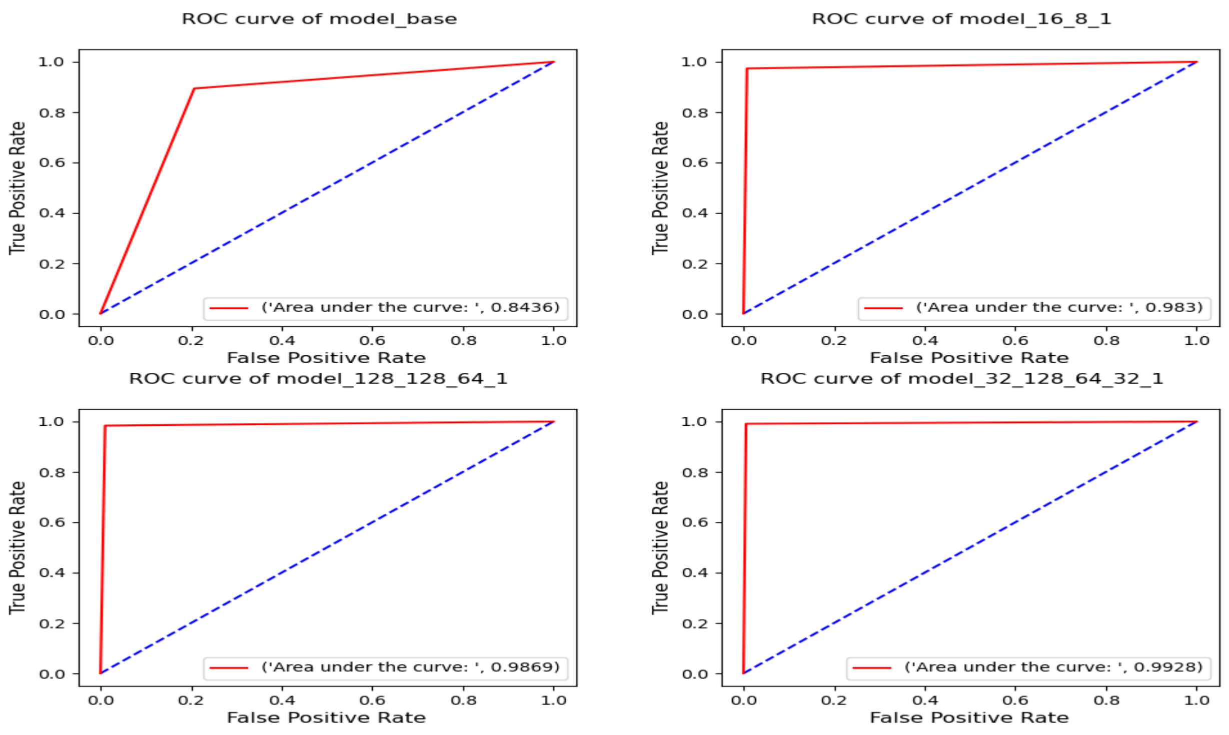
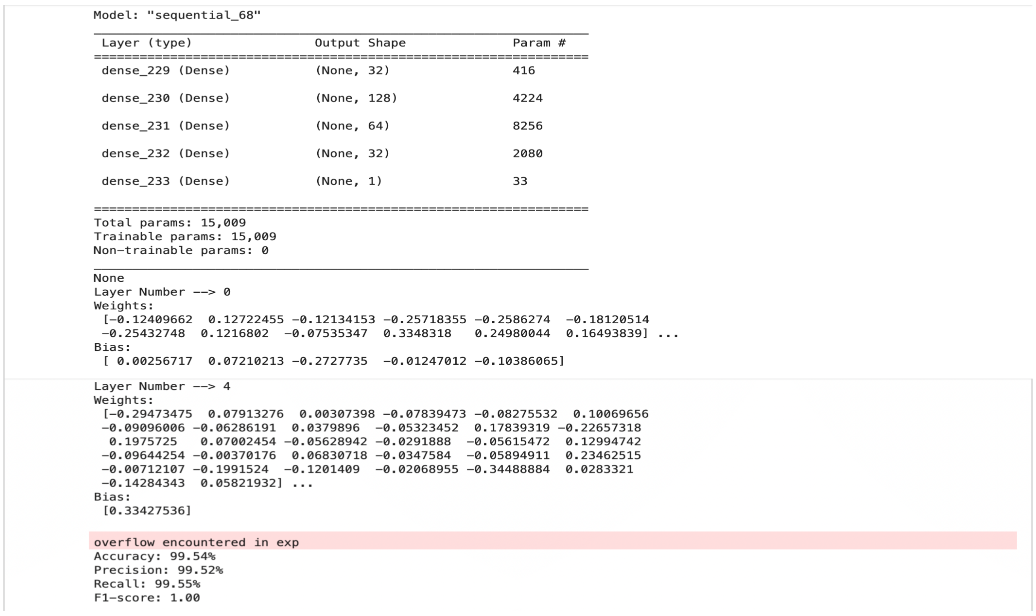


Figure 9: Input Data Distribution Histograms - Before Normalization

5.2 Custom Function and Keras Function



```
Model: "sequential_68"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_229 (Dense)           (None, 32)                416

 dense_230 (Dense)           (None, 128)               4224

 dense_231 (Dense)           (None, 64)                8256

 dense_232 (Dense)           (None, 32)                2080

 dense_233 (Dense)           (None, 1)                 33

=================================================================
Total params: 15,009
Trainable params: 15,009
Non-trainable params: 0
_________________________________________________________________
None
Layer Number --> 0
Weights:
 [-0.12409662  0.12722455 -0.12134153 -0.25718355 -0.2586274  -0.18120514
 -0.25432748  0.1216802  -0.07535347  0.3348318   0.24980044  0.16493839] ...
Bias:
 [ 0.00256717  0.07210213 -0.2727735  -0.01247012 -0.10386065]
Layer Number --> 4
Weights:
 [-0.29473475  0.07913276  0.00307398 -0.07839473 -0.08275532  0.10069656
 -0.09096006 -0.06286191  0.0379896  -0.05323452  0.17839319 -0.22657318
  0.1975725   0.07002454 -0.05628942 -0.0291888  -0.05615472  0.12994742
 -0.09644254 -0.00370176  0.06830718 -0.0347584  -0.05894911  0.23462515
 -0.00712107 -0.1991524  -0.1201409  -0.02068955 -0.34488884  0.0283321
 -0.14284343  0.05821932] ...
Bias:
 [0.33427536]

overflow encountered in exp
Accuracy: 99.54%
Precision: 99.52%
Recall: 99.55%
F1-score: 1.00
```

Figure 10: Custom Function Result

6 Feature Importance Analysis

As of now, we have a pretty good idea of which model to use, the number of epochs, and a reasonable validation set to use for the network architecture. The next step is to find out which input features is redundant or insignificant.

6.1 Significance of individual features

Five input features (PM1.0, PM2.5, NC0.5, NC1.0, and NC2.5) slightly impact the overall accuracy of the model. We can see the graph below for the performance.

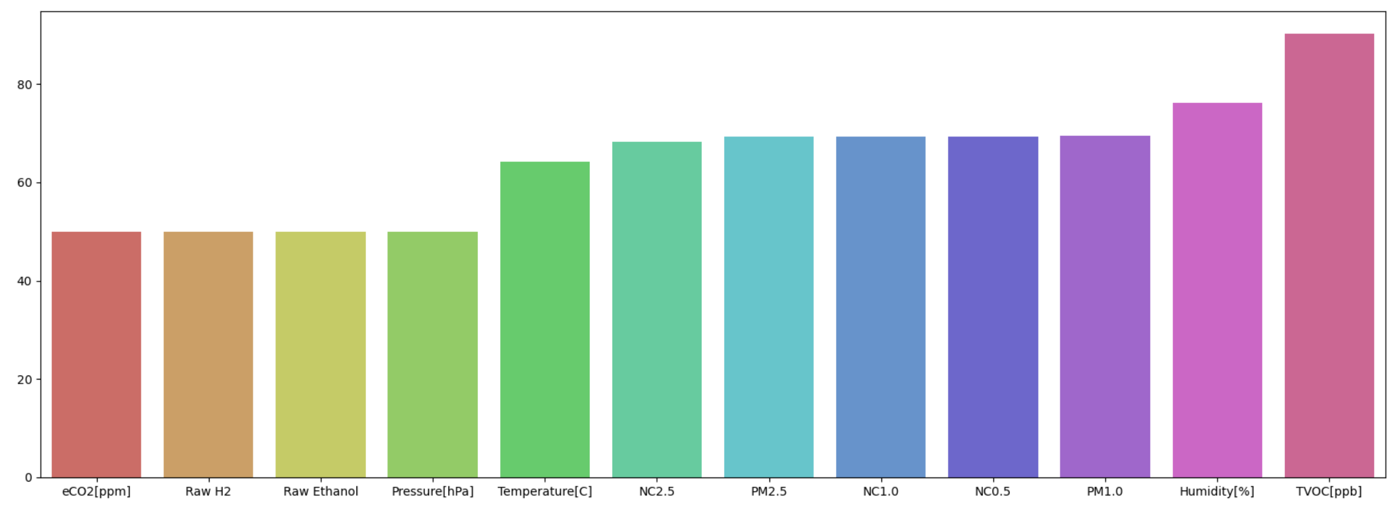


Figure 11: Significance of individual features

6.2 Performance after removing less important features

There was no significant drop (at all) in the performance after removing less important features one at a time. We can see the graph below for the performance.

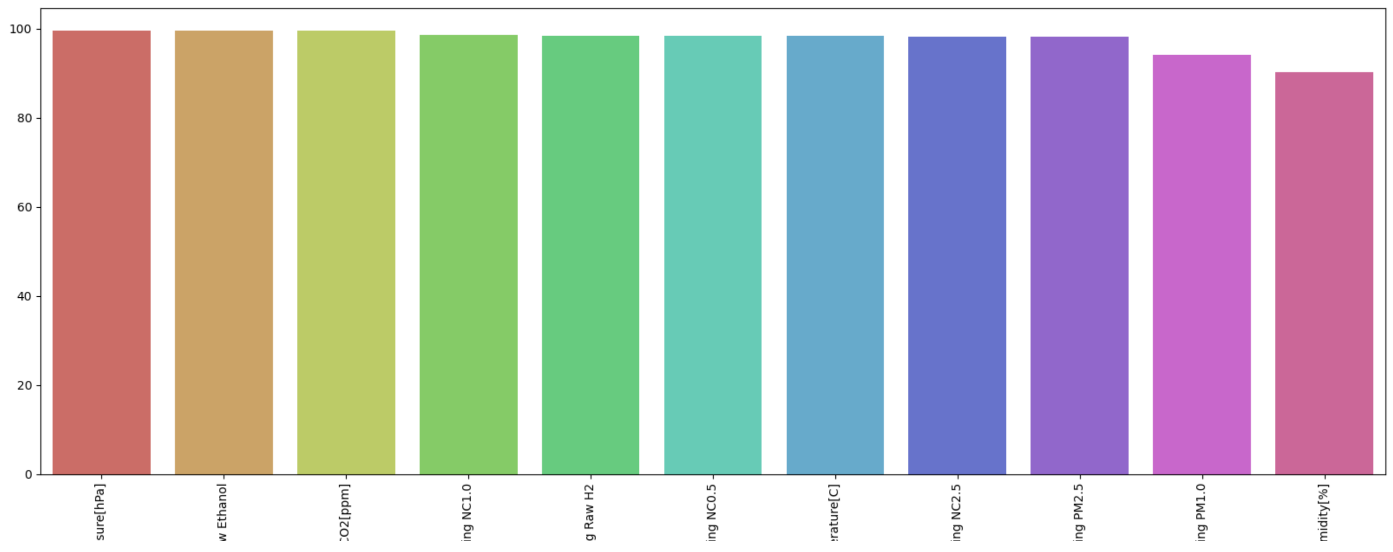


Figure 12: Performance after removing less important features

7 Challenges Faced

1. Due to the significant difference in sample numbers between the two classes, smoke detection is highly imbalanced, which leads to the model being biased towards the majority class and poor generalization. To address this, oversampling was used to balance the class distribution.

2. I encountered difficulties in determining the appropriate model architecture and number of neurons to achieve over-fitting, despite trying numerous different architectures.
3. In the early phase of this project, I didn't use an early stopping technique which in consequence causing training error and time.
4. When computing LIME values, it is not possible to use a neural network model instance object, and thus a custom predict function must be provided.
5. I encountered difficulty in selecting the appropriate explainer for neural network models, and I experimented with various options such as Linear, Tree, Deep, and Gradient. Eventually, I found that the Kernel explainer was the most suitable for neural network architectures.

8 Conclusion

To summarize, A neural network has been developed for smoke detection based on factors such as TVOC, Humidity, PM1.0, and NC0.5. The development process involved creating candidate models, training and evaluating them using a validation set. During this process, the impact of different neural network architectures on performance was tested and recorded. To obtain a generalized model, hyperparameters were adjusted based on the validation loss.

During the development of the model, it was not only determined which architecture was suitable, but it also revealed the importance of certain features to predict the output of a team. To determine the significance of each feature, the model was trained with the specific metric of interest, and its performance was evaluated on the validation set. It was found that humidity and TVOC were crucial in detecting smoke, while raw ethanol, which was initially considered significant in evaluating player performance, was deemed flawed.

Across the different phases, I have gained knowledge about several aspects, including data normalization methods, the potential to overfit data by adding an output column to the input, identifying feature importance to remove unnecessary features, effectively utilizing early stopping and model checkpointing techniques, analyzing learning curves to understand model behavior, and constructing a model function using weights and bias.

Further improvements such as cross-validation and hyper-parameter tuning are needed to enhance the project's performance. The project has enabled the acquisition of new knowledge in developing neural networks and provided insights into understanding their behavior.

9 References

1. Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. Neural Networks: Tricks of the Trade, 437-478.
2. Lundberg, S.M. and Lee, S.I., 2017. A unified approach to interpreting model predictions. In Advances in neural information processing systems (pp. 4765-4774).
3. M. Zabala and G. Ciampi, "A comparative analysis of early stopping strategies for deep learning," Neurocomputing, vol. 275, pp. 1538-1550, 2018.