Pollution Detection in a Low-Cost Electronic Nose

a Machine Learning Approach

Per Wessel Nore
Master’s Thesis Autumn 2016
Pollution Detection in a Low-Cost Electronic Nose

Per Wessel Nore

1st August 2016
Abstract

In this study, machine learning on a microcontroller-based electronic nose is proposed to identify danger in gas mixtures of nitrogen dioxide, carbon monoxide, and formaldehyde. This electronic nose is low-powered and low-cost, making new areas for pollution detection possible. This study researches the use of machine learning in combination with low-cost chemical sensors. The machine learning can filter the noise in the sensors and to some extent compensate for the influence of disturbances such as a change in humidity, temperature, and cross-sensitivity. With reliable, cheap sensors it is possible to produce low-cost electronic noses that may detect environments which represents a health risk.

By using a classification approach, it is possible to identify the compound of a threat and separate safe environments from dangerous. A Feed Forward Neural Network (FFNN) and a Random Forest Classifier (RFC) demonstrates a 96 % accuracy to identify harmful concentrations of nitrogen dioxide, carbon monoxide, and formaldehyde using the datasets collected during this study.

Keywords: Machine learning, electronic nose, pollution detection.
Preface

This master thesis is written at the Robotics and Intelligent Systems (ROBIN) research group, at the Department of Informatics, at the University of Oslo. The origin of the project was at chemiSense Inc. chemiSense is a company based at the UC Berkeley incubator QB3, developing a low-cost, low-powered electronic nose for pollution detection. From June to September 2015 I worked as a software engineer with pattern recognition and microcontroller programming on chemiSense electronic nose technology. I brought data from this project back to Norway, where I started a collaboration with the Norwegian Institute for Air Research (NILU) gaining access to their labs and receiving guidance on the project. This master thesis took form with the help from Kyrre Glette (ROBIN), and Morgan Kjølerbakken (NILU) and Brian Kim (chemiSense).

I would like to express my deepest gratitude to my supervisor Kyrre Glette, Postdoctoral Researcher at the Robotics and Intelligent Systems Department, at the University of Oslo for the support for this project and his continuous guidance.

My sincere thanks to Morgan Kjølerbakken, Professor at the Physics Department, at the University of Oslo for giving me access to resources at Norwegian Institute for Air Research, and the chemiSense team for giving me access to the chemiSense board, laboratories, financial support, and guidance.

I also thank all my fellow students at Robotics and Intelligent Systems research group, friends and family, and especially Ellen K. Sveen for support, feedback, and motivation through this project.
Contents

1 Introduction 1
   1.1 Outline .............................................. 4

2 Background 5
   2.1 Electronic Noses ........................................ 5
      2.1.1 chemiSense ......................................... 5
      2.1.2 RAE’s Electronic Nose’s ............................ 10
   2.2 Norwegian Regulations for Air Quality .................. 11
      2.2.1 Detecting Gases ..................................... 11
   2.3 Data Analysis Approach .................................. 13
      2.3.1 Pattern Recognition in Electronic Noses .......... 13
      2.3.2 Pre-Processing and Feature Extraction for Gas Sensor
            Arrays .................................................. 13
      2.3.3 Classification ....................................... 17
      2.3.4 Artificial Neural Network ........................... 18
      2.3.5 Hyper-Parameter Tuning .............................. 26
   2.4 Validation and Performance ............................... 27
      2.4.1 Over-Fitting Vs. Under-Fitting ........................ 27
      2.4.2 Cross-Validation .................................... 27
      2.4.3 Precision and Recall ................................ 30
      2.4.4 Cohen’s Kappa ...................................... 31
      2.4.5 There is No Free Lunch .............................. 31

3 Approach and implementation 33
   3.1 Data Gathering and Processing ........................... 33
      3.1.1 Data Gathering ...................................... 33
      3.1.2 Selecting Data ...................................... 36
      3.1.3 Formatting and Cleansing Datasets .................. 38
      3.1.4 Labels for Classification ............................ 39
   3.2 Development Environment ................................ 43
      3.2.1 Python .............................................. 43
      3.2.2 Rapidminer .......................................... 44
      3.2.3 Linear Regression Models in R ...................... 44
   3.3 Approach ............................................... 44
4 Experiments and Results
4.1 Experimental Data ........................................... 47
   4.1.1 Population Evaluation ................................ 47
   4.1.2 Feature Extraction ..................................... 49
4.2 Experimentation and Early Results .......................... 52
   4.2.1 Classifiers ............................................. 52
   4.2.2 Preprocessing .......................................... 53
   4.2.3 Results ................................................. 54
   4.2.4 Analyzing Early Results ................................. 54
4.3 Final Experiments and Results ............................... 55
   4.3.1 Algorithmic Tuning and Hyper-Parameters ............. 56
   4.3.2 Final Experiments ...................................... 57
4.4 Analyzing Final Results .................................... 58

5 Conclusion 47
5.1 Discussion .................................................. 61
5.2 General Conclusion .......................................... 62
5.3 Further Work ............................................... 64
   5.3.1 Automatic Baseline Detection .......................... 64
   5.3.2 Long-Term Drifting ..................................... 64
   5.3.3 Larger Array of Sensors With Embedded Reference
        Sensors ................................................. 64
   5.3.4 Refined Sensor Data .................................... 65
   5.3.5 Hardware Implementation ............................... 65

viii
List of Figures

1.1 The first electronic nose ........................................... 2
1.2 chemiSense prototyping board designs for the electronic nose. 3

2.1 chemiSense current electronic nose board, and the one used in this thesis. ........................................... 6
2.2 The sensors used through this project ................................. 6
2.3 Example of the amplification circuit, where the resistance in resistor $R_f$ modifies the magnitude of the amplification on the signal. ........................................... 8
2.4 Supervised Classification ........................................... 18
2.5 Feed forward neural network ........................................... 20
2.6 The Perceptron is illustrated with an array of inputs. The inputs are first weighted, and then processed by the transfer function, followed by the activation function. ....................... 21
2.7 The figure shows the optimal margin where $W^*X - b = 0$ is the optimal hyperplane, the optimal margin and the support vectors at the highlighted circles. ....................... 23
2.8 The figure illustrates a nonlinear separation, and the corresponding after a data transformation with a hyperplane as separator ........................................... 24
2.9 A simple decision tree, that is illustrated through weather analysis[46]. ........................................... 25
2.10 As illustrated, for each experiment, different parts of the data-set is used for training and validation. This is a k-fold cross-validation, where $k = 5$. ........................................... 28
2.11 Raw data distribution ........................................... 29
2.12 Shuffled data distribution ........................................... 29
2.13 Example of precision-recall for a multi-class problem displayed in a confusion matrix. ........................................... 31

3.1 The figure shows the climate chamber, chemiSense electronic nose, and oven used for the experiments. ....................... 34
3.2 ExSense ........................................... 36
3.3 Figure showing the exposure time-series data for the chemiSense system during exposure to cigarette smoke over 850 samples. 38
4.1 Figure showing the distribution for each class in the eight-class population. ............................................. 48
4.2 Figure showing the distribution for each class in the twenty-seven class population. ................................. 48
4.3 Figure showing the collected data represented by three principal components. The color represents the label of the class for each data point. .................................................. 49
4.4 NO$_2$ data ........................................................................................................... 50
4.5 Cumulative variance of PCA .................................................................................. 51
4.6 Figure showing the moving average filtering on data from the NO$_2$ sensor, the red dots are averages over fifty blue dots. 52
4.7 The box plots comparing the execution time for 864 samples. The three final algorithms are executed 6 times each on an identical computer. ................................................................. 58
4.8 ANN with moving average feature extraction, and PN-unit scaling with an accuracy on 95.75 and Kappa at 94.3 %. ... 58
4.9 SVM with PN-unit preprocessing with an accuracy on 90.17 and Kappa at 82.6 %. .............................................. 59
4.10 RFC with PN-unit preprocessing with an accuracy on 96.12 and Kappa at 93.2 %. .................................................. 59
List of Tables

2.1 The table shows specifications for different pollution detection systems by RAE Systems. .................................................. 10
2.2 Recommended limits in Norway for the three different toxic gases; Formaldehyde, Carbon Monoxide, and Nitrogen Dioxide .......................................................... 12
2.3 Example of preprocessing relationships .......................... 15
2.4 Over-fitting and Under-fitting ........................................ 28
3.1 Reference sensors used through this project .................. 35
3.2 Sensor baselines in testing environment with clean air. .. 37
3.3 Count of data points per sensor both in experiments, and in baseline environments. .......................................................... 37
3.4 Functions with its parameters to format datasets and generate missing data points. ......................................................... 39
3.5 The limits define what amount of each chemical that are safe, dangerous (TWA) and what is critical (STEL). With this, we can represent amounts to define classes in our population. ......................................................... 40
3.6 PPM range in the data .................................................... 40
3.7 Class representation for an eight-class solution ............ 41
3.8 Class representation for an twenty-seven-class solution. . 42
4.1 This table describes the eigenvectors for each features with each principal component ....................................................... 51
4.2 This table describes the eigenvalues for each principal component. S-dev is the standard deviation. ............................. 51
4.3 The table shows results for early iterative tests with each algorithm, to identify what algorithms to use in the final tests. 53
4.4 Early results .............................................................. 55
4.5 Table showing the results after the optimizing of hyper-parameters and selection of the best performing combinations of feature extraction, preprocessing, scaling, and classifier. ............................................. 57
Chapter 1

Introduction

Human exposure to high concentrations of pollutants can be harmful and in worst case deadly. In industrial and trafficated environments, fabrics or oil refiners, they often lack the ability to measure harmful pollutants on a day to day basis. Polluted environments represent a problem for people who work with materials and machines that emit pollutants. Polluted environments can be found in various types of occupations from painters to industrial workers. Exposure to gases like nitrogen dioxide (NO$_2$), carbon monoxide (CO), formaldehyde (CH$_2$O), and ozone (O$_3$), can among others affect asthmatics [39], Asphyxiation, and Chemical anoxia [40]. Consequently, lead to lung damage, methemoglobinemia [40] and cancer (lung, nasopharynx, oropharynx, nasal passages) [41].

Ideally, people would posses the facility to make sure that they are protected from toxic pollutants. Currently, this is not commercially done as a consequence of it being too expensive. To improve this situation, it is necessary to develop an affordable solution which is trustable over time. Julian Gardner and Philip Barlett refers in their book *Electronic Noses Principles and Applications* to environmental monitoring as "one of the most demanding application areas for electronic noses" [13]. They argue that in order to supply an annual market with millions of units, the product cost needs to be down below $150. Potential useful cases could then be to quantify and monitor airborne odors in city’s, inside buildings, checks of household air and polluted environments where workers are subjected every day. To make a low-cost system, we have to take advantage of cheap sensors, understand how they operate and then analyze their behavior. Developing this will make it possible to construct an electronic nose within the requested price range.

During the last four decades, considerable research has been directed towards developing electronic olfactory systems or electronic noses [33], see figure 1.1. During this time, automatic odor detection systems have
been applied in many industrial applications like agriculture [26], indoor air quality [56], environmental monitoring [6], quality control of food products [44, 57], and medical diagnosis [28]. Some development has been directed towards the construction of low-cost and compact electronic noses [52], which have led to a number of commercial electronic nose products [47]. These products are still too expensive for widespread use; with the consequence that there are still few products with these properties for the general public and commercial use. The use of a microcontroller in electronic nose-systems well documented [52] [24] [12] [22]. However, the possibility of implementing sophisticated machine learning algorithms on low-cost electronic noses has not been widely explored. This thesis will report on a machine learning approach for processing the raw data from cheap chemical pollution sensors. chemiSense has been developing an electronic nose to deliver an output considering the surrounding air quality. This design allows a very low-cost electronic nose around $50 as opposed to a range of $1460 to $6700, and is at the same time power efficient in comparison to other electronic nose devices [33].

Figure 1.1: "The first electronic nose device was reported by Persaud and Dodd [45]. The electronic devices simulate the different stages of the human olfactory system, resulting in volatile odor recognition, which can now be used to discriminate between various bacterial infections." [34]

The chemiSense electronic nose is a digital mimic of the human olfactory system for pollution detection, see design in figure 1.2. chemiSense is a company that focuses on making low-cost, low-powered devices to detect both gases and particulate matter pollution. chemiSense wants to make a product that combines the use of proprietary sensors and off-the-shelf sensors, both types with the strength of being small, power efficient and can be produced at a low-cost. Using sensors with these strengths also comes at an expense; the stability and accuracy of these sensors are somewhat difficult to predict. Cheap sensors often have the challenge of being highly cross-sensitive, temperature dependent, sensitive to noise,
and have issues with baseline drifting. In this case, chemiSense has been developed a system-on-chip solution to integrate the sensors into a product controlled by a low-energy microcontroller. The chemiSense electronic nose purpose is to identify toxic environments, and to do this, we need to understand what environments the data gathered by the chemiSense system represents. In brief words, the issue through this project is how to understand raw data sampled by the microcontroller, and classify that in respect to all interfering factors we can observe. The product should also identify what specific gases are present. By using techniques for preprocessing, and machine learning we will demonstrate how different approaches using these techniques perform to classify safe and unsafe environments with chemiSense’s system.

Figure 1.2: chemiSense prototyping board designs for the electronic nose.

This project goal is to evaluate the use of machine learning as compensation for the unpredictability in low-cost chemical sensors. The machine learning can hypothetically correct the inferring inputs that come with low-cost sensors. The low-cost system is chemiSense’s portable and affordable electronic nose, opposed to RAE System’s state of the art electronic noses. In more detail, the project has included data gathering and data cleansing, system design and hardware programming. The main subject of research has been on the evaluation and implementation of preprocessing and machine learning techniques common for electronic noses with data from a sensor array generated by the chemiSense board. Applying pattern recognition and machine learning on electronic noses are well documented, as well as sensors embedded in this system. However, no research is found evaluating machine learning as an approach for the pattern recognition with data from similar chemical sensors used through this project.

This project demonstrates that by using machine learning, we are able to differentiate between safe and dangerous environments for
Carbon Monoxide, Nitrogen Dioxide, and Formaldehyde in different climates. Both the classifiers Artificial Neural Network and Random Forest Classifier has been proven to do this with over 96% precision.

1.1 Outline

The thesis is divided into four additional chapters. (i) Background, (ii) Approach and Implementation, (iii) Experiments and Results, and (iv) Conclusion. The background chapter present the electronic nose, regulatory limits for air pollution, and the machine learning approach. Approach and implementation present how the data has been gathered and the software development environment. Experiments and results address the experiments done and an accordingly analysis of its results. Lastly, the conclusion chapter goes through a discussion of the thesis, and a general conclusion.
Chapter 2

Background

This chapter attempts to give an introduction to relevant research and methods for this project. An introduction to chemiSense’s electronic nose is given first, followed by regulations for air-quality monitoring. The next part introduces the data analysis with a brief introduction to common approaches for data analysis in electronic noses, and a more thorough run through the algorithms applied in this project.

2.1 Electronic Noses

An electronic nose is an instrument which comprises an array of chemical sensors that identifies the specific components of an odor in chemical detection with an appropriate pattern recognition system.

2.1.1 chemiSense

The chemiSense electronic nose is a printed circuit board in a system-on-chip design made as a mimic of an olfactory system tailored for pollution detection. A system-on-chip is an integrated circuit that incorporates all electronic components into one single chip substrate. Figure 2.1 shows the current system-on-chip design. In this section, we will go through the chemiSense board, relevant software, and hardware customization, and factors that are relevant for the data processing.

The chemiSense’s board is embedded with the following hardware and software customization:
Figure 2.1: chemiSense current electronic nose board, and the one used in this thesis.

**Sensors**

The chemiSense board architecture contains a combination of different types of chemical sensors. These are electrochemical sensors, metal oxide semiconductors, and polymer-chemo resistors.

(a) Nemoto’s NAP-550 NO₂ sensor  
(b) Nemoto’s NAP-505 CO sensor  
(d) The DART-ure and humidity formdehyde sensor  
(c) The temperature and humidity sensor

Figure 2.2: The sensors used through this project

**Electrochemical**  Besides temperature, humidity, and pressure, all sensor data used through this thesis is sampled from electrochemical sensors. The electrochemical gas sensor is based on the electrochemical oxidation and operates on the same principles as liquid electrolyte fuel cells. They are commercially bulky with a diameter around 1 to 3 cm, however they have the advantages of operating at room temperature. Having low-power consumption, they are robust and have operating working lives of around two years.
• NAP-550: The Nitrogen Dioxide (NO$_2$) sensor is made by Nemoto, which can be seen in figure 2.2. This is an electrochemical gas sensor with three electrodes for the detection of NO$_2$ in the range of 0-30ppm, tailored for a variety of gas detection applications. The sensor exhibiting high performance with long-term stability, this sensor is suitable for portable, and disposable gas detection instruments or fixed gas detection heads alike. The NAP-550 is a version of the popular NT-NO2, designed to be mechanically compatible with the NAP-505 CO sensor. Nemoto’s porous electrode technology enables accurate gas detection with high sensitivity. The mechanical design of the sensor gives optimum gas diffusion characteristics, and the hermetically sealed enclosure prevents costly electrolyte leakage [32].

• NAP-505: The Carbon Monoxide (CO) sensor is Nemoto’s NAP-505. It is described as a Low-cost Electrochemical CO Gas Sensor, which can be seen in figure 2.2. The NAP-505 Gas Sensor is a low-cost 3-Electrode Electrochemical cell designed for the detection and measurement of carbon monoxide in the range 0-1000ppm, in domestic carbon monoxide detectors, fire detectors and air quality monitors [31].

• Dart-Formaldehyde: The formaldehyde (CH$_2$O) sensor is a two-electrode electrochemical sensor operating at the diffusion principle, which can be seen in figure 2.2. In the presence of CH$_2$O gas, a small direct current is produced; the sensor requires no power supply of its own, but this current requires amplification to make it readable using external data collection equipment. The CH$_2$O sensor has the weakness of being very cross-sensitive to inter sulfur dioxide, and ethanol, but have the strength capable of detection CH$_2$O as low as 30PPB [30].

**Humidity and Temperature**  Humidity and temperature are measured by the HTU21D (F) shown in figure 2.2. The HTU1D (F) is a digital dedicated humidity and temperature transducers for OEM applications where reliable and accurate measurements are needed. Direct interface with a micro-controller is made possible with the module for humidity and temperature digital outputs. These low-power sensors are designed for high volume and cost sensitive applications with tight space constraints [48].

**Other sensors on the chemiSense board**  The system also consists of three other sensor types:

• chemiSense proprietor sensors that are various types of coated conductive polymers to detect Volatile Organic Compounds (VOCs).
The only metal oxide semiconductor (MOS) sensor on the board is a compact sensor for ozone. The MOS sensors material is heated up to a high temperature such as 400° C, and thereby making MOS sensors very sensitive to temperature and humidity variations.

The system also measures particle matters with a SHARP Compact Optical Dust Sensor measuring pm2.5.

Due to lack of data as a consequence of the problem with recreations of sensor performance, we will not focus on these sensors through this thesis.

**Amplification circuits**

An electronic amplifier is an electronic component that can increase the power of a signal. The amplifier modulates the output of the sensor based upon the properties of the input signal. By taking power from a power supply and controlling the output to match the input signal shape with a larger amplitude, we can modify an amplifier to output a signal with an amplitude of our requirements. In this case, the power-supply the circuit is controlling is the sensor itself. Illustration of an opamp is shown in figure 2.3

![Amplification circuit diagram](image)

**Figure 2.3:** Example of the amplification circuit, where the resistance in resistor $R_f$ modifies the magnitude of the amplification on the signal.

**Inferring inputs**

Ideally, a sensor would give an output that is only related to one single desired odor. In practice, chemical sensors are almost always sensitive to
quantities that we refer to as inferring inputs.

**Cross-sensitivity** Cross-sensitivity is a weakness for almost every sensor, where the sensor responds to other gases that are not filtered out and can react on the core of the sensor. Every sensor are influenced by some form of cross-sensitivity. We have to be aware of potential cross-sensitive compounds when interpreting and analyzing data.

**Interference by temperature** Temperature affects the conductivity of almost all materials. All wires, resistors, capacitors and sensors the current is flowing through will be affected by temperature and have an impact on the observed values from the sensors. For the models to be reconstructable, we, therefore, have to measure the whole exact circuit, not just each sensor individually. We control the temperature at an accuracy of \(\pm 1 \, ^\circ \text{Celsius}\).

**Interference by humidity** Variations in humidity cause significant changes in many sensors. We control humidity ad relative humidity (RH) at \(\pm 1 \%\). \(\phi\) denotes RH, shown in 2.1. It is the ratio of partial pressure of water vapor \((pH_2O)\) to the equilibrium vapor pressure of water \((P^*H_2O)\) at a given temperature.

\[
\phi = \frac{pH_2O}{P^*H_2O}
\]  

(2.1)

**Other components**

**Microcontroller** The board’s brain is an Atmel ATXmega32E5 microcontroller. It is one of Atmel’s high-performance low-power microcontrollers. The microcontroller handles sampling, preprocessing, analog to digital conversion (ADC), Digital to analog conversion (DAC), and all buses. The microcontroller functions with a master/slave model, where the sensors, and peripherals like Bluetooth are slaves, and the microcontroller is the master [2].

**Signal processing on chip**

- **ADC:** The ADC on the microcontroller is sampling at a 12-bit resolution with oversampling. We do this as quantization of noise. The customization of the ADC is with the 1 MHz clock and a signed single ended mode internal with 1v reference. Consequently for the purpose of data acquisition configured with no power saving.
• Oversampling: Oversampling is the process of sampling a signal significantly higher than the Nyquist rate. Oversampling lets us reduce noise, improves the resolution and helps us avoid aliasing. If we sample a signal N times the Nyquist rate, the signals are said to be over-sampled.

**Bluetooth** For the data gathering, we added a Bluetooth device to communicate with the device while it was in testing environments. Data gathered from the chemiSense board is serially transferred through Bluetooth to ExSense (chemiSense’s DAQ software) on a computer.

**Airflow** The casing of the board is mounted with a five voltage fan to control the flow rate. Variation in flow rate can influence sensor output. In the case of using a high-temperature sensor, as the $O_3$ sensor, it is important to keep the flow rate low.

**3d design of casing** The design of the casing is 3D-printed and made to have an air-flow leading air to the sensors.

### 2.1.2 RAE’s Electronic Nose’s

RAE Systems is a global provider of wireless gas detection systems for real-time safety and security detection. They deliver state of the art electronic noses such as the ppbRAE 3000, and the toxiRAE pro. These both portable systems have the following key features shown in following table 2.1. The ppbRAE 3000 data analysis is based on supervised learning. Accuracy is not presented for either of the systems.

<table>
<thead>
<tr>
<th>RAE Systems</th>
<th>ppbRAE 3000</th>
<th>toxiRAE PRO</th>
<th>toxiRAE PRO</th>
<th>toxiRAE PRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gases</td>
<td>NO$_2$</td>
<td>CO</td>
<td>NO$_2$</td>
<td>CH$_2$O</td>
</tr>
<tr>
<td>Price</td>
<td>$6,720</td>
<td>$420</td>
<td>$520</td>
<td>$520</td>
</tr>
<tr>
<td>Resolution</td>
<td>0 ppb to 1k ppm</td>
<td>0 to 500 ppm</td>
<td>0 to 20 ppm</td>
<td>0 to 10 ppm</td>
</tr>
<tr>
<td>Range</td>
<td>1 ppb to 1 ppm</td>
<td>1 ppm</td>
<td>0.1 ppm</td>
<td>0.05 ppm</td>
</tr>
<tr>
<td>Response time</td>
<td>3 seconds</td>
<td>&lt; 15 seconds</td>
<td>&lt; 15 seconds</td>
<td>&lt; 15 seconds</td>
</tr>
</tbody>
</table>
2.2 Norwegian Regulations for Air Quality

Folkehelseinstituttet advertises guidelines for air quality in Norway. Folkehelseinstituttet started with guidelines for indoor air first in 1991. The goal was to learn and recommend pollution levels indoor that should not be exceeded. Since then they have tightened the guidelines and currently they are as follows. In this section, we will go through regulations on exposure to pollutants, in particular, \( \text{NO}_2 \), CO, and \( \text{CH}_2\text{O} \). Table 2.2 shows the regulatory limits for these gases in Norway.

2.2.1 Detecting Gases

**Nitrogen Dioxide (\( \text{NO}_2 \))**  Nitrogen Dioxide is a chemical compound in the group of several nitrogen oxides. Health effects can include: impaired lung function, eye irritation, increased susceptibility to infections, irritation and inflammation of the airways, respiratory symptoms (coughing, increased mucus production, wheezing), acute and chronic bronchitis, and asthma attacks. The most sensitive individuals have increased mortality after short-term exposure.

**Carbon Monoxide (CO)**  Carbon Monoxide sources for where levels rise in areas with heavy traffic and poor air exchange. These areas can be narrow streets, tunnels, and garages. Close to heavily trafficked roads can polluted outdoor air give high concentrations even indoors. Important indoor sources are cigarette smoking, fireplaces, and stoves. Vulnerable individuals are people with lung and heart deceases. CO hemoglobin levels above 3 % can trigger angina among vulnerable individuals.

**Formaldehyde (\( \text{CH}_2\text{O} \))**  \( \text{CH}_2\text{O} \) is a naturally-occurring organic compound. Indoor sources include a variety of products such as resins, adhesives, insulating materials, particle boards, veneer, and textiles. Health effects of short time inhalation can cause irritation of eyes, nose and throat, lacrimation, sneezing, coughing, nausea, and difficulty breathing. At significant higher concentrations (occupational exposure), an increased risk of cancer between the nasal cavity and throat, and leukemia. Certain sensitive individuals are more sensitive to visual effects.

The threshold limit values

The Threshold limit values for chemical substances in air is a definition typically for inhalation or skin exposure. It is measured in parts per
million (PPM) for gases and milligram per cubic meter for particles. This measure is used to define what threats gases purpose over a given amount of time. We have different standards on how we define threat over time, these are:

**Time weighted average (TWA)** TWA is the limit of average exposure by an 8h/day, or 40h/week work schedule.

**Short-term exposure limit (STEL)** STEL is "spot exposure for a duration of 15 minutes; that cannot be repeated more than 4 times per day with at least 60 minutes between exposure periods".

**Ceiling limit (CL)** CL is absolute exposure limit that should not be exceeded at any time. [9]

Table 2.2: Recommended limits in Norway for the three different toxic gases; Formaldehyde, Carbon Monoxide, and Nitrogen Dioxide

<table>
<thead>
<tr>
<th>Substance</th>
<th>Regulatory Limits [10] (as of 2015)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formaldehyde</td>
<td></td>
</tr>
<tr>
<td>TWA</td>
<td>50 µg/m³ = 0.0378 ppm</td>
</tr>
<tr>
<td>STEL</td>
<td>100 µg/m³ = 0.0756 ppm</td>
</tr>
<tr>
<td>CL</td>
<td>500 µg/m³ = 0.378 ppm</td>
</tr>
<tr>
<td>Carbon Monoxide</td>
<td></td>
</tr>
<tr>
<td>TWA</td>
<td>10 mg/m³ = 8.11 ppm</td>
</tr>
<tr>
<td>STEL</td>
<td>25 mg/m³ = 20.3 ppm</td>
</tr>
<tr>
<td>CL</td>
<td>80 mg/m³ = 64.9 ppm</td>
</tr>
<tr>
<td>Nitrogen Dioxide</td>
<td></td>
</tr>
<tr>
<td>TWA</td>
<td>40 µg/m³ = 0.0197 ppm</td>
</tr>
<tr>
<td>STEL</td>
<td>100 µg/m³ = 0.0494 ppm</td>
</tr>
<tr>
<td>CL</td>
<td>300 µg/m³ = 0.148 ppm</td>
</tr>
</tbody>
</table>

A description of the conversion to PPM follows. A micro liter volume of gas is in one liter of air which is equal to 1 ppm:

\[
1 ppm = \frac{V_m}{M} \times \frac{1 \mu g \text{ gas}}{1 L \text{ air}} \tag{2.2}
\]

Where:

- \(V_m\) is standard molar volume of ideal gas (at 1 bar and 273.15 K)
- \(M\) is molecular weight of gas given as g/mol.
• $V_m$ is $22.71108$ $L/mol$
• $NO_2$ has a molecular weight of $41.01g/mol$
• $CO$ has a molecular weight of $28.01g/mol$
• $CH_2O$ has a molecular weight of $30.031g/mol$

2.3 Data Analysis Approach

This section seeks to give a perspective on common pattern recognition and machine learning approaches applied in electronic noses and sensor systems. It will also give an introduction to the techniques demonstrated through this thesis.

2.3.1 Pattern Recognition in Electronic Noses

Considerable effort has been directed towards the use of metal oxides for gas detection since the discovery by Brattain and Bardeen in 1953 that adsorption of a gas on the surface of a semiconductor can produce a significant change in electrical resistance [4]. The use of an array of sensors with partially overlapping sensitivities approach have been adopted by several professions and applied commercially; many have managed to sort out the problem with cross-sensitivity by doing so. Multiple combinations of pattern recognition, signal processing, and machine learning techniques have been used to determine correlations in datasets from both sensor arrays and none arrays with multi-dimensional datasets [36]. J. Gardner demonstrates in "Detection of Vapours and Odors from a multi-sensor Array Using Pattern Recognition Part 1. Principal Component and Cluster Analysis" [14], the use of regression, unsupervised hierarchical cluster analysis and Principal Component Analysis (PCA) for detection in multi-sensor arrays. He emphasizes that it is appropriate to apply a normalizing procedure in these methods. In his results, he shows that all approaches demonstrated gave a good classification for alcohols, but restricted for beverages a result of combining gases.

2.3.2 Pre-Processing and Feature Extraction for Gas Sensor Arrays

The goal with preprocessing is to select the parameters that contain descriptive information from the sensor array response. The choice
of parameters can have a significant effect on the performance of the following modules in the system [16]. Signal processing, Baseline manipulation, compression, and normalization are general steps that belong within the preprocessing step [21] [20]. In [27], cross-reactive arrays of chemical sensors coupled with different pattern recognition methods have been described. They demonstrated numerous applications using different types of sensors in an array format, some metal oxide semiconductors and some polymer-coated. They concluded that in some cases it is necessary to include a preprocessing approach combined with pattern recognition. The optimal preprocessing methods depend on the type of sensor used and the goal of the analysis. A common technique used is normalization. Principal component analysis, linear discriminant analysis, and partial least squares are best used in cases where sensor arrays are known to respond linearly. They also mention cluster analysis, computational neural networks, and fuzzy ARTmaps as alternatives to the previously mentioned linear techniques. These methods offer the potential to not only model linear systems, but also model non-linearity.

Baseline manipulation  Baseline manipulation transform the sensor response about its baseline, for the purpose of differentiating enhancement and drift compensation. There are commonly three different baseline manipulation methods: difference, relative, and fractional manipulation [19].

Normalization  Next step is normalization; we normalize to prepare the feature vector for the pattern analysis. We want the features to be on a suitable format for each algorithm. Normalization can have a large impact on the performance (complexity, training time, and accuracy) on each algorithm. Normalization can be mathematically more efficient if one plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples. Normalization may help correcting the error in sensors caused by natural variations in odor concentrations but at the cost of increased noise from sensors with low signals. There are two types of normalization procedures, local methods, and global methods. The local methods utilizes vector normalization to compensate for sample-to-sample variations caused by analyte concentration. Global methods are typically used to ensure that sensor magnitudes are comparable. Sensor auto-scaling and sensor normalization are commonly employed in electronic noses [19].

Feature scaling  Feature scaling is a common normalization technique. Scaling before applying both SVMs and ANNs is very important. The
main advantage of scaling is to avoid great numeric ranges in attributes, which makes the computational load too expensive, and rather work with small numeric ranges. Another advantage is to avoid difficult numerical calculations. In the example of a SVM where the kernel usually depends on the inner product of feature vectors, large attributes might cause exhausting numerical operations. Two common feature scaling techniques maps the population to a unit interval 0 to 1 or to the positive, negative unit interval (PN-unit) -1 to 1.

Many elements used in the objective function in learning algorithms assume that all features are centered around zero, and have variance in the same order, hence scaling between for instance -1 and 1 will thereby increase performance. Features that are orders of magnitude larger that others might dominate the objective function and therefore make the estimator unable to learn from other features as expected. One difficulty with scaling is scaling data with outliers. If the data contains many outliers, scaling using the mean and variance may not work well. In cases like this it is common to use a robust scaler. In table 2.3 we show the relationship of different types of normalization.

Feature scaling to unit interval is given by the following equation:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$ (2.3)

Other normalization techniques:

• Fractional: Fraction is when the baseline is subtracted and then divided from the sensor response.

• Differential: The baseline is subtracted from the sensor response.

• Relative: The sensor response is divided by the baseline.

Table 2.3: Example of preprocessing relationships

<table>
<thead>
<tr>
<th>Normalization</th>
<th>Temperature</th>
<th>RH</th>
<th>NO$_2$</th>
<th>CO</th>
<th>CH$_2$O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw baseline values</td>
<td>24</td>
<td>42</td>
<td>1100</td>
<td>1100</td>
<td>1200</td>
</tr>
<tr>
<td>Scaling [0, 1]</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Scaling [-1, 1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Differential</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Raw max values</td>
<td>60</td>
<td>70</td>
<td>3298</td>
<td>3298</td>
<td>3289</td>
</tr>
<tr>
<td>Scaling [0, 1]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Scaling [-1, 1]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Drift compensation**  A strong limitation in sensor array technology, in addition to selectivity and sensitivity constraints, arise from sensor drift. Like Gardner mentions in [14] the performance of semiconductor oxide sensors are unfortunately limited by problems as long-term drift in the baseline. This phenomenon degrades the stability of the device and makes obsolete the models built in order to recognize and quantify volatile. [58] Drifting results in a random temporal variation of the sensor response when exposed to the same gasses under identical conditions. There are ways to solve this suggested the most efficient drift compensation is periodic re-calibration with a reference gas that is chemically stable over time and highly correlated with the target analytes regarding sensor behavior [23]. The array response to the calibration gas can then be directly subtracted from the response to the analytes [11], used to infer a temporal drift model for each sensor [23] or for the entire array [17] [1] [18] [19]. Even if technical solutions to adjust for drifting have been proposed, we will not focus on solving drift compensation in this thesis.

**Feature extraction and selection**

The goal of the feature selection is to find an optimal subset of M sensors or features that maximize information content or predictive accuracy. We can do this by evaluating each feature individually and selecting the M features with the highest score. In 2.4, the logic for extracting and selecting features is shown.

\[ f : x \in \mathbb{N} \rightarrow y \in \mathbb{M} \ (M < N) \]  

Dimensionality reduction is required in most cases of sensor arrays to prohibit the co-variance matrix of the dataset to become singular. Either through feature selection or feature extraction. Feature extraction plays an major role in every chemo-sensor applications [43]; it is defined as a transformation mapping the sensor response to a space of lower dimension preserving the most meaningful portion of the information contained in the original sensor signal [53], that still possesses most of the information from the original feature vector. Linear feature extraction techniques like principal component analysis and Fishers’s discriminant analysis (LDA) are the most common for electronic nose-applications. PCA is a procedure that uses orthogonal transformations to convert a set of observations to principal components. This transformation is defined in such a way that the first principal component has the largest possible variance. LDA is a generalization of Fisher’s linear discriminate used to find features that separates two or more classes of objects or events; this may be used as a linear classifier or for dimensionality reduction [19].
Moving Average

Moving average approach is another feature extraction used at time-series data. It is a calculation to analyze data points by creating series of different subsets of the data. These subsets are represented with the average for the current subset. Mathematically, moving average is a type of convolution, an example of a low-pass filter used in digital signal processing. The simple moving average (SMA) is the mean of the previous $n$ data points; we represent it with the following equation, where $p_{M-i}$ represents the $i$'th previous measurement:

$$SMA = \frac{1}{n} \sum_{i=0}^{n-1} p_{M-i}$$

(2.5)

2.3.3 Classification

With classification, the goal is to generate a class label for an unknown feature vector. There are a vast amount of classification techniques that are applied in electronic noses, like K-nearest neighbors, K-means, Artificial neural networks, Support vector machines, and Quadratic Classifiers. Accordingly, we will go through the classifiers that are most relevant to embedded on a chip with limited computational power and are of the supervised learning model.

Machine Learning and Pattern Recognition

Machine learning (ML) is a branch evolved from combinations of pattern recognition and traditional artificial intelligence. There are different machine learning techniques that may be suitable to solve this problem. Machine learning is usually separated in supervised learning, and unsupervised learning. Supervised learning is the task of inferring a function from labeled training data, while unsupervised learning is the task of inferring a function to describe a hidden structure from unlabeled data. An illustration in figure 2.4 shows the workflow in supervised learning. Through this project, we will apply different supervised learning approaches but will discard the unsupervised learning since we have labeled data. In this project, we will evaluate the performance of supervised learning algorithms that has the advantages of having a time complexity running in real-time that lets us do pattern recognition on a chip. These machine learning techniques build a model based on a training dataset to classify or predict the elements of another dataset of interest called validation dataset. In this case, pattern recognition by
supervised learning is an approach that lets us train a model to do a classification. The advantages of having labeled data let us evaluate the performance of a model easily, and we can adjust it to classify the output which is the correct output according to its input. Doing this over and over again we train the model to converge towards the best possible performance on our classification. The algorithms we are evaluating will be tested in a combination of feature extraction by prepossessing techniques. We want to evaluate what combination of preprocessing and machine learning approaches performs best. The next sections will introduce these supervised learning algorithms.

Figure 2.4: Supervised Classification. During training, a feature extractor is used to convert each input value to a feature set seen in (a). These feature sets, which capture the basic information about each input that should be used to classify it, are discussed in the next section. Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model. During prediction, the same feature extractor is used to convert unseen inputs to feature sets. These feature sets are then fed into the model, which generates predicted labels, seen in (b). [3]

2.3.4 Artificial Neural Network

An ANN, usually called neural network (NN), is a mathematical computational model that is commonly known to be in the family of models inspired by biological neural networks. ANNs uses properties known from central nervous systems of animals, in particular, the brain to estimate decision functions. These artificial neurons are called perceptrons and are combined in a directed graph structure to make a network. ANNs can be found in many different fields of research, such as signal processing, control, natural language processing and pattern recognizing. That being a
result of the advantage ANNs has to generalize. Most cases of ANNs is adaptive systems that change its structure based on external or internal information that flows through the network during learning. Modern neural networks are used to model complex relationships in data to recognize a pattern. ANNs are generally presented as a network of interconnected artificial neurons, processing information using a connectionist approach to computation. The connections have a numeric weight that is changed when the network is learning.

Multiple approaches have concluded that neural networks have a good performance in classifying chemical compounds and greenhouse-gasses in real-time. Artificial neural networks are in different sensors detection domains capable of correctly identifying gases from combinations of sensors. In [15] they found that they were capable of identifying alcohols using an array of twelve tin oxide sensors. Combining with the use of the total sum of squared networks they could determine errors for their relative performance. They also found that the network offers considerable benefits in its ability to cope with non-linear and correlated data and outperforms both PCA and clustering.

The conclusion of [29] was that utilizing the signal from their sensor array on a neural network with back-propagation to recognize kinds and quantities of VOCs with their respective Threshold Limit Value and Lower Explosive Limit ranges. In a real-time process, the recognition system was successfully in both accurately classifying the gas species and indicating the approximate quantity of each VOC within a range of 0-2000 ppm.

In [49] they concluded that the use of transformed cluster analysis (TCA), original proposed by Nayak et al. [38] in conjunction with a conventional back-propagation neural network classifier may provide even better intelligent and reliable electronic nose system with the ability to identify target vapors successfully even in in noisy conditions. The TCA is used to transform the data by reducing the scatter of data in a cluster on the basis of mean and variance of individual gas-sensor combination by maintaining the originality of data as it is.

According to the results from [56] [7] neuro-fuzzy networks can outperform multi layer perceptrons in quantitative gas analyses. However, NFNs involve more complex learning methods when training. As in multi layer perceptrons, neuro-fuzzy systems allowed for the realization of low-cost, low-power consumption devices with higher sensitivity and enhanced selectivity.
The feed forward network with back-propagation

We can learn a model to classify by means of a feed-forward neural network. A feed-forward NN (FFNN) is a ANN where connections between the units do not form a directed cycle in comparison with other ANNs like the recurrent NN. In the feed forward network, the information moves only in one direction, from the input notes towards the output nodes, visiting the hidden layers on its way. The nodes combine to a directed graph, where each layer is fully connected to the next one as illustrated in figure 2.5.

Figure 2.5: The figure shows a feed forward neural network, With an input layer, two hidden layers, and an output layer. The nodes in each layer represent a perceptron with its connection to the next layer in the network[37]

The perceptron is a type of ANN invented by Frank Rosenblatt in 1957. It can be pictured as the simplest kind of a FFNN, a linear classifier which finds a separating hyperplane if exists. The MLP extends from the perceptron algorithm where an MLP consists of multiple layers of nodes in a directed graph where each node is a perceptron except for the input nodes. With each layer is fully connected to the next one. In many applications, the units of these networks apply a sigmoid function as an activation function. We also use a sigmoid on the output if the learning data describes a classification task. Where we would use a linear function to describe a numerical regression task.

Common activation functions are the sigmoid ($\sigma(x)$) and the tanh ($\tanh(x)$) functions.
\[ \sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.6) \]

\[ \tanh(x) = 2\sigma(2x) - 1 \quad (2.7) \]

Each perceptron in the network can be modeled in three steps as presented in Machine Learning An Algorithmic Perspective [35]. An illustration is shown in figure 2.6

1. A set of weighted inputs \( \omega_i \) that correspond to the synapses.
2. An adder that sums the input signals, equivalent to the membrane of the cell that collects electrical charge.
3. An activation function, initially a threshold function, which decides whether the perceptron fires for the current inputs.

Figure 2.6: The Perceptron is illustrated with an array of inputs. The inputs are first weighted, and then processed by the transfer function, followed by the activation function.

The FFNN utilizes a backpropagation algorithm to train the network. The backpropagation is a supervised learning method that we can divide into two phases that are repeated until the performance of the network is good enough. These phases are the propagation and weight update. In the backpropagation algorithms, the output of the network is compared with the correct answer to compute a value for a predefined error function. This error is then fed back through the network to reduce the value of the error function to some small extent. It does this by using the error information to adjusts weights of each connection. When repeating this process for a large number of training cycles the network converges towards a state where the error of the calculation is small. at this point we can say that the network has learned a certain target function to identify a pattern.
Support Vector Machine

The original Support Vector Machine (SVM) [8] were introduced by Vapnik in 1992 and is one of the most popular algorithms in modern machine learning. SVMs are based on statistical learning theory and aim to detect the location of the decision boundaries that produce the optimal separation of classes. SVM is essentially a binary classifier, where the classes are linearly separable. The SVM finds one linear decision boundary that leaves the greatest margin between the classes. The margin is defined as the sum of distance from the separator to the closest point from each of the two classes. These points are what we call the support vectors. The approach of finding the hyperplane and maximizing the margin can be solved by using Quadratic Programming optimization techniques. When two classes are not linearly separable the SVMs try to find a hyperplane that minimizes the number of miss-classifications in proportion to the optimization of the margin of the separator with a kernel function. The trade-off between the margin and miss-classifications error is controlled by one of its hyper-parameters C. The SVMs can also handle none linear decision surfaces. This is done by transforming the data into a higher dimensional feature space, and then form a linear classification. The kernel functions empower to reduce the computational cost of the data transformation, and high-dimensional feature space. Traditionally SVM’s has extended to solve multi-class problems by building a classifier for each pair of classes (one versus one strategy) or a classifier for each class with respect to the rest of the samples (one versus all strategy) [54]. In [54] they used variation of SVM called Inhibitory SVM (ISVM) [25] to provide a more robust multi-class classifier inspired by the inhibition process present in animal neural system. The result is a classifier more robust to the selection of the meta parameters when the models are built with small number of training examples. SVMs is known to provide significantly better classification performance compared to other machine learning approaches on medium sized datasets [35]. But not that well on very large datasets.

Optimal Separation for SVM  Figure 2.7 shows simple classification problem with a possible linear classification. The SVMs criteria to linear classify this dataset is a measure of distance in a direction perpendicular to the separator. We use this distance to find the largest margin to find the best separator. This is called maximum margin linear classifier. The support vectors are the data points in each class that lies closest to the classification line. The cylinder that between both classes oriented around the separator is a "no mans land". In this area it is to uncertain for the algorithm to give an unseen sample a class label.
Figure 2.7: The figure shows the optimal margin where \( W \times X - b = 0 \) is the optimal hyperplane, the optimal margin and the support vectors at the highlighted circles.

**SVM kernels**  The kernels is the SVMs ability to transform data when its not linearly separable. The idea is that if we modify the features in some way, we might be able to find a linear decision boundary. We introduce a new function \( \phi (x) \) to generate new dimensions, the new features are derived from the current ones. The kernels are the reason why the new function works, The kernels produce a transformation of the data, but at the same time it lets us do computations in original low-dimensional space, ensuring a commutable space to work in. Figure 2.8 shows an example of data transformed by a SVM.

**Multi-class SVM**  The SVM is by first glance a two-class one versus one classificator, but for the N-class problem, there is an approach. For multiple classes we train the SVM to separate one class from all other classes. In our space that would mean we need to train 8 SVMs.

In [54] they demonstrated use of a variation of SVM called Inhibitory SVM (ISVM) [25] to provide a more robust multi-class classifier inspired by the inhibition process present in animal neural system. The result is a classifier more robust to the selection of the meta parameters when the models are built with a small number of training examples.
Random Forest Classifier

The random forest classifier (RFC) is an ensemble learner of classification trees for regression, classification among others. Each tree in the ensemble casts a unit vote for the most popular class. These ensembles are often grown by generating random vectors that govern the growth of every tree in the ensemble learning. RFC through this thesis is based on trees with optimal splits. The RFC fits multiple of decision tree classifiers on different sub-samples of the dataset. By using averaging, the accuracy of the prediction is improved, as well as the control of over-fitting.

Decision tree or a classification tree is a tool that uses a tree graph of decision boundary’s and its consequences, an illustration is shown in figure 2.9. The decision tree approach has the advantage of giving meaningful representations of the data compared with other approaches. This makes decision trees easy to interpret. The goal with the decision tree is to build a classification model to predict the value of the target or class based on the input attributes. The tree consists of a tree graph structure containing a number of interior nodes and its edges. Each interior node corresponds to a test to one of the features. The number of edges of a interior node is equal to the number of possibilities values of the corresponding input, the result of the test in the node. The leaf nodes represent the label given by the input or the outcome class. The correspondence of the input and its label is represented by the path from the root to the leaf. Decision trees are generated recursively by partitioning of the attributes. The design of a decision tree requires an attribute selection measure and a pruning method. The most frequent attribute selection measures are the Information Gain Ration creation and the Gini
Index. The random forest uses the Gini Index. The Gini index measures the impurity of an attribute with respect to the class. The Gini Index can be written as:

$$Gini \text{ Index} = \sum_{j \neq i} (f(C_i, T) / |T|)(f(C_j, T) / |T|),$$

(2.8)

where $T$ is the training set, $C_i$ is some class, and $f(C_i, T) / |T|$ is the probability that the selected case belongs to the class $C_i$. For every training set, the tree is grown to the maximum depth using a combination of features, the fully grown trees are not pruned. This is one of the major advantages of the random forest classifier over the other decision tree methods, where the pruning method, and not the attribute selection measures affect the performance. Breimann [5] suggests as the number of trees increases the general error converges, even without pruning and that over-fitting is not a problem because of the Strong Law of Large Numbers [42].

![Figure 2.9: A simple decision tree, that is illustrated through weather analysis][46].

**Clustering**

Clustering is an unsupervised learning process that seeks to find similarities or spatial relationships among data samples. Techniques like Hierarchical Clustering, K-means and Self-Organizing Maps have been used to process electronic nose data. Clustering involves defining a measure between the feature vectors, defining a criterion to be optimized, and defining a search algorithm to find a real cluster. The clusters can be used as classifiers in combinations with other classification algorithms or by using...
the defined measure to calculate the distance between a new feature vector and centers of the clusters.

2.3.5 Hyper-Parameter Tuning

Hyper-parameters are the parameters that govern the underlying algorithm on a higher level than the parameters of interest. The goal of tuning these parameters is to identify a set of parameters that optimize the performance of a learning algorithm. Different approaches to this is grid search, gradient-based optimization, or a knowledge-based approach. Through this thesis, we optimize this by a combination of knowledge-based approach and grid search.

SVM: In the process of identifying suitable kernels we can evaluate the performance of both the RBF kernel and the sigmoid kernel. The SVM kernels were first tested by finding the optimal parameters for C and gamma. And then trained and tested with the cross-validation to find the maximum performance in combination with the different feature extraction and preprocessing approaches. For the SVM we evaluate:

- Kernel function
- C
- Gamma

ANN: First we start with identifying what amount of layers and perceptrons that give a good accuracy but at the same time keep the complexity at a minimum. When we start to get good results we evaluate how we can tune the algorithm with a number of training cycles and learning rate to compensate for a amount of trainable variables. When the number of layers, perceptrons and training cycle is set we can evaluate the performance of the activation function, which gives the best result. For the ANN we evaluate:

- Layers
- Perceptrons
- Training cycles
- Learning rate momentum
- Activation function

RFC: The RFC has the most trivial parameters to work with out of the three classifiers, Where we first evaluate attribute selection method and the maximum number of features. For the RFC we evaluate:
2.4 Validation and Performance

This section intends to introduce concepts regarding testing of machine learning models. We will go through how we train a model and make sure that it generalizes.

2.4.1 Over-Fitting Vs. Under-Fitting

With over-fitting, we mean that the model we made is made of noise from the data. Over-fitting occurs when a model fits its training data too well. Scientifically, this happens when an algorithm or model shows high variance but low bias. By calculating the mean squared error (MSE) on the validation sets we can evaluate if the data is over or under fitted. The higher the MSE is, the less likely the model generalizes correctly from the dataset. We evaluate over-fitting or under-fitting by using cross validation. Under-fitting as the name states is the opposite of over-fitting. It takes place when the machine learning algorithm fails to capture the underlying trend of the data. Under-fitting happens if the algorithm or model shows high bias and low variance. Figure 2.4 shows a comparison of an over-fitted and an under-fitted model.

2.4.2 Cross-Validation

Cross-validation is a technique to assess an estimate on how the performance of a model will generalize to an unseen dataset in practice. Validation is the most common way to evaluate the predictive accuracy when working with supervised learning algorithms. As a rule of thumb we usually use 30% of the data for validation and 70% for training. The training data is used to train the model to do the classification. The validation data is used to evaluate the model on data that is unseen by the algorithm. We do this to simulate a real world classification more realistically, hence giving a more credible estimate of the statistical performance of a learning operator running on an unseen dataset. In the case of this study, where the data set is not very large. We use a type of cross-validation called k-fold cross-validation. In k-fold cross-validation we partition subsets of data into n complementary subsets illustrated in 2.10. Then we iterative train on n-1 of the subsets, and test on the last. The benefit of using the k-fold
Table 2.4: Over-fitting and Under-fitting

Cross-validation is that it lets us use all data we have available to its full extent.

<table>
<thead>
<tr>
<th>Experiment 1</th>
<th>Experiment 2</th>
<th>Experiment 3</th>
<th>Experiment 4</th>
<th>Experiment 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Validation</td>
<td>Training</td>
<td>Validation</td>
<td>Training</td>
</tr>
</tbody>
</table>

Figure 2.10: As illustrated, for each experiment, different parts of the dataset is used for training and validation. This is a k-fold cross-validation, where \( k = 5 \).

When we use cross-validation, we show a statistical analysis on how accurately the model will perform in practice. By using cross-validation, we generate datasets to test and evaluate the model. Cross-validation also helps us limit the problem with over-fitting. Since our data consists of few but large datasets, it is important that we use a shuffling cross-validation. We shuffle the data so we minimize the risk of training the model on compound exposures we don’t represent in our testing data. The two following figures; 2.11 and 2.12 shows why shuffling data in cross-validation is very important.
Figure 2.11: The figure shows raw data, and illustrates the distribution of data points. If we were to split non-shuffled data in one training and one testing set, there would not be data in to represent the testing data in comparison to shuffled data in Figure 2.12.

Figure 2.12: The figure shows shuffled data in comparison to non-shuffled data in Figure 2.11. With the shuffled data we ensure that there is an even spread between the testing sets and the training sets of different data points.
2.4.3 Precision and Recall

Precision and recall is used in pattern recognition as information on the performance on a classification. Precision is the fraction of predictions that are relevant, while recall is the fraction of relevant instances that are retrieved. We use the precision and recall together, to express the accuracy in the classifier, where the accuracy is used to benchmark the performance each model. In figure 2.13 there is an table of precision and recall in a confusion matrix.

We represent it by following equations:

Precision (P) is defined as the number of true positives over the number of true positives plus the number of false positives (F_p).

\[ P = \frac{T_p}{T_p + F_p} \]  \hspace{1cm} (2.9)

Recall (R) is defined as the number of true positives (Tp) over the number of true positives plus the number of false negatives (Fn).

\[ R = \frac{T_p}{T_p + F_n} \]  \hspace{1cm} (2.10)

Accuracy (A) is defined as the number of true positives plus the number of true negatives over the total number of samples.

\[ A = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \]  \hspace{1cm} (2.11)

These quantities are also related to the (F_1) score, which is defined as the harmonic mean of precision and recall.

\[ F_1 = 2 \frac{P \times R}{P + R} \]  \hspace{1cm} (2.12)

- Tp = True Positives
- Fp = False Positives
- Tn = True Negatives
- Fn = False Negatives
2.4.4 Cohen’s Kappa

The kappa statistics for classification is thought to be a more robust measure than simple percentage correct prediction calculation. It measures the agreement between two raters who each classify N items into C mutually exclusive categories. The equation for $\kappa$ is:

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

where $p_o$ is the relative observed agreement among raters, and $p_e$ is the hypothetical probability of chance agreement.

2.4.5 There is No Free Lunch

Even for machine learning, There is no free lunch. Every algorithm has its strengths. Some have a low run-time complexity, some are easily trained, and some perform well on small datasets. Any elevated performance over one class of problems is offset by performance over another class. When we compare them, their weaknesses surface. When we discover how the algorithms perform on our problem and data. We can evaluate the trade-off for each approach [55].
Chapter 3

Approach and implementation

In this chapter, we first intend to explain how the data was formed and used. The second section presents the algorithmic development and testing environment.

3.1 Data Gathering and Processing

This section goes through the processes of gathering data with the chemiSense board, and what parts of the data we are using and how it is formatted. Last we go through how the data is labeled.

3.1.1 Data Gathering

This section is intended as a data gathering work through, how the data is gathered, what tools were used, and factors to take into consideration when doing so.

Environment

Since most sensors depend on the temperature and humidity, the electronic nose needs to work in different climates. We, therefore, have to use a testing environment that resembles this for the system to be reconstructable in a real world environment. The environment where our data is gathered is a climate chamber connected to a gas diluter. The diluter lets us moisturize the air and dilute air with pollutes. This lets us expose the closed environment to pollutes, humidity and clean air. The climate chamber also works as an oven where we heat the air in the climate chamber. The heated air surrounds the sensing system at the same time as it is
exposed to gases or humid air. In figure 3.1, there is an image of an experiment with the chemiSense electronic nose inside the climate chamber.

![Image of climate chamber, chemiSense electronic nose, and oven used for experiments.]

Figure 3.1: The figure shows the climate chamber, chemiSense electronic nose, and oven used for the experiments.

**Range of voltage**

The data represents a range of quantity of pollutes we want to identify. This from safe, to deadly. To identify what ranges in voltage that makes sense to work with, and at the same time making sure to have a high enough resolution for the sensing area of interest, we need to find a suitable amplification circuit for each sensor. This also has to cope with what is physically possible for the microcontroller to handle. The amplification circuits for each sensor needs to be identified for cross-referenced output voltage with PPM of pollutes. Amplification circuits with 150 kilo, 330 kilo, 680 kilo, and 1 mega ohm have been tested. The current amplification circuit used on the current sensors is with a 680 kilo ohm resistor.

**Reference sensors**

A reference sensor generates the check-sum data, and the cheap sensor data is generated by the chemiSense board.
**Temperature and humidity reference**  The temperature and humidity sensor is referenced with one RH and one temperature sensor that is embedded in the climate- and testing chamber. The sensing environment is exposed to temperatures ranging from 20° C to 70° C, and RH from 0 to 100. The embedded RH and temperature sensors are previously calibrated and are predictable.

**Pollution reference sensor**  In the process of knowing the true amount of gases present, we use a reference sensor giving us an exact measure of the pollutes of interest. The actual amount of PPMs present is manually logged with the corresponding current voltage output from the sensing system. We do this respectively for each pollute at different concentration and climates. RAE Systems deliver the two reference sensors used. The ToxiRAE Pro [51] and the ppbRAE 3000 [50] is shown in figure 3.1.

- ppbRAE measures $CH_2O$ and $NO_2$
- ToxiRAE Pro measures CO.

| ExSence | ppbRAE reference sensor | ToxiRAE Pro reference sensor |

**Table 3.1: Reference sensors used through this project.**

**ExSence**

ExSense is a tool made by chemiSense to gather and preview data directly from the chemiSense system storing it in excel files, figure 3.2 showing
Figure 3.2: This figure showing ExSense while logging data in an experiment, where the chemiSense board is exposed to formaldehyde. The O₃ sensor is reacting as one of its inferring gases, a consequence of cross-sensitivity. The other sensors are operating at baseline. ExSense stores data transmitted by the chemiSense board in an excel sheet. The data also includes current time, date, and seconds since the beginning of the measurement.

3.1.2 Selecting Data

Addressing the data used through this thesis, we need to consider what data we have, what we need, and what parts we can use. There is a strong desire to include as much data as possible; consequently, that comes at the cost of what makes sense to include concerning its quality. We should also consider what we can generate new data, identify what ranges of deviations in sensor response, and exposure limits we are working with and identifying, all dependent on each other. The data gathered is not perfect and limits us to some restraints. That being said it is enough to demonstrate the purpose of the models we are investigating and for most of the sensors on the chemiSense board. The data available limits us to exclude the O₃ sensor in further investigations. It is also not adequate to demonstrate an analysis of dealing with cross-sensitivity between the three sensors and its relevant gases. The data we have available and are using is gathered on multiple, but identical chemiSense boards at the same time and at the same location. In table 3.2 the sensor baselines is shown. In table 3.3 we show the count of data points for different gases, temperatures and RH.

Included from data available:
• Temperature for all sensors.

• Humidity for all sensors.

• Exposures to $NO_2$.

• Exposures to CO.

• Exposures to $CH_2O$.

• Real world data from cigarette smoke and car exhaust.

Excluded data:

• Exposures to $O_3$.

Table 3.2: Sensor baselines in testing environment with clean air.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Temperature</th>
<th>RH</th>
<th>$O_3$</th>
<th>$NO_2$</th>
<th>CO</th>
<th>$CH_2O$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>24</td>
<td>25</td>
<td>2500</td>
<td>1100</td>
<td>1100</td>
<td>1095</td>
</tr>
</tbody>
</table>

Table 3.3: Count of data points per sensor both in experiments, and in baseline environments.

<table>
<thead>
<tr>
<th>Data points in:</th>
<th>Temperature</th>
<th>RH</th>
<th>CO</th>
<th>$NO_2$</th>
<th>$CH_2O$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>~50k</td>
<td>~60k</td>
<td>~60k</td>
<td>~50k</td>
<td>~60k</td>
</tr>
<tr>
<td>Exposure</td>
<td>~30k</td>
<td>~20k</td>
<td>~8k</td>
<td>~20k</td>
<td>~11k</td>
</tr>
</tbody>
</table>

The real world data is included to look at edge cases and data with noise. This is data that would look as outliers in comparison with the data gathered at the laboratory. The data here is exhaust from diesel car and cigarette smoke; an example is shown in figure 3.3. In total, it is included 83000 data points with 5 five features each.
3.1.3 Formatting and Cleansing Datasets

The original data format was in multiple excel files where some or all features was represented. This was merged and put on a standard format in separate .CVS files. The experiments were features were lacking results in a challenge on how to generalize data from each experiment. How this problem is solved is demonstrated in the next section.

Missing data points

The data generated at chemiSense facility in Berkeley has some issues. In some of the lab experiments not all features where logged. This issue was solved by generating missing features for all the data points. We did this to as a criteria to make the process of testing algorithms more trivial. Table 3.4 shows what parameters are used in each function for feature generation.

Missing feature generation

The algorithmic approach demands an input where each data point contains the same features. To simplify the algorithmic experiments,
Table 3.4: Functions with its parameters to format datasets and generate missing data points.

<table>
<thead>
<tr>
<th>Temperature</th>
<th>RH</th>
<th>NO$_2$</th>
<th>CO</th>
<th>CH$_2$O</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T = f (RH)$</td>
<td>$RH = f (T)$</td>
<td>$NO_2 = f (T)$</td>
<td>$CO = f (T)$</td>
<td>$CH_2O = f (T)$</td>
</tr>
</tbody>
</table>

we generate the missing features for the data points where it is needed. When we generate features, it is important that we only generate features where there is no inference from the current exposure and the missing feature. We do this by looking up the tables for cross-sensitivity at each sensor datasheet. Experiments where the exposing chemical could cause an inferring deflection on the missing feature, where excluded from the dataset.

**Temperature feature generation**  Data from experiments where temperature is missing the temperature is set by baseline temperature in the testing chamber which is 24° Celsius.

**RH**  In data from experiments where RH is missing the RH is generated as a function of temperature. We also change the attribute from integer to real, and the value is rounded off for the conversion, this by adding 0.5, and then casting it to an integer.

$$RH = \text{int}(-1.0772 \times \text{temp} + 71.888 + 0.5) \quad (3.1)$$

**NO$_2$**  Data from experiments where NO$_2$ is missing we set the NO$_2$ value to the NO$_2$ sensor baseline.

**3.1.4 Labels for Classification**

We use the information from the chapter about limits to human exposure to define the labels or classes for the classification. Table 3.5 shows the threshold limit values separated in safe, dangerous and critical. Table 3.6 shows the range of PPM represented in our data.
Table 3.5: The limits define what amount of each chemical that are safe, dangerous (TWA) and what is critical (STEL). With this, we can represent amounts to define classes in our population.

<table>
<thead>
<tr>
<th>Substance</th>
<th>Safe</th>
<th>Dangerous</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formaldehyde</td>
<td>&lt; 0.0378 ppm</td>
<td>0.0378 &gt; 0.0756 ppm</td>
<td>&lt; 0.378 ppm</td>
</tr>
<tr>
<td>Carbon monoxide</td>
<td>&lt; 8.11 ppm</td>
<td>8.11 ppm &gt; 20.3 ppm</td>
<td>&lt; 20.3 ppm</td>
</tr>
<tr>
<td>Nitrogen Dioxide</td>
<td>&lt; 0.0197 ppm</td>
<td>0.0197 ppm &gt; 0.148 ppm</td>
<td>&lt; 0.148 ppm</td>
</tr>
</tbody>
</table>

Table 3.6: By looking at the data we can confirm that the range of possessing data is within the amounts of gases we are investigating, the table shows the range of PPM we possess for each substance.

<table>
<thead>
<tr>
<th>Substances</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formaldehyde</td>
<td>[0, 4] PPM</td>
</tr>
<tr>
<td>Carbon monoxide</td>
<td>[0, 70] PPM</td>
</tr>
<tr>
<td>Nitrogen Dioxide</td>
<td>[0, 4] PPM</td>
</tr>
</tbody>
</table>

A class label represents the data. The class label function is to separate between dangerous amounts of chemicals, and also identify the compound of threat. When structuring data with class labels, we represent all features in one sample with one single class label, instead of measuring each individual sensor. Doing so lets the model that predicts the class label, and evaluates the data take inferring inputs as cross-sensitivity and temperature in consideration. To define the classes, we use a binary approach to describe classes. This lets us make a structure on how to identify the different classes and at the same time makes it easy to label samples. We represent each possible outcome with a binary number and use a bitwise OR operator to combine all possible outcomes into one label. The advantage with this method of labeling is that is generalized to whatever big array of we want to label and how detailed the classifications should be. In the first example of a class representations, we have $2^3(8)$ class population, shown in table 3.7. With a three bit number we can denote every combination of outcome where we distinguish safe from dangerous, 0 represents safe 1 accounts for a dangerous amount. The second example demonstrates how we represent data which can separate between safe, dangerous and critical for all possible outcomes for each sensor, giving a population of $3^3(27)$ classes, shown in table 3.8.
Table 3.7: Class representation for an eight-class solution. In this class representation we separate safe environment from environments within the threshold limit values TWA, for each sensor, and combinations.

<table>
<thead>
<tr>
<th>Class ID as integer</th>
<th>$CH_2O$</th>
<th>CO</th>
<th>$NO_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 3.8: Class representation for a twenty-seven-class solution. In this class representation, we separate and identify between safe, dangerous and critical amounts, for all three sensors.

<table>
<thead>
<tr>
<th>Class ID as integer</th>
<th>(CH_2O)</th>
<th>CO</th>
<th>(NO_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>01</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>01</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>24</td>
<td>01</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>01</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>01</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>32</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>36</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Label generation

In the lab experiments we note down the reference sensor value and the corresponding voltage output from our system for every chemical sensor. With these corresponding measurements we were able to make independent models for each individual sensor. The fitted model represents what amount of gases was present at the current voltage for each sensor. The model was fitted with the differential data where the current baseline was subtracted beforehand. The current baseline is identified by a
individual model made for each sensor. The baseline generation model is made with respect to temperature and humidity. With the differential data from the temperature and humidity generated baselines, we were able to generalize to one voltage to PPM model for each sensor. With this model it was easy to incorporate both climate and chemical experiments at the same time in the same model.

The following equations represent the general labeling model used in three steps, where c represents an estimated constant individual for each model.

\[
\text{Baseline} - s_i = c_1 \times \text{temp}^3 + c_2 \times \text{temp}^2 + c_3 \times \text{temp} + c_4, \quad (3.2)
\]

\[
Dd - s_i = \text{currentvoltage} - s_i - \text{baseline} - s_i, \quad (3.3)
\]

\[
\text{label} - s_i = (Dd - s_i - c)/c, \quad (3.4)
\]

where:

- \( Dd \) is the differential data.
- \( s_i \) is sensor \( i \).

### 3.2 Development Environment

Moving on from the explanation of the data, this next section intends to briefly inform of the development environment used to implement the algorithms.

#### 3.2.1 Python

Python is an efficient prototyping framework that delivers the possibility to use well-documented library’s. These library’s are known to offer various kinds of data processing techniques that are efficient and modular. They also give us freedom to customize each algorithm in detail. Scikit-learn, Numpy, SciPy, and PyBrain are the library’s that is used for testing and developing algorithms.

**Scikit-learn**

Scikit-learn is an open source library that consists of multiple machine learning algorithms for python. It is built on NumPy, SciPy and matplotlib. Scikit learn offers the whole range of tools needed to do rapidly prototype and deploy machine learning algorithms. With tools
for classification, regression, clustering, feature extraction, dimensionality reduction, model selection, and preprocessing. Most of the algorithms that is reported on in this thesis are from the scikit-learn library. This is the SVM, K-means, AdaBoost, PCA, K-nn, linear models, and the RFC. Other tools from scikit-learn that has been used are model selection tools, statistical tools, and preprocessing tools. Some of these are the cross-validation, confusion matrix, scaling algorithms, and accuracy scores.

SciPy and NumPy

SciPy and NumPy are other library’s used to process, format and do statistical analysis on data.

3.2.2 Rapidminer

Rapidminer is a tool used to rapidly display data, test algorithms, and do data processing. The tests for the ANN was done in Rapidminer’s Neural Net algorithm which is a configurable feed-forward neural network trained by backpropagation. The ANN was not developed in Python because the performance of the Rapidminer version was much higher than the Python library PyBrain.

3.2.3 Linear Regression Models in R

The linear model is a linear regression for modeling the regression between the dependent variable and one or more explanatory variables. Through this thesis R’s lm () function has been used to fit an ordinary linear regression model.

3.3 Approach

This section will present the approach to identify the optimal model for the issue at hand. The investigation to determine the most suitable combinations of feature extraction, scaling and machine learning. The presentation of the experiments from this approach is presented in chapter 4. The stepwise method for finding an optimal solution are:

1. Evaluate the data population. What class labels is best representing the data.
2. Qualitatively evaluate the feature extraction methods to investigate further.

3. Qualitatively evaluate the scaling methods to investigate further.

4. Evaluate the performance of different machine learning algorithms to investigate further.

5. Evaluate combinations of the selected feature extraction, scaling, and machine learning algorithms, and iteratively optimize hyper-parameters.

6. Further, optimize the best performing models from point 5.

7. Find a result for the best performing method for machine learning, feature extraction and scaling.
Chapter 4

Experiments and Results

This chapter intends to present results of the experiments that were done to make a model that classifies safe and dangerous environments and which of the gases are present, for the three relevant gases, $NO_2$, $CO$, and $CH_2O$. First, we will go through how the data is represented. Then we will go through results from experimentation, and lastly the final experiments.

4.1 Experimental Data

This section presents an evaluation of the population of different class representations, and also feature extraction.

4.1.1 Population Evaluation

In chapter 3.1.4, we displayed two different ways to define our populations. The only difference between these two populations where that one separated between TWA, and STEL, and the other separated between TWA, STEL, and CL. The twenty-seven class model describes how dangerous an observed amount of chemical is, in somewhat more detail. While the eight-class only separates safe environments from the dangerous. There is no binary or three class representation exclusively represent only safe, critical or dangerous environments. This because we also want to identify the chemical of threat.

For the eight-class population, we get a good spread with seven out of the eight classes, as shown in figure 4.1. Data with class label 7 is almost not represented in the data, and the classifiers will struggle to classify this class. As for the evaluation of the 27 class labeled data, the spread in the populations is not sufficient to represent all classes shown in figure 4.2.
Figure 4.1: Figure showing the distribution for each class in the eight-class population.

Only 11 out of the 27 classes had a sufficient amount of its belonging data points. With that, we are not able to describe the data in more detail than the eight-class population. Thus we will not focus on the evaluation of the 27 class population.

Figure 4.2: Figure showing the distribution for each class in the twenty-seven class population.
4.1.2 Feature Extraction

This section presents two different ways we do feature extraction on the data, and the third presents raw data. These models are to either quantify noise, or extract features with the most information. The three different data setup methods will be evaluated in combination with various scaling and algorithms. Figure 4.3 shows the data represented by three principal components.

![Figure 4.3](image)

Figure 4.3: Figure showing the collected data represented by three principal components. The color represents the label of the class for each data point.

The data is in a time series format, where T is 83378 for the moving average data, and 83397 for the raw data and the PCA data. All features looks as follows for each feature extraction:

- **Time series:** \( D = \{D_t : t \in T\} \)
- **Raw data:** \( D_t = \{RH_t, Temp_t, NO_2_t, CO_t, CH_2O_t\} \)
- **Moving average:** \( D_t = \{RH_t, Temp_t, NO_2_t, CO_t, CH_2O_t\} \)
- **PCA:** \( D_t = \{PC1, PC2, PC3\} \)
Raw Data

With raw data, we have the benefit of no overhead in feature extraction, and at the same time, we can ensure that we preserve all the information in the data. The raw data will look like figure 4.4, where we can observe an experiment with exposure to $NO_2$. As shown in the figure, some features have more noise than others, and therefore it might be useful to process the data on beforehand.

![Figure 4.4: Data from an $NO_2$ experiment are demonstrating cross-sensitivity for all sensors, we can observe the reaction starting at ~270 samples, and ends at ~2000, where all sensors go back to baseline. The red, yellow, green, turquoise, and navy represents $CH_2O$, CO, $NO_2$, RH, and temperature, respectively.]

PCA

We use the PCA to reduce the dimensions of our dataset. We will here present the results on how our algorithms perform when we reduce the data down into it is basic components, stripping away any unnecessary parts. We find the five eigenvectors of the dataset seen in table 4.1 and the eigenvalues, seen in table 4.2. From the table of eigenvalues we observe how well the principal components describe the dataset, or how large the variance is for each principal component. In figure 4.5 we observe how the principal components cumulative variance from the eigenvectors describes the dataset. PC 4 and PC 5 does not seem to supply much information about our dataset. Thus we will discard PC 4 and PC 5 and transform our dataset into the space of PC 1, PC 2, and PC 3.
Table 4.1: This table describes the eigenvectors for each features with each principal component.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>PC 1</th>
<th>PC 2</th>
<th>PC 3</th>
<th>PC 4</th>
<th>PC 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp</td>
<td>0.013</td>
<td>-0.005</td>
<td>0.015</td>
<td>0.296</td>
<td>-0.955</td>
</tr>
<tr>
<td>RH</td>
<td>-0.011</td>
<td>0.006</td>
<td>0.024</td>
<td>-0.955</td>
<td>-0.296</td>
</tr>
<tr>
<td>NO₂</td>
<td>0.007</td>
<td>0.012</td>
<td>-1.000</td>
<td>-0.018</td>
<td>-0.021</td>
</tr>
<tr>
<td>CO</td>
<td>0.544</td>
<td>-0.839</td>
<td>-0.006</td>
<td>-0.014</td>
<td>0.007</td>
</tr>
<tr>
<td>CH₂O</td>
<td>0.839</td>
<td>0.544</td>
<td>0.012</td>
<td>-0.008</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Table 4.2: This table describes the eigenvalues for each principal component. S-dev is the standard deviation.

<table>
<thead>
<tr>
<th>Component</th>
<th>S-dev</th>
<th>Proportion of Variance</th>
<th>Cumulative Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC 1</td>
<td>250.365</td>
<td>0.551</td>
<td>0.551</td>
</tr>
<tr>
<td>PC 2</td>
<td>222.776</td>
<td>0.436</td>
<td>0.987</td>
</tr>
<tr>
<td>PC 3</td>
<td>36.439</td>
<td>0.012</td>
<td>0.998</td>
</tr>
<tr>
<td>PC 4</td>
<td>11.518</td>
<td>0.001</td>
<td>1.0</td>
</tr>
<tr>
<td>PC 5</td>
<td>7.217</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Figure 4.5: As illustrated, the cumulative variance shows that most of the variance lies in PC 1 and PC 2. Where the x-axis represents each principal component, and the y-axis represents the cumulative proportion of variance.
Moving Average

Transforming the data with moving average gives us data with less noise as shown in figure 4.6. We, therefore, want to evaluate how the algorithms that already have proven to perform well with data transformed with by a moving average. After applying moving average with a window of 20 samples, we get data on the same time series form as earlier. Now the overall population classes are somewhat different from the original eight-class population. This is because extreme values will be adjusted towards its surrounding values.

Figure 4.6: Figure showing the moving average filtering on data from the NO\textsubscript{2} sensor, the red dots are averages over fifty blue dots.

4.2 Experimentation and Early Results

This section intends to inform on various choices that led to the final algorithmic approach.

4.2.1 Classifiers

Approaching the issue of finding what algorithms to investigate further we start with a combination of a breadth exploration of algorithms with diverse characteristics and what has been demonstrated to perform well in previous research. The evaluated algorithms are K-nn, K-means, linear regression model, AdaBoost, RFC, ANN, and SVM. Criteria to considered
as important at this stage is performance. Criteria to considered as important later on are both run-time complexity and performance.

We start off by iteratively testing each algorithm, five to ten times each to adjust the hyper-parameters, and experiment with input scaling. All algorithms starting with recommended default values for all parameters, except for the ANN where the initial network had two hidden layers containing twelve, and eight neurons each. When the algorithms show a performance converging towards a better accuracy, we stopped the experiments. Their performance is shown in table 4.3. With these results we can see that four of the algorithms performing well, this the ANN, SVM, AdaBoost and RFC. AdaBoost and RFC are both ensemble learners made with decision trees. Since they are very much alike and RFC scored significantly better than AdaBoost we chose to only take on RFC in further experiments. The algorithms we will continue to investigate are the ones we walked through in the background and research chapter, the ANN, the SVM, and the RFC.

Table 4.3: The table shows results for early iterative tests with each algorithm, to identify what algorithms to use in the final tests.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-nn</td>
<td>0.21</td>
</tr>
<tr>
<td>K-means</td>
<td>0.25</td>
</tr>
<tr>
<td>Linear regression</td>
<td>0.17</td>
</tr>
<tr>
<td>AdaBoost</td>
<td>0.72</td>
</tr>
<tr>
<td>RFC</td>
<td>0.94</td>
</tr>
<tr>
<td>ANN</td>
<td>0.62</td>
</tr>
<tr>
<td>SVM</td>
<td>0.87</td>
</tr>
</tbody>
</table>

4.2.2 Preprocessing

The same goes for how we select methods for preprocessing of the data. In some cases, it is useful to combine different feature extraction and scaling, with different algorithms. Further in this section, we present what is evaluated and how they are applied in combination with the three remaining algorithms. Criteria to consider as crucial at this stage is, increased performance for the classifier. Since our population doesn’t follow the Gaussian distribution we can exclude the global normalization methods in our scaling. We do not want the population to be weighted in respect to a frequency of occurrence.

For scaling we evaluate:
• Differential baseline manipulation
• Uniform scaling
• PN-unit scaling
• Raw data

For feature extraction we evaluate:
• PCA
• Moving average
• Raw data

4.2.3 Results

Table 4.4 shows all the experiments that were done after the qualification of the classifiers. Here each classifier is combined with different scaling and feature extraction techniques. All the results are found by using k-fold cross-validation with five validation sets.

4.2.4 Analyzing Early Results

When observing table 4.4, it becomes clear that scaling of the data has a high impact on the ANN and SVM, but the RFC is not affected at any significance. The random forest is built on decision trees that don’t rely on the distance between feature vectors, hence feature scaling can be discarded for RFC. This also reflects on RFC performance with all different feature extraction.

The raw data and moving average data demonstrates a very similar performance overall, but the moving average data performed best. The main reason for this is that with moving average, noise from the signal from each sensor is reduced, which gives us a more stable signal and most outliers are diminished. The overall lowest performance is with PCA, this probably due to lost information in the dimensionality reduction. At the same time, we are only handling five dimensional data, the PCA don’t reduce much complexity with only a two dimensional reduction. PCA will probably be more relevant to use in a higher dimensional array of sensors. Consequently, we will discard PCA in the final experiments.

Regarding the classifiers, there was some difference in the performance for all experiments. The mean accuracy of the RFC is significantly higher than the other classifiers. The SVM had the lowest performance for its best classifier. This is most likely due to its weakness to perform well on big
Table 4.4: Table showing the results after the optimizing of hyperparameters, and selection of the best performing combinations of feature extraction, scaling, and classifier.

<table>
<thead>
<tr>
<th>Feature extraction</th>
<th>Classifier</th>
<th>Scaling</th>
<th>Accuracy in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw data</td>
<td>RFC</td>
<td>Raw data</td>
<td>95.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>95.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>95.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential</td>
<td>95.30</td>
</tr>
<tr>
<td>Raw data</td>
<td>ANN</td>
<td>Raw data</td>
<td>73.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>90.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>95.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential</td>
<td>70.0</td>
</tr>
<tr>
<td>Raw data</td>
<td>SVM</td>
<td>Raw data</td>
<td>83.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>89.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uniform</td>
<td>80.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential</td>
<td>87.39</td>
</tr>
<tr>
<td>Moving average</td>
<td>RFC</td>
<td>Raw data</td>
<td>95.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>95.72</td>
</tr>
<tr>
<td>Moving average</td>
<td>ANN</td>
<td>RAW data</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>95.55</td>
</tr>
<tr>
<td>Moving average</td>
<td>SVM</td>
<td>Raw data</td>
<td>89.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>90.43</td>
</tr>
<tr>
<td>PCA</td>
<td>RFC</td>
<td>Raw data</td>
<td>90.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>90.0</td>
</tr>
<tr>
<td>PCA</td>
<td>ANN</td>
<td>Raw data</td>
<td>73.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>90.18</td>
</tr>
<tr>
<td>PCA</td>
<td>SVM</td>
<td>Raw data</td>
<td>82.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PN-unit</td>
<td>79.21</td>
</tr>
</tbody>
</table>

datasets [35]. The ANN had the overall lowest performance on raw data, although it performed well with feature scaling. Namely a consequence of ANN’s sigmoid activation function demanding a scaled input to perform well.

4.3 Final Experiments and Results

After qualification of algorithms, feature extraction and data scaling we narrowed down what techniques to further investigate. The following experiments presents the performance on the best performing combinations of preprocessing, and the three algorithms RFC, ANN, and SVM. Again all
the results are found by using k-fold cross-validation with five validation sets. Here we use both the two performance measures described earlier for each experiment, Kappa statistics, and precision-recall. First, we will go through which hyper-parameters were used in the final runs, and then present the results.

4.3.1 Algorithmic Tuning and Hyper-Parameters

After we evaluated what algorithms to test in combination with prepossessing the final hyper-parameter optimization is done to optimize the algorithms for the data with the current the feature extraction and scaling. This section informs on which parameters are chosen in the final run-through of each algorithm, all selected exclusively on its accuracy.

ANN

In the evaluation of layers and perceptrons in the ANN, different combinations have a tendency to give us similar accuracy. There is a trade-off here with accuracy against over-fitting and complexity. That being said, the results from early tests shows that we can get an accuracy up to the same level as the two-layer ANNs performed with one large hidden layer. This is a perfect example of an over-fitted model. Through the evaluation of layers and perceptrons we continue the final tests with a network contained two hidden layers, with 15 and 7 neurons in each layer, and an output layer of 5 neurons.

With all biases in the network, the output, and the hidden layer we can represent the amount of learning parameters for our network as:

\[
Weights = [5 \times 15] + [15 \times 7] + [7 \times 5] \\
Biases = 15 + 7 + 5
\] (4.1)

This giving 215 weights and 27 biases, in total we have 242 learnable parameters. All layers using the sigmoid activation function. With a learning rate at 0.3 the momentum at 0.2 and 500 training cycles. The input layer is not counted as a part of the learning parameters.

SVM

For the SVM we are using the following parameters:

- The penalty parameter C is 1
The kernel function is the radial basis function

\[ K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \]  

The kernel coefficient gamma is 1/5

**RFC**

The RFC shows to perform best with the Gini function for measure the quality of a split and with the max features as the square root of features.

### 4.3.2 Final Experiments

The final experiments is displayed in table 4.5, where the final combinations of feature extraction, scaling, and classifier are presented with its accuracy and Kappa statistics. Note that the accuracy and Kappa statistics is the average accuracy of the combination of all eight classes. The average accuracy for each class is the average from every run in the cross-validation. All experiments were done twice to increase reliability.

Table 4.5: Table showing the results after the optimizing of hyper-parameters and selection of the best performing combinations of feature extraction, preprocessing, scaling, and classifier.

<table>
<thead>
<tr>
<th>Feature extraction</th>
<th>Scaling</th>
<th>Classifier</th>
<th>Kappa in %</th>
<th>Accuracy in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw data</td>
<td>-</td>
<td>RFC</td>
<td>93.2</td>
<td>96.12</td>
</tr>
<tr>
<td></td>
<td>PN-unit</td>
<td>ANN</td>
<td>92.3</td>
<td>95.75</td>
</tr>
<tr>
<td></td>
<td>PN-unit</td>
<td>SVM</td>
<td>82.6</td>
<td>90.17</td>
</tr>
<tr>
<td>Moving average</td>
<td>-</td>
<td>RFC</td>
<td>91.1</td>
<td>95.72</td>
</tr>
<tr>
<td></td>
<td>PN-unit</td>
<td>ANN</td>
<td>94.3</td>
<td>95.75</td>
</tr>
<tr>
<td></td>
<td>PN-unit</td>
<td>SVM</td>
<td>81.2</td>
<td>90.69</td>
</tr>
</tbody>
</table>

Comparing the best performing classifiers in respect to execution-time is done by measuring the time the classifier uses to classify unseen data. We use data from a cigarette experiment with 864 samples. Execution time is displayed in a boxplot shown in figure 4.7.
4.4 Analyzing Final Results

The results from the final experiments reflect the highest achieved results for this project. It is also important to remember that the solution should be a method that is supposed to run on a microcontroller; therefore we also have to include execution time for the best performing classifiers in our final evaluation.

![Boxplot of execution time](image)

Figure 4.7: The box plots comparing the execution time for 864 samples. The three final algorithms are executed 6 times each on an identical computer.

![ANN with moving average feature extraction, and PN-unit scaling](image)

Figure 4.8: ANN with moving average feature extraction, and PN-unit scaling with an accuracy on 95.75 and Kappa at 94.3%.
Figure 4.9: SVM with PN-unit preprocessing with an accuracy on 90.17 and Kappa at 82.6 %.

|        | true 4.0 | true 0.0 | true 1.0 | true 2.0 | true 6.0 | true 5.0 | true 3.0 | true 7.0 | class pr...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pred. 4.0</td>
<td>18060</td>
<td>4578</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>62</td>
<td>0</td>
<td>1</td>
<td>80.08%</td>
</tr>
<tr>
<td>pred. 0.0</td>
<td>1407</td>
<td>46740</td>
<td>363</td>
<td>19</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>96.19%</td>
</tr>
<tr>
<td>pred. 1.0</td>
<td>3</td>
<td>106</td>
<td>5447</td>
<td>0</td>
<td>6</td>
<td>1014</td>
<td>12</td>
<td>0</td>
<td>82.76%</td>
</tr>
<tr>
<td>pred. 2.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3153</td>
<td>238</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>92.25%</td>
</tr>
<tr>
<td>pred. 6.0</td>
<td>45</td>
<td>6</td>
<td>0</td>
<td>226</td>
<td>873</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>75.65%</td>
</tr>
<tr>
<td>pred. 5.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.00%</td>
</tr>
<tr>
<td>pred. 3.0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>312</td>
<td>0</td>
<td>92.40%</td>
</tr>
<tr>
<td>pred. 7.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

class recall 92.49% 90.88% 93.17% 92.79% 78.23% 0.46% 88.89% 0.00%

Figure 4.10: RFC with PN-unit preprocessing with an accuracy on 96.12 and Kappa at 93.2 %.

|        | true 4.0 | true 0.0 | true 1.0 | true 2.0 | true 6.0 | true 5.0 | true 3.0 | true 7.0 | class pr...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pred. 4.0</td>
<td>18569</td>
<td>1995</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>92.97%</td>
</tr>
<tr>
<td>pred. 0.0</td>
<td>1511</td>
<td>50105</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97.06%</td>
</tr>
<tr>
<td>pred. 1.0</td>
<td>0</td>
<td>0</td>
<td>5812</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>92</td>
<td>0</td>
<td>98.54%</td>
</tr>
<tr>
<td>pred. 2.0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3315</td>
<td>97</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97.10%</td>
</tr>
<tr>
<td>pred. 6.0</td>
<td>14</td>
<td>1</td>
<td>0</td>
<td>87</td>
<td>1008</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90.81%</td>
</tr>
<tr>
<td>pred. 5.0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>6</td>
<td>1018</td>
<td>0</td>
<td>0</td>
<td>96.86%</td>
</tr>
<tr>
<td>pred. 3.0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>329</td>
<td>0</td>
<td>97.92%</td>
</tr>
<tr>
<td>pred. 7.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>85.71%</td>
</tr>
</tbody>
</table>

class recall 92.43% 97.29% 99.32% 97.19% 90.40% 94.17% 96.21% 100.00%

The previous figures 4.8, 4.9, and 4.10 are the complete confusion matrix for the best performing models for each algorithm. We can here observe that there are some variations on how each algorithm performs per class. The SVM fails to classify class five, and seven with a 0.46 % accuracy and 0 % accuracy respectively. While the RFC have an good overall score for every class and is the only classifier that manages to classify class seven. The ANN has the highest score on baseline detection, class zero, and also fails to classify class seven. Both the SVM and ANN demands a well represented class to learn the models to classify well since class seven is only represented by five data points these models don’t manage to learn to classify them. This could have been solved by generating data for class seven or duplicate the data points that we already possessed. A decision was made to not do anything with the data representing class
seven because class seven is not originally represented in our dataset, and the five data points that occur for this class are outliers from noise.

The ANN with PN-Uniform and Uniform Scaling and the RFC both with and without scaling gave the best and very similar results. The best result from ANN had an accuracy of 95.75 % with an inter-rater agreement for qualitative items $\kappa$ is at 92.3 %. The best result from RFC was at 96.12 % accuracy, and 93.2% for the inter-rater agreement for qualitative items $\kappa$ at. However, there is not a significant difference in the performance of the RFC in combination with any preprocessing.

From the boxplot in figure 4.7, RFC and ANN shows the best and very similar performance in execution-time, while SVM has a significant lower execution-time, probably a consequence of many support vectors in the SVM and an exhaustive dimensionality transformation to achieve linear separability.
Chapter 5

Conclusion

This chapter concludes this project with a general conclusion and discussion of the results. Lastly, possibilities for further work is described.

5.1 Discussion

The most important parts to question in this thesis is the labeling process. This may have resulted in a bias in the relationship between the labels and the data. This weakness follows as a consequence of the method used to reference the dataset with the exact amount of pollutes present. In the labeling process, generation of class labels is done with a regression model; that may lead to bias in the dataset which tracks to other complications in following processes. However the time series seem consistent over time, and the data points labels do not jump between classes. The labeling is also consistent with the lab notes from the experiments.

We could argue that a linear model would be a better approach to this problem. We already made an algorithm with the linear model to label the samples. This could be an argument for that the linear model algorithmic approach itself is a good approach for this problem. Making an independent linear model for each sensor would provide the opportunity to make a very low complex model. Using a linear regression model, we could at the same time get more exact output on the amount of gases present. However, we can also make the argument that the linear model approach takes one independent sensor in consideration, in comparison with our ML approach, which takes all features from the sensor array in consideration. Although there was only cross-sensitivity to a small extent between the current chemical sensors, this is rarely the case. Thus the linear model approach would not manage to solve the issue with external cross-sensitive factors as alcohols. One of the goals of this project
was to identify models that generalize over sensor arrays. Therefore it will also strengthen the argument from the benefits of using a model that generalizes. Where it is easy to expand to a sensor array with features of a higher magnitude, without any extra overhead.

The data used in the tests are all gathered on the same circuit and same system-on-chip design but generated from a different but supposedly identical boards. Even if the boards hypothetically are identical, there might be a difference in how much the sensors drift over time, and the total resistance in the board design. This may have resulted in a slightly different baseline for the chemical sensors, and may have had an impact on how we labeled the dataset.

Comparison with the state of the art is somewhat difficult. Other electronic noses in this price range are not well documented, and no approach for pattern recognition with the eight-class population on sensor data has been found, thus it is challenging to compare the results to the state of the art technology. The benchmark of the performance of the algorithm is also questionable. Ideally we would need to test the performance for the classifiers on a microcontroller. The execution time on a computer doesn’t say much on how they will perform on a microcontroller, especially since we do not take memory complexity in consideration.

5.2 General Conclusion

This thesis has investigated the possibility of identifying polluted dangerous gas-environments using a low-cost electronic nose. The approach was using machine learning on sensor data provided by the chemiSense board. The three detection gases were \( \text{NO}_2 \), CO, and \( \text{CH}_2\text{O} \). The measurements was done separately for each gas, with mixtures of gases, with cross-sensitive compounds, in variations of humidity and temperature. All measurements were compared to a safe baseline. The majority of the tests were done at the chemiSense lab at UC Berkeley. The goal was to use machine learning combined with cheap sensors to make a low-cost commercial pollution detection system possible.

We investigated a population separated into eight different groups, where each data point was labeled with its associated group as class label. The eight classes are intended for a classifier to identify danger and for what compound. The classifiers are combinations of different feature extraction, scaling technique and a variety of supervised learning techniques from the machine learning domain. By combining these two processes, we developed different classifiers with a variance
in performance. The models performing poorest in the classification accuracy were identified and discarded. This enabled the optimization of fewer models. At the end, we also evaluated the best classifiers based on lowest execution-time. The two classifiers which performed best, are presented as the final results.

Besides the class that was not well represented in the dataset, the model performed worst on classifying noisy datasets. Test mixtures of gases that react on multiple sensors are the most challenging for the system, such as smoke, that besides nitrogen and oxygen, contain a bewildering assortment of toxic gases, such as carbon monoxide, formaldehyde, acrolein, hydrogen cyanide, and nitrogen oxides. These experiments with mixed gases had the lowest accuracy in classification performance, but our model managed to deliver an acceptable classification.

The best models were the RFC, without any preprocessing, and the ANN in combination with PN-unit scaling, and raw data input. With an eight-class representation of the population, our approach manages to classify with up to 96 % accuracy. The RFC delivers a 96 %+ accuracy with every preprocessing technique, and the ANN performs 95 %+. There are some differences in how the classifiers perform across the different classes. Both models perform best on identifying the safe baseline environment. The time complexity results strengthen the conclusion that the RFC and ANN are the best suited algorithms for this problem, where the RFC and the ANN had similar and significant better execution time than the SVM.

This means that the models will classify 96 out of 100 measurements correct. In chemiSense case, the system sample two times per second, over a few seconds of measuring the classification error over multiple samples will be negligible. With such a high sampling rate the 96 % accuracy is adequate to ensure a credible classification. Comparing this to the state of the art electronic noses for pollution ppbRAE with a 3 seconds response time, and the toxiRAE with a 1.5 seconds response time. Further comparing the system with other state of the art electronic noses, we can state that the chemiSense system with a cost of $50, is magnitudes cheaper than both the ppbRAE 3000 and toxiRAE. The chemiSense system is less precise than the state of art systems, but this is not needed. Based on the price reduction alone we can truly state that chemiSense electronic nose offers an edge on the state of the art systems available.

The results confirm that the chemiSense system using these sensors and combined with proper data analysis manages to identify danger levels of toxic gases according to official regulations. Both the RFC and the ANN has proven to perform as good classifiers on this matter.
5.3 Further Work

5.3.1 Automatic Baseline Detection

A self calibration baseline detection would be beneficial. First of all it would by itself understand how the baseline on each sensor behaves in respect to humidity and temperature. It will also calibrate each board that is made, automating parts of calibration after production. Last the Automatic baseline detection can increase lifetime for sensors in respect to long time drift. Approaches to solve this could be applying a particle filter that automatically detects the current baseline.

5.3.2 Long-Term Drifting

Chemical sensors have the weakness that they drift over time. Making a sensing system that lasts for years would demand a better understanding of the sensors long-term drifting. Measurements of long-term drifting takes time, but it might pose the benefit of increasing the lifetime of the sensors. The simple approach here would be to calibrate the baseline as a function of time by detecting the magnitude of the sensor deflection.

5.3.3 Larger Array of Sensors With Embedded Reference Sensors

In order for the system to measure more gases the board, first of all, need to extend its range of sensors. There are a vast amount of gases, and Volatile Organic Compounds that pause a threat to humans at high concentrations. Next step would be to evaluate the already embedded $O_3$ sensor, and add an Ammonia sensor to the array. Since the majority of metal oxide gas sensors and electrochemical gas sensors are cross-sensitive to alcohols it would be interesting to embed an alcohol sensor to the array as well. By adding these sensors there could potentially be a solution for identifying cross-sensitivity, by making a “fingerprint” from the magnitude response of the sensor array. The fingerprint would be a way to identify what gases are present, and reduce the risk of miss interpret data with inferring factors.
5.3.4 Refined Sensor Data

With a larger sensor array there will occur more challenges in the process of collecting reference data, a bigger array would demand a higher number of measurements. As a consequence there will be more cross-sensitivity across the sensor array. By using more accurate reference systems we can automate this process to automatically gather very precise data at the same sampling rate as the sensing system.

5.3.5 Hardware Implementation

This project will be implemented on a microcontroller after this thesis is completed. We have evaluated the run time for each classifier, but not observed its performance implemented on a microcontroller. At this point it will be interesting to see how the algorithm performs in real-time, and what performs best on a minimal clock frequency.
Bibliography


