

Norwegian University
of Life Sciences

Master's Thesis 2019 30 ECTS

Science and Technology
Odd Ivar Lekang

Development of algorithm for pre- processing and prediction in Capacitive Micromachined Ultrasonic Transducers

Julia Førde

Renewable energy and environmental physics
Faculty of Science and Technology

PREFACE

This thesis concludes my master study in renewable energy and environmental physics. The thesis is written for Fluenta in collaboration with the *Eik Idéverksted* at NMBU and the Khuri-Yakub research group at Stanford University.

During my years as a student my interest in technology and especially Machine Learning has grown. In my work, I hope to give a brief introduction to the possibilities and challenges in Machine Learning, as well as demonstrating a helpful method for handling raw data from chemical sensors.

I would like to thank *Eik Idéverksted* for the opportunity to work on real-world applications including the work on chemical sensors that has been used as a foundation for this thesis. It has contributed to practical knowledge and experience adding to my years of study. I would like to thank Odd Ivar Lekang and Ola Ohmberg for feedback during the writing of my thesis. And a special thanks to Kristian Ohmberg for motivation and feedback on my thesis and other work at *Eik Idéverksted* for the past 3 years. Lastly, to my friends for making my student life great with lots of laughs and good memories, and my family for all their love and support.

Ås

May 2019

Julia Førde

ABSTRACT

The World Health Organization states that air pollution leads to 6.4 million premature deaths yearly and the number is increasing. Developing gas sensors that can measure air quality is therefore important. In this thesis, the potential for using a Capacitive Micromachined Ultrasonic Transducers (CMUTs) for air quality monitoring has been evaluated by comparing different technologies and commercial products. Especially as a supplement for more complex and stationary devices the CMUT stands out as a strong contender compared to more established technologies. Compared to other commercial sensors, the CMUT is documented to be more sensitive. In addition, the sensor is small-sized and easy to fabricate. The main challenge for the CMUT is developing selective layers so it can distinguish between different gases with a higher accuracy.

Sensors can generate a vast amount of data. For the public, this information is nothing more than a chaos of numbers. For the CMUT to outcompete other sensors on the commercial market, the sensor must translate the data into comprehensive information answering two questions: 1) which gases are present and 2) at which concentration. As the second part of the thesis an algorithm (Auto-CMUT) was developed to answer these questions. The Auto-CMUT is an automatic system for pre-processing, classifying and quantifying gases in the air based on a Machine Learning approach. Due to lack of data from the CMUT the Auto-CMUT was applied to data from MOX sensors, which share several properties with the CMUT. The results showed that the algorithm performed substantially better on CO measurements than on NO₂. Based on literature and findings when visualizing the dataset, it is likely that this difference is due to a poor selective layer on the MOX rather than the algorithm itself. The algorithm obtained scores as high as the best commercial sensors evaluated in the first part of the thesis with an R²-score of 0.80 for the CO measurements and 0.43 for NO₂. It was also shown that the regression part of the Auto-CMUT increased the R²-score with 0.27 for CO and 0.43 for NO₂.

Obtaining extensive datasets for different real-world applications from CMUT sensors should be prioritized to increase the performance of the algorithm. In the future, the CMUT should be sold developed and sold for specific applications. Checking how chemical degradation affect the sensor over time should also be examined further. Overall, the CMUT technology combined with an automatic system for translating the sensor output seems like a potential competitor on the commercial E-nose market

SAMMENDRAG

Ifølge Verdens Helseorganisasjon fører luftforurensing til 6.4 millioner premature dødsfall hvert år og tallet stiger. Utvikling av sensorer som kan måle luftkvalitet er derfor viktig. I denne oppgaven ble potensialet for å bruke kapasitive mikromaskinerte ultralyd transdusere (CMUT) for og måle luftkvalitet evaluert. I første del av oppgaven ble CMUT teknologien sammenlignet med andre teknologier og kommersielle produkter. CMUT står frem som en sterk konkurrent sammenlignet med mer etablerte teknologier. CMUT har dokumentert høyere sensitivitet enn andre produkter. Sensoren er i tillegg liten i størrelse og enkel å produsere. Hovedutfordringen for CMUT teknologien er utviklingen av selektive lag som kan skille mellom ulike gasser med høy presisjon.

Sensorer generer store mengder data. For folk flest er denne informasjonen ingenting mer enn et kaos av tall. For at CMUT sensoren skal kunne konkurrere på det kommersielle markedet må sensoren svare på to sentrale spørsmål: 1) Hvilke gasser er tilstede i luften, 2) Hva er konsentrasjonen til disse gassene. I andre del av oppgaven ble en algoritme (Auto-CMUT) utviklet for å svare på disse spørsmålene. Auto-CMUT er et automatisk system for pre-prosessering, klassifisering og predikasjon av konsentrasjon av gasser. Grunnet mangel på gode data fra CMUT sensoren ble algoritmen testet på et datasett fra en metal oksidert halvleder sensor (MOX), som har flere likheter med CMUT sensoren. Resultatene viste at algoritmen presterte bedre på CO målinger sammenlignet med NO₂. Litteratur og visualisering av resultater indikerer at denne forskjellen mest sannsynlig skyldes en lavere selektivitet for NO₂ enn CO og ikke algoritmen. Videre viste resultatene at algoritmen oppnådde like gode resultater som de beste kommersielle sensorene evaluert i første del av oppgaven med en R² verdi på 0.80 for CO og 0.43 for NO₂. Det ble også vist at regresjonssteget i algoritmen økte R² verdien med 0.43 for NO₂ og 0.27 for CO sammenlignet med rådata fra sensoren.

For å forbedre sensorens resultater burde det i fremtiden prioriteres og skaffe datasett for ulike bruksområder slik at sensoren kan utvikles og selges for spesifikke bruksområder. Hvordan kjemisk degradering endrer sensoren over tid må også undersøkes nærmere. Hovedkonklusjonen i oppgaven er at CMUT sensoren sammen med et automatisert system for håndtering av rådataen har et potensial på det kommersielle markedet for elektroniske neser.

ABBREVIATIONS

Electronic Nose – E-Nose	SVM -Support Vector Machines
CMUT – Capacitive micro machined ultrasonic transducer	RBF – Radial Basis Function
PZT – Piezoelectric Transducer	AQC – Air Quality Control
FBAR - Film Bulk Acoustic Resonator	Mean Squared Error – MSE
SAW – Surface Acoustic Wave Resonator	Proof of Concept – PoC
QCM – Quartz Crystal Microbalance	NIPH – Norwegian Institute of Public Health
MOX – Metal Oxide Semiconductor	WHO – World Health Organization
PCA- Principal Component Analysis	EPA- United States Environmental Protection Agency
ML – Machine Learning	AQ-SPEC – Air Quality Sensor Performance Evaluation Center
AutoML – Automated Machine Learning	NO ₂ – Nitrogen dioxide
NN- Neural Networks	SO ₂ – Sulphur dioxide
HPO - Hyperparameter Optimization	CO – Carbon monoxide
DL – Deep Learning	CO ₂ – Carbon dioxide
AI – Artificial Intelligence	
LR – Logistic Regression	
OvR – One-versus-Rest	
RF – Random Forest	

TABLE OF CONTENT

PREFACE	I
ABSTRACT	II
SAMMENDRAG	III
ABBREVIATIONS.....	IV
TABLE OF CONTENT	V
1 INTRODUCTION	- 1 -
1.1 PROJECT DETAILS	- 2 -
1.2 THE PROJECT GROUP.....	- 4 -
1.3 AIR QUALITY	- 5 -
2 COMPARISON OF TECHNOLOGIES AND COMMERCIAL SENSORS	- 8 -
2.1 GAS SENSORS: E-NOSES	- 8 -
2.2 USE OF E-NOSES IN AIR MONITORING.....	- 10 -
2.3 GUIDELINES FOR PORTABLE E-NOSES	- 10 -
2.4 SENSOR TECHNOLOGIES	- 13 -
2.5 COMMERCIAL SENSORS	- 18 -
2.6 COMPARING DIFFERENT TECHNOLOGIES	- 22 -
3 MACHINE LEARNING CONCEPTS AND METHODS	- 24 -
3.2 MACHINE LEARNING CONCEPTS	- 26 -
3.3 CLASSIFICATION ALGORITHMS	- 38 -
3.4 ARTIFICIAL NEURAL NETWORKS	- 43 -
4 METHOD.....	- 47 -
4.1 LITERATURE STUDY	- 47 -
4.2 AIR QUALITY DATASET	- 48 -
4.3 SOFTWARE	- 48 -
4.4 PRE-PROCESSING	- 49 -
4.5 PREDICTION	- 50 -
4.6 CLASSIFICATION	- 50 -
4.7 EVALUATION	- 56 -
5 RESULTS	- 57 -
5.1 VISUALIZATION	- 57 -
5.2 PRE-PROCESSING	- 59 -
5.3 CLASSIFICATION	- 60 -
5.4 REGRESSION.....	- 62 -
5.5 RESULT SUMMARY	- 68 -

6	<u>DISCUSSION</u>	- 69 -
6.1	COMPARING CMUT TO OTHER E-NOSE TECHNOLOGIES.....	- 69 -
6.2	COMMERCIAL SENSORS	- 70 -
6.3	DATA SCIENTIST IN A BOX	- 70 -
6.4	LACK OF CMUT DATA	- 71 -
6.5	DATA QUALITY.....	- 72 -
6.6	MULTIPLE CLASSIFIER	- 73 -
6.7	REGRESSION.....	- 74 -
6.8	PRE-TRAINING CMUT FOR DIFFERENT APPLICATIONS	- 74 -
6.9	NO FREE LUNCH THEOREM	- 75 -
6.10	CHALLENGES AND LIMITATIONS	- 76 -
7	<u>CONCLUSION</u>	- 77 -
7.1	RECOMMENDED FUTURE WORK.....	- 77 -
7.2	CONCLUSION SUMMARY.....	- 79 -
	<u>BIBLIOGRAPHY</u>	- 80 -
	<u>APPENDIX</u>	- 88 -
	APPENDIX A: OVERVIEW OF DIFFERENT ACTIVATION FUNCTIONS	- 88 -
	APPENDIX B: RESULTS FROM THE MULTIPLE CLASSIFIER IN THE AUTO-CMUT	- 89 -
	APPENDIX C: RESULTS FROM REGRESSION STEP IN THE AUTO-CMUT	- 94 -
	APPENDIX D: CODE PREPROCESSING AND MULTIPLE CLASSIFIER FROM AUTO-CMUT	- 100 -
	APPENDIX E: CODE FOR REGRESSION FROM AUTO-CMUT	- 106 -

1 INTRODUCTION

In future smart cities, the different sectors such as health, transport, power and surveillance will be connected using technology [1, 2]. An important part of this connectivity will consist of sensors [3, 4]. Sensors can be used for monitoring air quality both in private homes and factories, monitoring food degradation and self-driving cars [5]. These applications are just a few examples of the broad use and field of sensors. WHO has stated air pollution to pose a critical environmental and human health risk at all parts of the world[6]. Improving systems for gas sensing and making them available for a larger part of the population is therefore of great importance.

Throughout history air quality monitoring has been performed by expensive and complex technologies including gas chromatography and infrared absorption as these devices have yielded the best performances and lowest error estimates. Today, a market for using a higher number of sensors for covering bigger areas has emerged. These sensors are intended to be used as an addition to the more complex devices, and do not need the same accuracy. As a result, the size, price, and the number of sensors becomes more important [7]. When it comes to price and size, several Electronic Nose sensors (E-noses) have shown a considerable advantage over the more complex devices.

Within the field of E-noses, there is a wide range of technologies: Quartz Crystal Microbalance, Electrochemical, Thermal Conductivity Detectors and Metal Oxide sensors. This study aims to evaluate the commercial potential for capacitive micromachined ultrasonic transducers (CMUT).

The increasing number of sensors will create vast amounts of data. For most people the amount of data is overwhelming, and the information is often lost in the chaos. In the last decades, machine learning has become the go-to solution when working with big volumes of data [8]. This thesis aims to develop and test an algorithm able to translate raw data from chemical microsensors into gas concentration in the air for the end user by using a Machine Learning approach.

1.1 PROJECT DETAILS

1.1.1 Goals

The goal of the CMUT project is to develop a commercially available air quality sensor based on a machine learning approach. This thesis will consist of two parts:

1: Perform an extensive comparison of different gas sensing technologies and commercial

Electronic-noses: This part aims to evaluate if the CMUT technology can compete with other technologies and sensors on the commercial market. In addition, this section aims to identify a reference framework for testing the Auto-CMUT algorithm developed in part two of the thesis, by gathering results from commercial sensors.

2: Develop an algorithm for handling data from the CMUT sensor: The CMUT will generate big data volumes. The sensor needs software that can handle large data volumes in all phases of the project. To handle big data volumes a machine learning approach is suggested, the algorithm should be capable of:

1. Pre-processing the data
2. Classify the gas type when given a sample
3. Predicting the concentration
4. Evaluate the model against the the commercial sensors found in part one of the thesis.

This algorithm is a “Data Scientist in a box” which will work as an automated system handling raw data by picking the system architecture with highest performance. The automated system will reduce the labour cost in the project along with the number of human induced errors. To keep the computational cost low the thesis focuses on developing several simple models with different parameter values instead of one algorithm with a high degree of complexity. The algorithm developed is further referred to as the Auto-CMUT.

Objectives for first part of thesis:

- I) Evaluate the commercial potential for the CMUT technology.
 - Compare the CMUT technology to other technologies on the market.
 - Perform a comparison between different commercial sensor. The comparison should later be used as a benchmark for the CMUT.

Objectives for second part of thesis:

- II) Develop an algorithm for pre-processing and predicting the concentrations of gases for a CMUT sensor in a real-world application.
 - Implementing some of the methods studied in the literature study using python.
 - Make an algorithm that works as automatic as possible.
 - A Proof of Concept will be performed
 - Identify an air quality data repository that can be applied as an early benchmark test.
 - Test the algorithm on a dataset obtained from a chemical microsensor

Secondary goals:

- Identify and suggest future work in the CMUT project.
- Suggest ways for improving data quality.

1.1.2 Limitations

- Since the samples are collected over a short period of time it's not possible to know how chemical degradation will affect the performance.
- A limited number of samples.
- A limited number of gases is forwarded to the algorithm (CO and NO₂).
- The research on CMUT as a gas monitor is limited.
- The algorithm developed in this thesis is limited to the use of methods for pre-processing, classification and prediction that are already implemented in Python

1.2 THE PROJECT GROUP

The thesis is part of a development project in collaboration with the Khuri-Yakub research group at Stanford University. The group is led by a professor in electrical engineering, Butrus T. Khuri-Yakub. This group work with the development of the CMUT sensor for use in various applications and is meant as an alternative to the conventional piezoelectric transducers.

The project is funded by Fluenta, a company that focuses on sensors for gas measurements. The company was founded in 1985 and is a global leader in sensors used for flare gas measuring with over 75 % of the market [9].

A team at NMBU has for the past 2 years performed field tests, produced several feasibility studies and is currently working to improve and develop algorithms for pre-processing and prediction of the data from the CMUT.

From autumn of 2018, Fraunhofer Institute for Interfacial Process engineering in Stuttgart joined the project group. Fraunhofer has responsibility for coating and testing the second batch of CMUTs in a controlled environment

1.2.1 Earlier work

Master Thesis spring 2018: In January – May 2018 the thesis: “CMUT based chemical sensor for classification and quantification with machine learning in a real-world application” was written by Maureen Byrne. The thesis was based on data from a field-test performed at NMBU in the fall/winter of 2017/2018. Due to drift in data, and high variability in testing conditions this led to a dataset of poor quality which resulted in predictions with high error estimates. Due to the poor quality of this dataset it was not used in this thesis. Instead a dataset from a MOX sensor with higher data quality was used to perform a Proof of Concept on the Auto-CMUT.

1.3 AIR QUALITY

The CMUTs has today sensing layers with selectivity towards SO₂, NO₂, CO₂ and CO. As there is no standard for air quality, suggested guidelines from various sources are presented. An introduction to the different gasses is given. Carbon Dioxide is not a pollutant and is therefore not included in the guidelines.

1.3.1 Air Quality

WHO states that air pollution poses a critical environmental and human health risk. This risk affects everyone in all countries. In 2016 air pollution was estimated to cause 4.2 million premature deaths due to ambient air and 2.2 million due to household air pollution [10]. The risk of experiencing air pollution is highest in low and middle income countries [6]. A Broad selection of publicised work supports the claim of health risks linked to air pollution [6, 10-12]. Despite an increase in deaths related to air pollution, few countries and cities uphold the guidelines for pollution from organizations like WHO [12].

With the increasing degree of air pollution, interest in air quality measurement has also increased. Progress in embedded system and low-cost gas monitors have expanded to the use of microsensors in gas monitoring [13].

1.3.2 Target gases: An introduction

Nitrogen dioxide (NO₂) has a characteristic brown colour and strong odour. Most of the NO₂ appearing in the air is formed as NO and O₃ reacts in the air. The main source for NO₂ is road traffic. Nitrogen dioxide can in some cases convert to nitrate, which is classified as particular matter. The gas is especially harmful to asthmatic patients, children and older people [14].

Sulphur dioxide is colourless, with a recognizable smell. Most of the SO₂ gas originates from the burning of fossil fuels. In western countries in recent years the concentration has declined. For short-term exposure, an increase in respiratory diseases shown. Few studies have focused on the consequences of long-term exposure to SO₂, but a correlation between high exposure and increased mortality has been shown [14].

Carbon monoxide (CO) is a colourless gas that mainly comes from improper combustion of organic matter. Poisoning due to CO causes deaths in many countries, both suicidal and

unintentional [15]. Combustion of oil, gas and coal releases significant amounts of CO. The gas binds itself to haemoglobin in red blood cells and reduces the amount of oxygen carried in the blood [16].

Carbon dioxide (CO₂) is not considered a pollutant but a greenhouse gas [12]. Increasing emissions of CO₂ contribute to global warming. The main source for increasing emission come from the burning of fossil fuels [17]. As global warming has become an vital topic for many people, detecting and measuring CO₂ has become important.

1.3.3 Recommended guidelines for air quality

Around the world, there are different health standards regarding the concentration of pollutants in air. A comparison for different countries is presented in Table 1. The Table shows a high degree of consensus among WHO and the other countries especially on the guidelines for carbon monoxide. On guidelines for sulphur dioxide, only Norway follows the WHO recommendations while China has a limit that is 7.5 times higher.

Table 1: The recommended guidelines for SO₂, NO₂ and CO from various countries and organizations [9, 13-15]

Pollutant	CO		NO ₂		SO ₂	
	Level	Averaging time	Level	Averaging time	Level	Averaging time
Europe	10 mg/m ³ (8.7 ppm)	8 hours	50 µg/m ³ (27 ppb)	24 hours	125 µg/m ³ (48 ppb)	24 hours
USA	(10.3 mg/m ³) 9 ppm	8 hours	(100 µg/m ³) 53 ppb	1 year	(75 µg/m ³) 75 ppb	1 hour
China	4 mg/m ³ (4 ppb)	24 hours	40 µg/m ³ 21 ppb	1 year	150 µg/m ³ (57 ppb)	24 hours
WHO	(7mg /m ³) 7 ppb (indoor)	24 hours	40 µg/m ³ (21 ppb)	1 year	20 µg/m ³ (7.7 ppb)	24 hours
Norway	(10 mg/m ³) 10 ppb	8 hours	40 µg/m ³ (21 ppb)	1 year	20 µg/m ³	24 hours

2 COMPARISON OF TECHNOLOGIES AND COMMERCIAL SENSORS

This section aims compare the CMUT technology to other E-nose technologies on the market. It also explains the working principle for a selection of technologies and how this combined with machine learning can give us an inexpensive microsensor with low error estimates. Lastly a comparison of commercial E- nose devices is conducted to set a benchmark for the CMUT.

2.1 GAS SENSORS: E-NOSES

The first design of an electronic nose sensor was published in 1982. The design combined several chemical sensors together with pattern recognition [18]. In literature, different definitions of E-noses can be found, but one of most well-known definitions comes from Gardner and Bartlett [19]

“An electronic nose is an instrument, which comprises an array of chemical sensors with partial specificity and an appropriate pattern recognition system, capable of recognizing simple or complex odours.”

E-noses are based on the definition an array with multiple sensor. These sensors are selective to different chemical compounds [20]. To obtain information from the sensor output pattern recognition techniques are used. These techniques often include the use of neural networks, [21].

To attract molecules for the air a functionalizing layer is applied on the E-noses, also called sensing layer. Technologies like GC and IR do not use functionalization layers and are therefore not defined as E-noses [2]. By chemical interaction between the air and the functionalization layer, molecules are extracted from the air and onto the sensor. This interaction leads to a shift in output from the sensor, for example change in the electric signal from the sensor. A E-nose typically consists of an array with several sensors with different functionalization layers [22]. The enormous number of known chemical substances gives rise to the challenge of finding layers that are selective to one specific compound. One of the E-noses biggest challenges is that no functionalizing layers are perfectly selective, it will therefore be an overlap of information from different functionalization layers. This is referred to as cross-selectivity [23].

Due to cross-selectivity predicting the concentration of a specific gas is a complex task. By using pattern recognition techniques, an algorithm is trained to recognize the pattern between the shift in signal to the amount of the gas or gas type, referred to as the target [2]. To learn patterns in data the algorithm is dependent on having the true concentration so the algorithm can learn from its mistakes. The true concentration is in Machine Learning referred to as the target value. To provide the target sensors with high accuracy, reference sensors are used. The working principle of E-noses is illustrated in Figure 1.

E-noses have a wide range of applications: monitoring freshness of food [24, 25], medical diagnostics [26, 27], agriculture [28] and air quality monitoring [29]. In this thesis, the focus is on E-nose for air quality measurements.

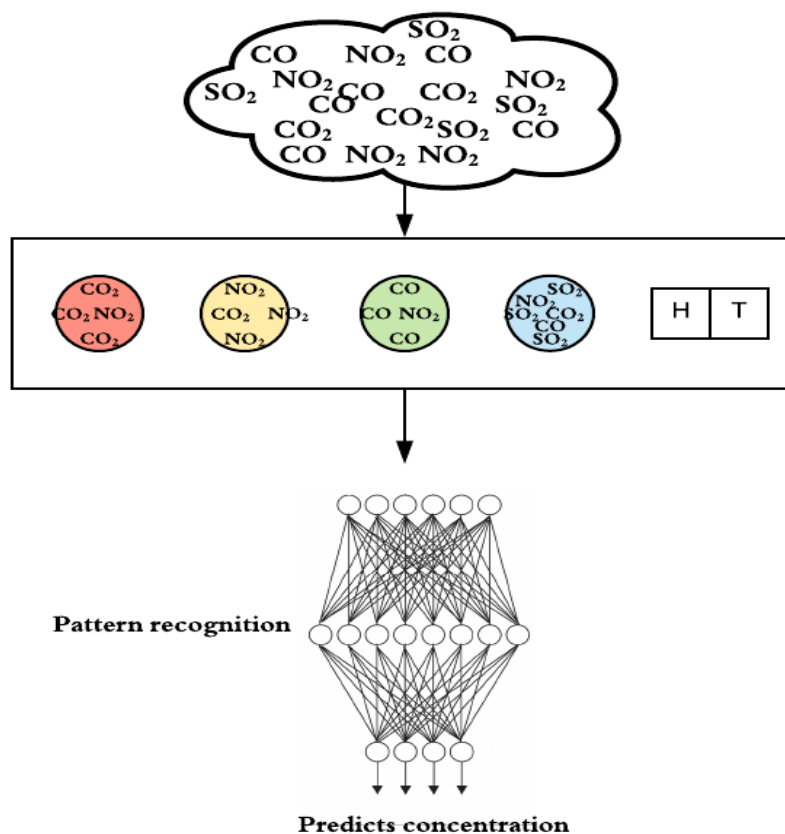


Figure 1: Working principle of E-noses. Different colours refer to different sensing layers. The problem with cross selectivity is illustrated. E-noses usually measure temperature and humidity which is illustrated with H and T. Raw data from the sensing layers are forwarded into Neural Networks.

2.2 USE OF E-NOSES IN AIR MONITORING

Historically, air quality monitoring has been done by government authorities and experts. The methods used for monitoring is dominated by governmental approved instruments, these instruments are often stationary due to their size. In addition to being heavy they are in the price range between €5000 and €30 000 per device and are dependent on frequent calibrations. Since the stations are not portable the area covered by them is limited-. Often only big cities with good economy have stations at all [30].

Currently there is a trend to increase the volume of measurements regarding air quality. To acquire these volumes of data the need for more low-cost, simple to use and portable sensors is necessary [7, 30, 31]. Portable E-noses introduces an opportunity to obtain measurements from a network of sensors, that can be distributed to cover larger geographical areas(parks, cities and even countries) [29]. It should be noted that the E-nose technology has challenges with obtaining data of high quality so that meaningful information can be obtained from these sensors [31]. Even though these sensors have not fully been tested, or currently being regulated by standards, the use of them is rapidly increasing. This rapid increase highlights the need for a system that defines expectations and requirements for E-nose used in air monitoring [29]. The European Commission is currently working on setting standards for low-cost sensors, they are positive, but highlight the fact that the biggest challenge for E-noses is the stability of the selective layer [32].

2.3 GUIDELINES FOR PORTABLE E-NOSES

There are no guidelines for portable devices used in air quality monitoring. Due to the increasing market for portable E-noses the European Committee of Normalization on Air Quality has created a working group to define such guidelines. As no guidelines has been published this thesis evaluates technologies and products against the recommended guidelines from the United States Environmental Protection Agency (EPA) [33]. The recommendations are defined in Table 2.

Table 2: Overview of recommended criteria for portable E-noses from EPA

Criteria	Recommended by EPA	
Small-size	< 2 kg	
Low- cost	< \$2500	
Selectivity	Not given	
Sensitivity	Not given	
Detection Limit	CO	0.1 ppm
	SO ₂	10 ppb
	NO ₂	10 ppb
	CO ₂	100 ppm
Error	>20 %	
Data completeness	<80%	
Capable of continuous measurements	Seconds → 5 minutes	

It should be noted that the weight requirement for portable E-noses was set based on the weight of other portable devices, and is not given as a criterion from EPA.

Sensitivity measures how sensitive the sensor is to changes. For sensors measuring frequency shift based on changes in mass the sensitivity says how small changes in mass that lead to a change in frequency.

Selectivity measures how effective a sensor is to distinguish between different gases. Sensing layers with high selectivity would be able to absorb only NO molecules, while a layer with low selectivity would absorb NO, NO₂ and CO measurements.

It should be noted that although a requirement for selectivity and sensitivity is not defined, better selectivity and sensitivity will lead to a lower error. Aiming towards a better selectivity and sensitivity should therefore always be prioritized.

Data completeness is the amount of measurements obtained compared to the expected amount.

Error tells how much a measurement from a E-nose should maximally deviate from the measurements performed by the reference sensors. To measure this the thesis uses the R² score.

Detection limit is the lowest concentration a sensor can measure.

Continuous measurements say how frequent the sensor can take measurements.

2.4 SENSOR TECHNOLOGIES

Currently much work is being done in the field of sensors for gas detection, it should be noted that some of these technologies are currently only used in research. In Figure 2 an overview of different sensor types is shown. In this thesis, the CMUT potential compared to other E-noses is evaluated.

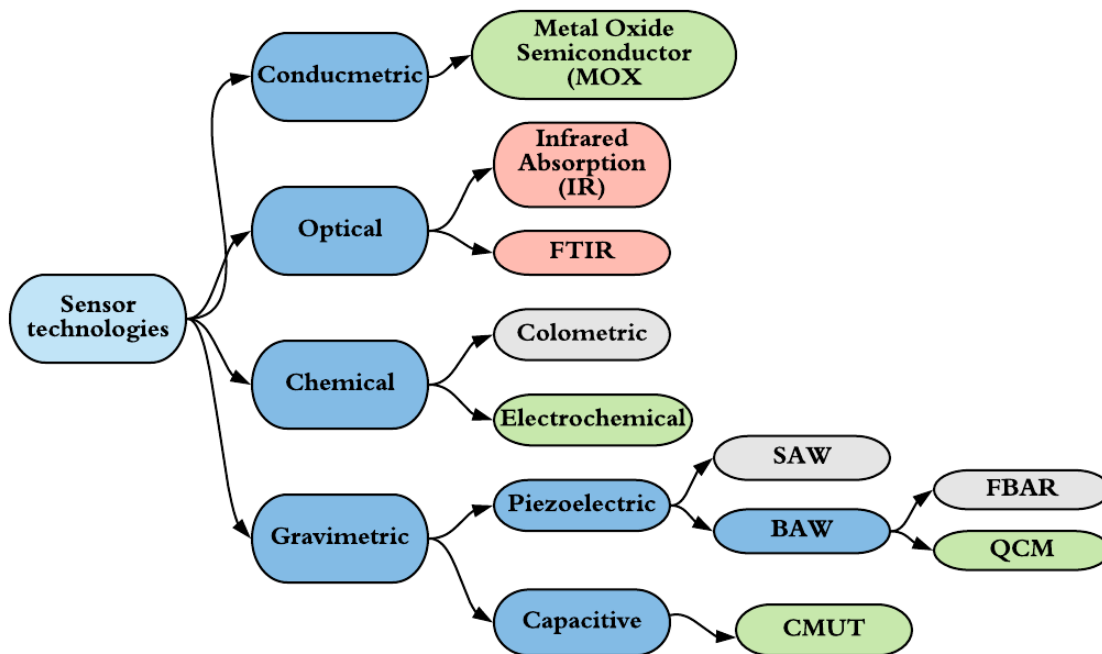


Figure 2: The Table gives an overview over sensor technologies that could be used for gas detection. Conducmetric, chemical and gravimetric sensors are E-noses.

Gravimetric sensors are mostly found in research and academia as few sensors have reached the commercial market [2, 34]. The working principle of the gravimetric sensor is that mass change on the sensor surface leads to a change in some electric property, this change is measured. This is illustrated in Figure 3. To obtain a change in mass a selective layer must be applied to the sensor. Obtaining layers that only are selective to one gas is the biggest challenge for using gravimetric sensors for gas detection. Common factors that make gravimetric sensor well suited for gas detection are: small size, consume little power and are low-cost [35]. There are also big similarities between fabrication for metal oxide semiconductors and gravimetric sensors [2, 36].

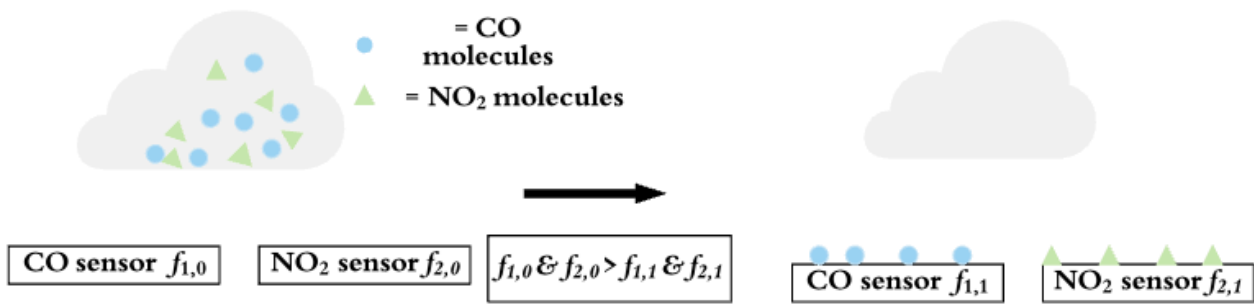


Figure 3: Illustrates the working principle of a gravimetric sensor: The absorption of specific molecules give a higher mass. Higher mass gives a lower frequency.

Optical sensors are traditionally used for application where a high accuracy is needed. Optical sensors have the advantage that they do not need a selective layer. Since this types of sensors are big , highly priced as well as having a long response time, they are not discussed further for application of E-noses [2].

Conductometric sensors measure the electric conductivity. For use in E-nose application they have several of the same advantages as gravimetric sensors: small-size, low-cost, simplicity in fabrication and use. In addition to the possibility to measure a range of gases. For gas sensing the metal oxide semiconductor(MOX) has gotten most attention. [37].

Chemical Sensors produce a change in output signal due to a change in some chemical property. In section 3.7 Electrochemical sensors stand out as the chemical sensor that is mostly used among commercial sensors.

An overview of different technologies is presented in Figure 2. Technologies marked in red do not fulfill the E-nose requirements. The four types marked in green are further explained and presented in the next sections. Sensor types given in grey are not explained in the next sections as they have substantial similarities with at least one technology marked in green.

The MOX and electrochemical sensor are further explained as they frequently appeared among commercial sensors (section 2.5). The QCM is further explained to compare CMUT to other gravimetric sensor.

2.4.1 Metal oxide semiconductor sensor

The sensor consists of a semiconductor and metal oxides as selectivity layers. As the selectivity layer absorbs molecules from the air, the conductivity changes. Depending on the sensing layers a wide range of gases can be detected [38]. The MOX sensor fulfils most of the requirements for the portable E-noses, it is low-cost, small size, simple to both use and fabricate, long lifetime and able to detect a wide range of gases [39]. Due to its characteristics, it is among the most studied group of gas sensors [37]. Main drawbacks of the technology are varying selectivity for different gases. The sensor also experiences a change in chemical and physical properties over time due to oxidation in the sensing layer [40]. The sensors can also respond differently at different temperatures and humidity levels [37].

2.4.2 QCM

Quartz crystal microbalance is traditionally used for precise gravimetical measurements. Figure 4 illustrates the working principle of a QCM.

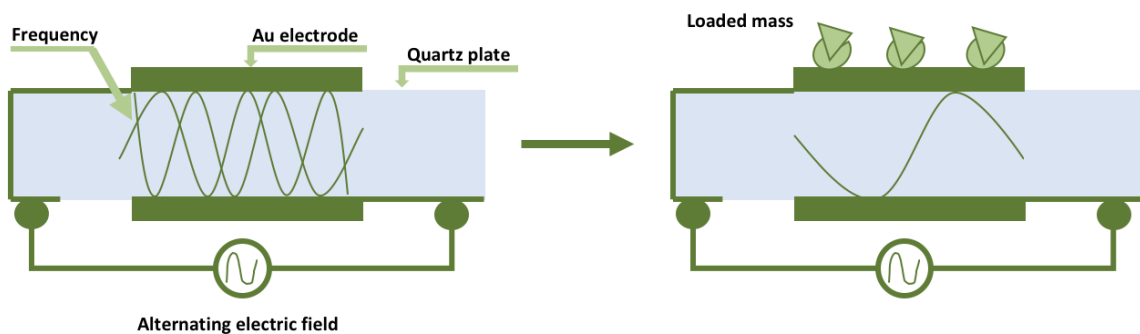


Figure 4: The working principle of the QCM sensor. Made by: Byrne Maureen

The QCM sensors are based on the piezoelectric properties of the quartz crystal. The quartz crystal is placed between two electrodes. By sending an alternating current through the bulk of the sensor, the frequency is measured. As more molecules are attracted to the chemical layer on the sensor the mass and frequency changes. The relationship between change in frequency and mass is proportional and the sensor works in both liquids and air [41]. Drawbacks of the QCM are complicated fabrication process, varying response time and sensitivity for noise due to surface interference [42]. The sensitivity for QCM is substantially lower than the CMUT,

and the fabrication method is more complex. In his work Mølgaard states that the QCM is least suitable for use as a gas detection device among all the gravimetric sensors listed in Figure 2 [2].

2.4.3 CMUT

The first Capacitive Micromachined Ultrasonic Transducer (CMUT) was first presented in 1994 [43]. Originally the transducers were used to transmit or receive ultrasound. Ultrasound is defined as sonic waves with a frequency over 15 kHz. The working principle of CMUTs used as a transmitter is illustrated in Figure 5. The CMUT is mostly used in medical imaging, but in recent years several publications have focused on using CMUT for gas detection [2, 44-46]. Most of the publications for using CMUT for chemical sensing is published by the Khuri-Yakub research group at Stanford University.

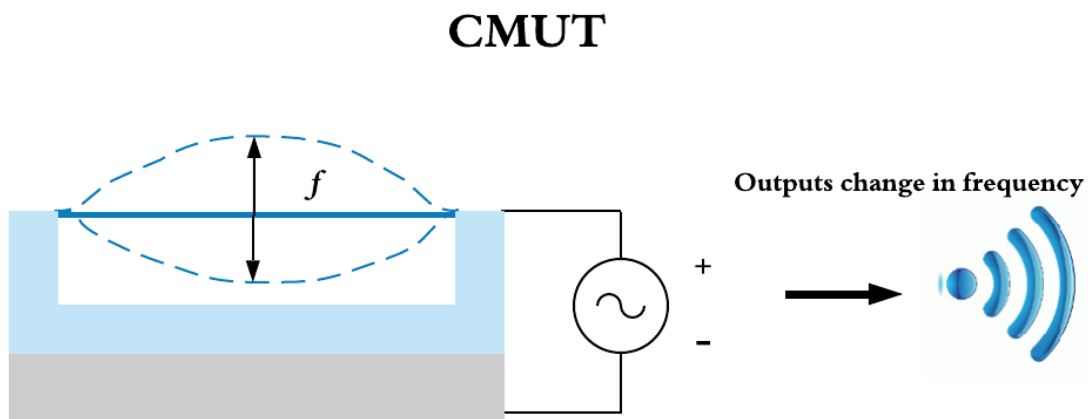


Figure 5: Working principle For CMUT as a transmitter, by applying a voltage between the top electrode and the bottom the plate vibrates with a frequency. Frequency shift is outputted from the sensor. Modified from [47]

The CMUT consists of a conductive plate over a gap of vacuum. To achieve selectivity towards different chemical compounds a chemical coating is applied on top of the plate. By applying a voltage over the plate, it starts to vibrate. When applying a selective layer onto the plate the plate absorbs molecules from the air. As the mass increases the frequency decreases. The CMUT continuously measures the frequency shift [43].

Since the coating of the sensors ideally are selective to one gas, the CMUT sensor should consist of an array of sensors with different coatings. This gives a product that can measure several different gases [48]. CMUTs are easily fabricated into arrays as shown in Figure 6.

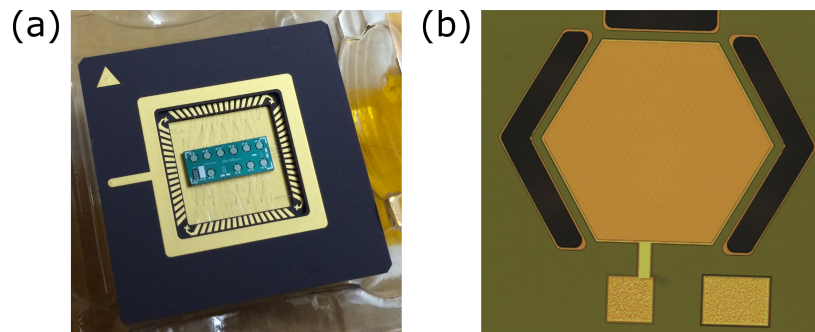


Figure 6: a) complete sensor chip with 9 sensor, illustrates the small-size of the CMUT chip. b) Shows one sensor consisting of 271 plates covered with the same chemical coating.

In Mølgaard's work he presents results showing sensitivities as low as 0.83 Hz/ag. This result is similar with results published by Khuri-Yakub research group of 0.49 Hz/ag [2, 45]. The mass sensitivity of 0.49 Hz/ag is to the knowledge of the author the lowest published result for the sensitivity among CMUTs.

As mentioned about gravimetric sensors the CMUT advantages in addition to excellent sensitivity are: low-cost, easy fabrication and small-size. Advances in obtaining functionalizing layers with higher selectivity will make the sensor capable of measuring more gas types. Like other gravimetric sensors, MOXs and electrochemical sensors the CMUTs biggest challenge is obtaining a higher selectivity for different gasses.

2.5 COMMERCIAL SENSORS

As no universal strategy for testing or other benchmarking for chemical microsensor exist, this section will present sensors tested by EPA and AQ-SPEC. These sensors are tested and evaluated in an objective way and therefore the results are considered more trustworthy. Lastly, a summary of the findings is provided. Sensors evaluated by iScape will not be presented in this work, as they have not performed the testing themselves.

2.5.1 Benchmarking air sensors

During the literature study, no testing system to validate different air sensors was found. As a result, companies can sell sensors for gas and air monitoring without providing any quality assurance. However, there are projects and companies that evaluate sensor to inform the public.

Environmental Protection Agency U.S(EPA): Through research EPA wants to accelerate the development of air monitoring devices that are low-cost, portable and user friendly for the public. With an increasing interest in air quality, a growing number of sensors are made commercially available. Often limited or no information of the performance over longer periods of time in real-world application or lab is provided. EPA performs an objective evaluation and testing of a selection of different air sensors commercially available. Over a longer period with continuously measurements the different sensors are evaluated against measurements from federal used monitors. The results can be found on EPA's website [49].

iScape project: The iScape project is funded by the European Community(EU). The project aims to advance the control of air quality and carbon emissions in European cities. Part of the project is to give scientific guidance to the end user [50]. This is done by checking literature and testing done on low cost sensors in the range 100 - 500\$. Sensors with a price over 100\$, software for data collection/handling is required. The recommendations from iScape are based on testing done by other parties, but the scientific credibility is evaluated by the iScape team [50].

AQ-SPEC: The Air Quality Sensor Performance Evaluation Center aims to perform evaluation and testing of currently available low-cost sensors under laboratory conditions and real-world applications. This research seeks to inform the public about the actual performance of commercially available sensors. The cost must be lower than 2000\$ and provide real-time measurements [51].

Note: Having a sensor that has undergone evaluation from an objective third party as iScape, EPA or AQ-SPEC increases a sensors credibility.

2.5.2 General about EPA and AQ-SPECs testing

EPA: EPA has performed a selection of sensor test trials. The sensors are tested in labs, and if they obtain good results in lab tests, some of the sensors undergo a pilot test in a real-world application. All tests are performed in a collaboration with the sensor developers [52].

Generally power requirements vary widely between different sensors. Collection and storage of data also varied. Some sensors directly output values that easily can be interpreted like concentration while others give information about change in voltage, electricity, conductivity, etc. When obtaining the latter type of output the results are translated into concentration through EPAs own algorithms. It is preferred that the developers provide an algorithm for translating output. When selling commercial products, software should be included so the output is given in concentration [52].

AQ-SPEC: Sensors are first tested in the field for roughly two months. The measurements given by the tested sensor is compared to measurements from a federal reference monitor [51]. Sensors showing certain degrees of performance are further tested in a laboratory. The test subjects the sensor to known gas concentration under controlled temperatures and humidity levels [53].

Linearity: Both EPA and AQ-spec ranks the tested sensors performance with a correlation measure, the R^2 score between the low-cost sensor and a reference sensor. The R^2 score tells how much the output from the tested sensor deviates from the reference value. The max score is 1, higher correlation meaning higher similarity between the tested sensor and the reference [54].

2.5.3 Different commercial sensors

In this section, several sensors tested by AQ-SPEC are presented. In the end a table summarizing the characteristics of each sensor is presented. Only sensors measuring one or several of the gases NO_x , NO_2 , CO , CO_2 and SO_2 are looked at as these are the target gases for the CMUT sensor.

Air Quality Egg Version: The Air Quality Egg take real-time measurements and can be accessed through web, mobile app or manual download. The Egg can measure NO_2 , CO , CO_2 , SO_2 , PM and O_3 . One Egg can measure up to two gases in addition to PM [55]. The Air Quality Egg uses a SGX Sensortech MICS-4514 metal oxide sensor to measure NO_2 , CO , CO_2 , SO_2 [56].

Vaisala AQT410: Electrochemical sensors that can measure NO_2 , CO , SO_2 , H_2S and O_3 . Each sensor can measure up-to four gases. The sensor is intended for stationary use[57]. The sensor does not fulfil the cost requirements for a E-nose, but is presented due to its good results in field.

AQ Mesh version 4.0: Is a stationary sensor system measuring NO_2 , CO , CO_2 , SO_2 , and O_3 . The system uses electrochemical sensors. The evaluation performed by AQ-SPEC was stopped by the developer of AQMesh after the field test [58]. The system does not fulfil the cost requirements for a portable E-nose but obtained good results in field.

CairPol Cairsens: CairPol offer sensors for measuring NO_2 , CO , SO_2 , H_2S , PM and O_3 . Cairpol use electrochemical sensors. One sensor can measures one gas [59].

Unitec-SENS-IT: the Unitec SENS sensor uses MOX sensor for measuring NO_2 , CO and O_3 . Each sensor weighs around 200 grams and a price of around \$2000 [60].

In Table 3 an overview of the results obtained from the AQ-SPEC tests are presented

Table 3: Shows test results obtained by AQ-SPEC for the sensors: Air Quality Egg, CairPol Cairsens, Unitec, Vaisala AQT410 and AQMesh. Used with permission from AQ-SPEC

Sensor	Cost	Size	Type	Linearity	Data Recovery	Response time
Air Quality Egg version 1	\$200	0.2 kg	MOX	CO: 0.0 NO ₂ :0.33-0.40	CO: 100 % NO ₂ :100%	1 min
CairPol Cairsens	\$1243 \$1198	0.06 kg	Electro-Chemical	CO: 0.93-0.94 NO ₂ : 0.05-0.12	CO: 92% NO ₂ : 4.3	1 min
Unitec SENS-IT	\$2200 (O ₃ , NO ₂ and CO included)	0.2 kg	MOX	CO: 0.33-0.43 NO ₂ : 0.59-0.62	CO: 99% NO ₂ : 99%	1 min
Vaisala AQT410 v.1.15	\$3700 (NO ₂ ,SO ₂ , CO, O ₃)	0.69 kg	Electro-Chemical	CO: 0.78-0.80 NO ₂ : 0.43-61	CO: 96% NO ₂ : 96%	1 min
AQMesh c.4.0	\$10 000 (NO, NO ₂ , CO, O ₃ , SO ₂ included)	< 2 kg	Electro-Chemical	CO: 0.41-0.80 NO ₂ : 0.1-0.46	CO:90-100% NO ₂ : 90-100%	1 min

2.6 COMPARING DIFFERENT TECHNOLOGIES

In Table 4 different sensor technologies are ranked against the E-nose requirements presented in section 2.5.

Table 4: Ranks different sensor technologies against requirements set for E-noses: e: excellent, g: good, p: poor, b: bad. Modified with permission from: [37]

	CMUT	MOX	Electro chemical	TCD	Infrared absorption
Small size	e	e	e	b	b
Low-cost	e	e	g	g	b
Lifetime	-	g	p	g	e
Sensitivity	e	p	g	b	e
Selectivity	p	p	P	g	e
Error	-	p	p	g	e
Continuous measurements	e	e	e	g	b

It should be noted that the error criteria for electrochemical and MOX sensors was ranked as poor since the commercial sensors presented in section 2.5 showed high variability in R^2 score between the same gases. Even though the CO in some cases obtain a lower error rate than 20%,

this is not always the case. For NO₂ the commercial E-noses seldom obtain error rates lower than 20%.

The weights of MOX and Electrochemical has some products with higher weight than 2 kg, but due to several products with lower weight it is possible to make E-nose products with lower weight than 2 kg.

Among gas sensors infrared absorption is a device that generally obtains the highest accuracy and is often used for application where the accuracy is highly important along with gas chromatography. However, this device does not fulfil the requirements for E-noses, with a big size, high price and low capability for performing continuous measurements. Both electrochemical and TCD perform well on most points except sensitivity. The MOX and CMUT sensor obtains the highest average ranking over all requirements. However, little information is found on the CMUTS lifetime. Based on literature presented earlier the CMUTs outperforms both the MOX and Electrochemical sensors on sensitivity.

2.6.1 Summary Commercial Sensor

In section 2.6 three different projects testing or evaluation commercial sensors have been presented: the iScape project, EPA's testing of low-cost sensors in laboratory and AQ-SPEC's testing and evaluation. Additionally, some sensors tested by AQ-SPEC were presented along with testing results.

In Table 4 the results for various sensors is provided. The dominating type of sensors tested by AQ-SPEC are Electrochemical and MOX sensors. The sensors gave varying results. AirQuality Egg performed overall poorly on both NO₂ and CO, but is sold at a low price. CairPol performed very well on CO, but useless on NO₂. Unitec performed satisfactory on CO and a little better on NO₂. Both Vaisala and AQMesh performed good, but have a price higher than the recommended price from both EPA and AQ-SPEC

The findings for evaluating a sensors performance showed that companies/organization like EPA and AQ-SPEC ranks sensor against a linearity score [51]. Based on these results the algorithm developed in the second part of the thesis should calculate the R² score. Using the R² score will make it easier to compare the results from the Auto-CMUT against sensors on the commercial market.

3 MACHINE LEARNING CONCEPTS AND METHODS

General concepts in machine learning will be presented to give basic background in the field. The relationship between Artificial Intelligence, Machine Learning and Deep Learning is explained. The section will further focus on pre-processing techniques and methods for classification and regression

3.1.1 What is it?

Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) are among the most frequent buzzwords in technology. The words are often used interchangeably even though it is not quite the same. Easiest explained is that machine learning and deep learning are branches of the broader concept AI. This is illustrated I Figure 7. AI can be defined as a system that can interact with its environment [4].

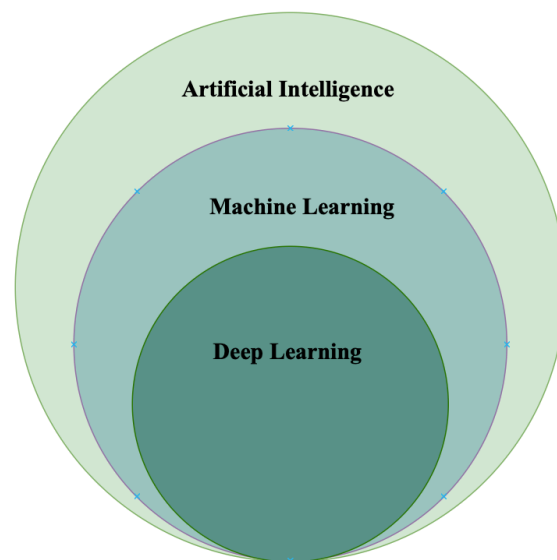


Figure 7: Illustrates the relation between artificial intelligence, machine learning and deep learning.

Machine learning creates systems or algorithms that can improve its predictions through experience, especially when introduced to big volumes of data [8]. ML offers an efficient way for obtaining information from data and discovering patterns in this data. The ML subfield

contains a vast amount of applications: detecting spam, voice recognition, translation, image analysis, predictions from sensor and other electronic devices etc. In coming years the range of application will increase further [61].

Deep learning is a class of algorithms within ML. All these algorithms are based on the basic building block in the human brain, neurons. For decades' researchers have tried to understand the building blocks of intelligence. There is consensus among scientists that the brain generates information based on a complex network of neurons. By electric connection between these neurons information for movement, breathing and thinking is generated [62]. By using neural networks deep learning uses the basic concept of how the neuron in the human brain works. These networks can be taught to classify information and identify more complex patterns [61]. Neural Networks (NN) are further discussed in section 3.4.

3.1.2 Applications for AI

Even though artificial intelligence has been frequently used for decades, it is nothing compared to the amount of applications that has occurred in recent years. The technology is advancing much quicker today than 10 years ago. In addition to this the ordinary guy in the street is now aware that artificial intelligence exists and often has some insight in what it is and can how it be used. Today, people are surrounded by AI: Each time you log into Netflix a bunch of machine learnings algorithms gives suggestions of which movie to play next based on your previous choices and ratings, Siri on iPhones is nothing more than a network of machine learning algorithms [63]. All Google's search engines are based on AI techniques, increased use of ML in medical imagining can detect sick patients quicker than a doctor [64], [65]. These are only a few examples of the AI people use every day.

In the future, the complexity of task assigned to AI will increase. In a decade, maybe your automated car will drive you to your appointment with your doctor, and the doctor is probably a super computer that in few second could analyze symptoms together the results from blood test and blood pressure to predict a diagnosis [66]. In addition, portable biometric sensors will most likely be able to monitor breathing, temperature and blood pressure continuously and give an alarm if there is a high likelihood that a person is about to get sick [67].

3.2 MACHINE LEARNING CONCEPTS

The section will further focus on pre-processing techniques and algorithms for predicting the target gas and concentration. The section will give a theoretical overview of the different algorithms and techniques used in this work. Because of the amount of techniques and method available in the field of machine learning only a selection of methods are applied to the dataset in this thesis.

3.2.1 Supervised and unsupervised learning

In machine learning tasks often get divided into two categories supervised and unsupervised learning, this is illustrated in Figure 8. In supervised learning the desired output of the algorithm is known a priori and is called the target. Supervised learning tries to find a function that best can represent the relationship between features and the target. When features are forwarded the into the algorithm it should be able to predict the corresponding target label or value [61]. For a chemical microsensor, the features would be the output from the sensor while the target is the true answer given from a reference sensor. It is important to point out that the target may be wrong due to noise, inaccurate measurements. Logistic regression, support vector machines, neural networks and random forest are examples of supervised learning techniques [61].

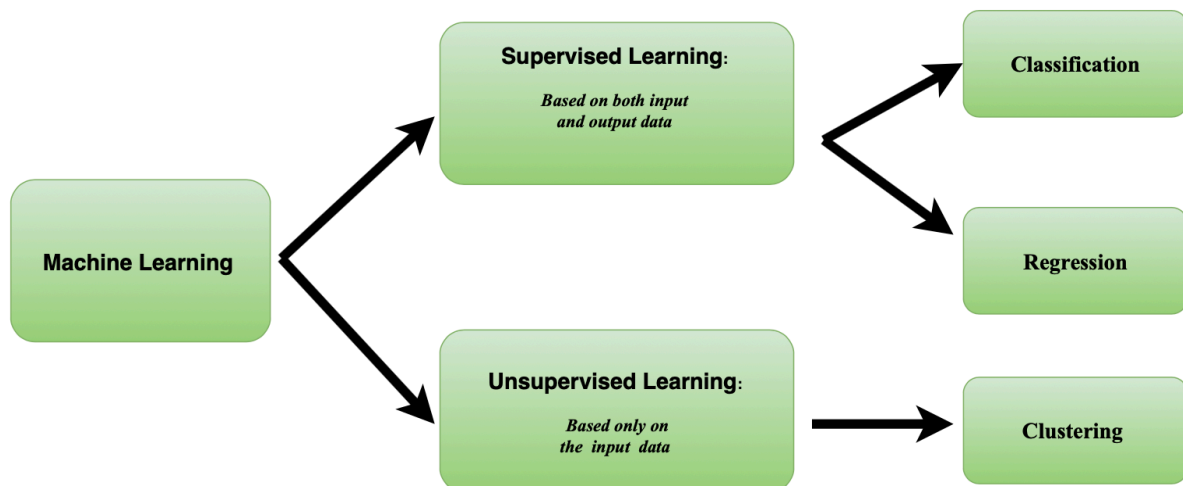


Figure 8: Explains the difference between supervised and unsupervised learning.

Unsupervised learning does not have a known target value or label, the structure of the data is unknown. The goal is to detect the structure in the data and extract meaningful information without depending on a known target value. Usual examples of this is clustering where the goal is to find how a bunch of samples are grouped together, or dimensionality reduction of data [61]. As this thesis is based on supervised learning, unsupervised learning will not be discussed further.

3.2.2 Cost functions

A cost function measures an algorithms ability to estimate the relationship between features and targets. The higher values of the cost function, the higher difference between the algorithms prediction and target. Minimizing the value of the cost function in supervised learning is crucial [61].

The cost function is minimized by adjusting the parameter values (for cost functions referred to as weights). During the learning process the algorithms learns to adjust the weights of the cost function in a way that minimizes the output from the cost function.

3.2.3 Training and testing

The method of training and testing the algorithm is central in supervised learning. The data is normally divided into train and test sets. During the training process, the algorithm tries to detect patterns in the data by predicting the target and then checking it against the true target. If the prediction is correct nothing is done, if the prediction is wrong the algorithm adjusts its weights. When an accepted accuracy is obtained or maximum number of epochs is reached, the algorithm is used on the test data. On the test set the algorithm predicts based on the features. The performance is calculated as the difference between prediction and target. The test score is the first test for how the algorithm will perform on unseen data as the training set normally is seen several times during the training process. The challenge of obtaining a high test score and a low difference between the two scores is a critical challenge in supervised learning [61].

3.2.4 Overfitting vs under fitting

Overfitting is characterized by a high training score and a low test score. This occurs when a model is too complicated or the model has been trained so many times that the model has memorized the training data instead of finding a general pattern. When applying to simple models the data is underfitted, and the model is not complex enough to discover the general pattern in the data. Underfitting leads to a low training and test score but a low variance between them [61]. The case of overfitting is known as the high variance case and the underfitting as the high bias case [68]. A challenge in machine learning is finding a good trade-off between over and underfitting. These three cases are visualized in Figure 9.

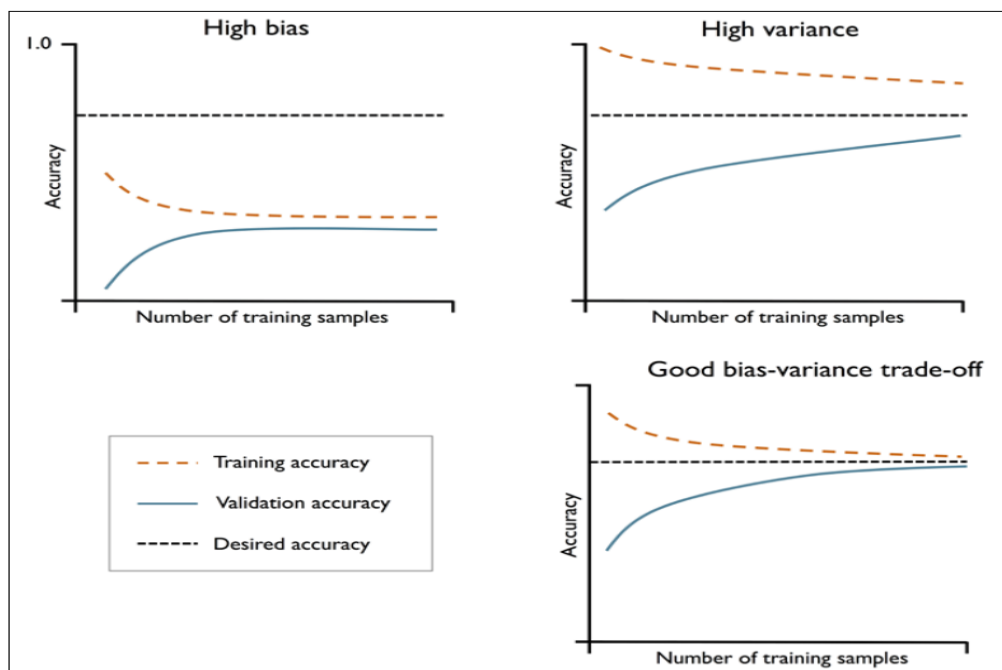


Figure 9: The accuracy as a function of number of training samples for the high bias and high variance case. A graph for a good trade-off between the two cases is also shown. Obtained from [69]

Overfitting can be avoided by getting more data. When getting more data is not possible, then restrictions on the amount data the model can store or what information the model can remember should be restricted. This type of restrictions is in machine learning known as regularization techniques. Regularization techniques forces the model to remember the most prominent patterns. Applying less complex models will also reduce overfitting. To prevent underfitting the opposite methods can be applied [61].

3.2.5 Validation methods

A normal approach in machine learning is splitting data into training and test data. The models are trained and tuned on the training set. When the accuracy is high enough the model is tested on the test data. Often in machine learning the models are trained and tested many times with different choices of hyper parameters to obtain a high accuracy, eventually the test set becomes a part of the training set as the model favored is the one performing best on the test set. This can lead to a poor generalization for future data as it favors a specific test set and not necessarily a model that on average performs well on different test sets. Therefore, the test set should only be used one time after the final model is chosen. To achieve a higher generalization validation methods are used [61].

The holdout method divides the data into training, validation and test set. During the training the validation set is used as test for all models computed. Approximately $2/3$ of the data is used for training and validation. The final model is chosen as the model with best results based on the training and validation scores. The drawback of the holdout method is its sensitivity to how the data is partitioned [61].

The k-fold cross validation reduces the sensitivity by randomly splitting the training data into k folds. $k-1$ of these folds are then used for training and the last one is used for validation. The data is then trained k times, so all folds are used $k-1$ for training and one time for validation. For the final model the data is trained on all k folds and tested on the unseen test set [70]. Validation methods increases the robustness of the model and reduces bias in the model selection [61].

3.2.6 Evaluation metrics

Evaluation metrics gives information about the performance of a model and are important to obtain the optimal model. There are many metrics available depending on the data and problem in hand. For classification, accuracy and confusion metrics are used. For regression problems mean absolute error or R^2 can be used. Below the toolbox of evaluation metrics used in this thesis is presented.

Classification accuracy is defined as the ratio of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (1)$$

The metrics gives the best result when used on evenly distributed class data, uneven class distribution can lead to false interpretation. Classification accuracy is the most used evaluation metric for classification problems [71]. In literature, the metric is often referred to as accuracy.

Confusion matrix: The confusion matrix compares the predicted class to the actual class and gives a visual representation of how the misclassifications are distributed. The method can be applied to multiclass problems as well [72]. An explanation of the different entries in the matrix is presented in Figure 10.

True label	CO	Correctly classified CO measurements	CO measurements classified as NO ₂
	NO ₂	NO ₂ measurements classified as CO	Correctly classified NO ₂ measurements
		CO	NO ₂
		Predicted label	

Figure 10: Explains the different entries for a confusion matrix applied to a two-class problem.

Mean Squared Error (MSE) is commonly used as a standard statistical evaluation metric in regression problems. The method assumes the error in the samples being unbiased. And is given by:

$$MSE = \frac{1}{n} \sum_{i=1}^n e_i^2 \quad (2)$$

Where n is the total number of samples and e_i is the error between the actual target value and the prediction. MSE performs well on normally distributed data. The drawback of the metric is its sensitivity to outliers .

R² score is a goodness-of-fit measure for linear regression. Linear regression calculates the equation resulting in the smallest sum of squared errors that is possible for a given dataset. The R² score calculates the proportion of variance in the response variable which is explained by the feature. In machine learning, the R² score can be used to determined how close the predicted value is to the target value. If the prediction is 100% correct the R² score would be 1. When a model is evaluated the R² score for the whole dataset is calculated as the average R²-score between al target- prediction pair in the dataset. In Excel the R² is calculated as [54]:

$$R^2 = \sum_{i=0}^n \frac{(y_i - \bar{y})^2}{(y_i - 0)^2} \quad (3)$$

Where y_i is the data points and \bar{y} is the average of all data points.

3.2.7 Optimization

Machine learning consists of a big selection of methods, these methods have a variety of parameters to tune. Finding the best method with the best parameter values is called hyperparameter optimization(HPO). HPO is crucial for optimizing the model's performance. The large variety of methods and possible parameter values makes the tuning process computationally expensive and time demanding [73].

In 1997 Wolpert and Macready presented the “No free lunch theorem”(NLFT) [69]. NLFT states that finding a universal optimization strategy that works best on all problems are impossible [74]. The NLFT implies that optimization strategies must be tuned for each problem individually for a model to perform as good as possible as no model is superior on all problems and datasets. Due to the statement in NLFT optimization of parameters should be prioritized.

The increased use and demand for machine learning in general, has sparked the interest for commercially available hyperparameter optimization strategies [75]. These strategies should be able to perform an optimization among many different machine learning methods and a grid of different parameter values [73].

Due to a vast number of methods and parameters the HPO is computationally expensive. Other challenges are: parameters vary between integers, floats or categorical lead to a complex space, limited training size limits the generalization ability. Commercial products must also have the ability recalibrate after the product is shipped to the end user. This requires an embedded optimization system, or the possibility for optimization through the cloud [73].

Several HPO methods have been developed, in the thesis the most prominent will be discussed. Grid search is the most used method for optimizing hyper parameters [73]. The method uses a brute-force exhaustive search. The programmer defines a grid of parameters with a range of different values and the search evaluates all combinations within the grid. Grid-search often provides high performances, but it also computational expensive when testing several algorithms with multiple parameters.[61].

Random search is emerging as an alternative to the grid search and has in some applications proved more efficient, with a fraction of the computation time. Random search draws randomly chosen combinations of parameter and parameter values from a predefined value space.

Random search is often effective as a baseline as it gives a good overview of different hyperparameter values within a shorter timeframe than the grid-search [76]. For information on several HPO methods, the reader is recommended [73].

Neural Architecture Search (NAS):

Increasing use of neural networks has initiated the field of automating the architecture selections for neural networks. The field overlaps with the HPO. NAS consists of three building blocks. 1) Search space containing the possible architectures that can be evaluated in the search. This introduces a bias toward the person defining the space. 2) Search strategy: How the search is conducted. 3) Performance estimation is the strategy for evaluating the performance [77]. Examples of performance metrics are given in 3.2.6 Evaluation Metrics. As most NAS strategies developed are used for image classification they are not presented in this thesis. During the literature study, no algorithm or approach for NAS on output from sensors was found.

Benchmarking:

In the machine learning community, no general benchmark requirements are set for HPO methods. Comparing different methods in an objective manner is therefore a complicated task. Feurer and Hutter states that the community needs to set clearly defined metrics, but that different methods work best on different problems which makes this hard. Further they suggest that the performance requirements should not be set, but rather set a standard for how the methods should be evaluated [73].

3.2.8 Automated Machine Learning

The number of companies and private persons interested in implementing and using machine learning is increasing. To be able to satisfy all the demands and tasks the field of automated machine learning (AutoML) is quickly developing. ML problems that consist of both selection of best algorithm and hyperparameter optimization are referred to as CASH problems [78].

Automated machine learning aims to automate the whole machine learning pipeline, including data collection, pre-processing, prediction and evaluation. The pipeline will be able to test several models and choose and exhaust the best model without orders from the user. By automating the entire pipeline, the technology and possibilities within machine learning are made available for people with low technical experience and knowledge. The automated pipeline will make decisions in an objective and effective manner [78].

3.2.9 Benchmarking in machine learning

As mentioned earlier there is a large variety of classification algorithms, hyperparameter optimization techniques and a nearly infinite number of NN architectures. This leads to a practically impossible task of calculating and comparing all possible models. Even when having a limited number of models that are calculated ranking them can be difficult [73]. As models can be ranked according to different criteria's:

- III) *Performance*: How many classifications are correct or for regression how low the estimated error is.
- IV) *Computational Complexity*: Time consumption of the model.
- V) *Overfitted/ underfitted*: If the model is overfitted the performance will most likely plummet when exposed to data in the future, and should therefore not be ranked high even though it has a high performance.

In some applications, the performance accuracy is prioritized before computational complexity (time consumed). When predicting if someone has a severe diagnosis a high accuracy is preferred compared to low time consumption. When calculating the CO₂ concentration in a city the possibility of measuring on many sites at once and many times a day is more important than knowing if the concentration is 407 or 411 ppm. The Auto-CMUT developed in the thesis ranks the model after performance.

3.2.10 Pre-Processing

Reduction of noise and removal of outlier is crucial to obtain a well generalized model. Reducing noise and preparing the data for prediction is done during pre-processing. Due to the vast number of methods to pre-process and alter the data it can be a time demanding process [79]. The pre-processing steps looked at in this thesis are:

- Data visualization
- Data cleaning
- Feature selection/ extraction
- Visualization

3.2.11 Data Cleaning

Data cleaning deals with missing values, abnormal values(outliers) to improve data quality [80]. In this section frequent used methods for data cleaning is presented.

Missing values

In real world applications, the datasets are often noisy and often contain missing values [36]. These value can lead to misinterpretation and inaccurate predictions [80]. In this section, the most frequent strategies for dealing with missing values is presented.

Remove missing values: A popular method is removing all samples with one or more missing values. Especially popular for datasets with many samples. The method can lead to loss of information in a dataset with many missing values. If many values are missing the validity of data should be discussed [36].

Inserting values: This method consists of inserting a value for the missing value. Examples is inserting the mean value for that feature or inserting the most common value. [36]

Missing values as an own value: Instead of trying to discover the true value, the missing value is an own value and treated as other values. [36]

The methods mentioned in the thesis are traditional methods for missing values. Traditional methods for missing values can lead to biased results [81]. The problem of bias when handling

missing values will not be discussed further in this thesis. In addition, the problem on missing values specifically in time-series problems are not looked at, but should be investigated in a later stage of the project.

Outlier detection

Outliers are defined as data points that strongly deviates from the rest of the samples in the data set [82]. Typically, the threshold for strong deviation is defined as three standard deviation away from the mean. Outliers normally form a small part of the data set, but ignoring them may cause the findings in the work to be invalid [4]. Outliers are mostly considered to be noise, but can contain important information [82]. Because of lack of time outlier removal was not implemented into the Auto CMUT.

3.2.12 Feature Selection

Feature selection refers to the process of choosing the best subset of features. In cases where the dataset consists of many features the model can often consist of redundant information. Redundant information can confuse the model and making it harder to detect the most prominent patterns in the data. In such cases, the model can improve by removing some of the features or by combining some of the features to decrease the dimensionality of the data. By reducing the dimensionality of the data, the computation cost and degree of overfitting is also reduced [61].

In machine learning, Principal Component Analysis (PCA) is widely used for choosing the best subset of features. PCA reduces the dimensionality of the data by creating features in the data based directions in the data. The directions or Principal Components are computed as the features that capture most of the variance in the data [61] the working PCA is illustrated in figure 11.

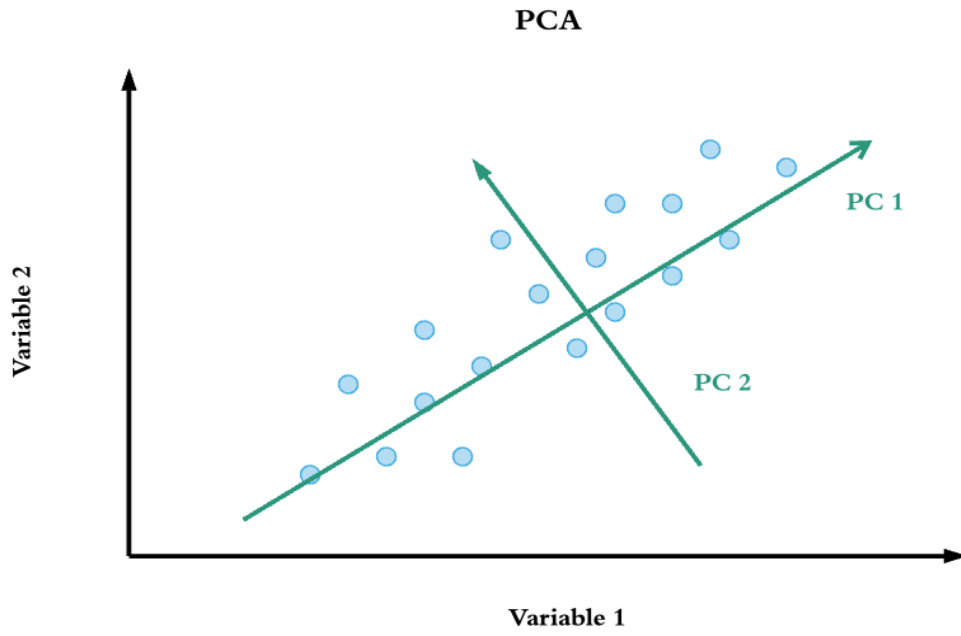


Figure 11: PCA finds the direction that capture most of the variance in the data. PC 2 finds the direction that is orthogonal to PC 1 and capture the second most variance.

3.3 CLASSIFICATION ALGORITHMS

In this section, the classification methods tried in the thesis are introduced. Classification algorithms aim to predict a samples class membership. When used on CMUT sensor the algorithm should classify which gas a sample belongs to.

3.3.1 Logistic Regression

Logistic regression is one of the most common algorithms used for binary classification in machine learning. In the *sklearn* package used in this thesis the method is extended for use in multi-class problems. Logistic regression calculates the probability that a sample belong to one class and not any of the others(multiclass). For predicting class membership, a cost function is used. The function takes a linear combination of the input values x and the corresponding weights for each of the x values:

$$z = w_0x_0 + w_1x_1 + \dots + w_nx_n \quad (4)$$

z is a linear combination of the input values x and the weights, w for each input.

During training these weights are updated to minimize the cost function. In logistic regression, the following cost function is used:

$$\varphi(z) = \frac{1}{1 + e^{-z}} \quad (5)$$

Where φ is the probability for z belonging to a certain class, and z is the net input. The probability lies in the range $0 \rightarrow 1$.

The advantages are it's is ease of implementation, computationally efficiency and it does not assume any distribution of the data. The drawback is that is has a linear decision border and is mainly applicable for linear problems [61].

3.3.2 Decision trees and Random Forest

Decision trees are used both for classification and regression tasks. When used for classification the method will predict a target label for each of the final nodes. The algorithm divides the data into several nodes by performing a series of binary splits until a certain criterion is met, as illustrated in Figure 12 [4]. If a maximum number of splits isn't specified the algorithm will split until there is only one sample in each node, which lead to overfitting. Therefore, the maximum depth of each tree should be specified. To increase the robustness of the model an ensemble of trees that individually suffer from high variance is used to gain a better generalization performance. This ensemble of trees is called random forest. The algorithm averages the results from many trees to make a final prediction [61].

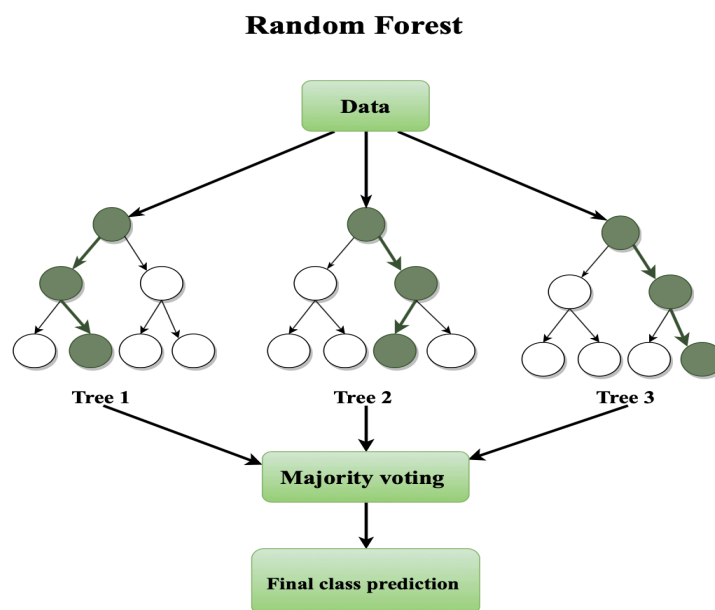


Figure 12: RF consists of individual decision trees. In each tree the information is split to maximize a criterion. The final prediction is computed as a majority vote between all individual trees.

Random forest has become a popular choice because of their high performance, scalability and simplicity. Random forest is quite robust against noise from the individually trees. Decision trees can produce complex decision boundaries and is therefore not limited by linearly separable classes. The disadvantages RF is the possibility for overfitting, and deep trees can lead to long computation time [61, 83].

3.3.3 Support vector machines (SVM)

Support vector machine have in recent years gotten increasing attention in the applications of classification, regression and detection of outliers. The objective of the algorithm is to maximize the margin. The margin is the distance between the samples closest to hyperplane that separates the data into different classes. These samples are called support vectors. Larger margins lead to a lower generalization error, lower risk of overfitting and lower probability for misclassification [84].

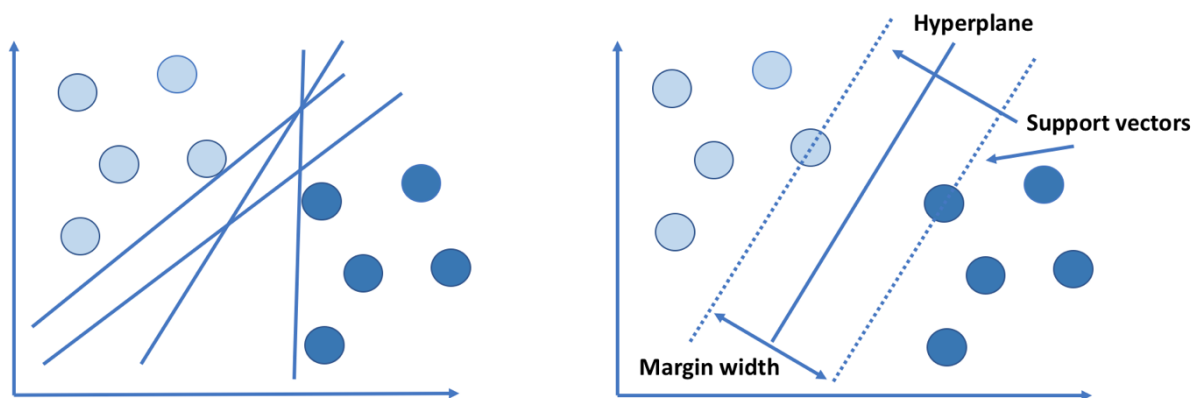


Figure 13 : a) shows the wide range of hyperplanes that can separate the two classes (dark and light blue dots) b) The hyperplane that maximizes the margin. Made by Byrne, Maureen

For nonlinear problems, the kernel trick is usually used. The kernel method deals with nonlinear problems by combining the original features in a nonlinear way and projecting them onto a higher dimensional feature space using a mapping function. In the higher dimensional feature space, the data becomes linearly separable. Depending on the data different kernels can be used, which all are based on the calculations of the inner product between all the samples. The kernel functions can all be interpreted as a similarity function between all the samples [61]. Figure 14 shows the most used kernel functions and how they separate the data.

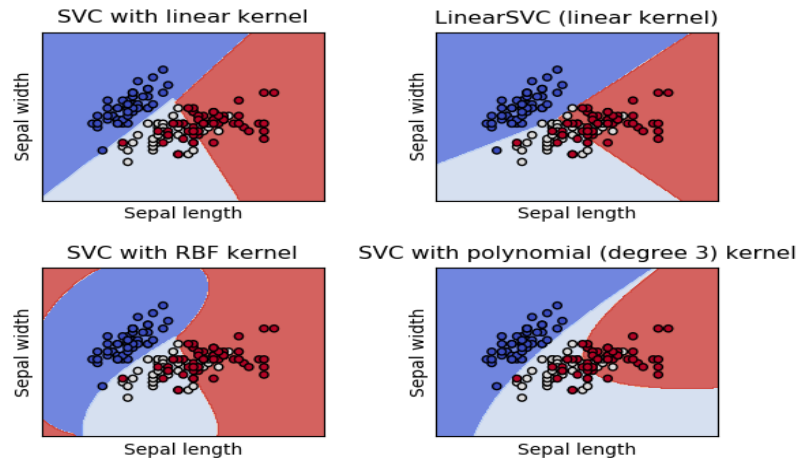


Figure 14: How the data is separated by the different kernels in SVC.

SVM has become popular especially in classification task because of the high performance on nonlinear problems, it can also use different functions locally on the data. This gives the model a high degree of flexibility. In addition, no assumption about the distribution of the data is needed. As the model focuses on the support vectors, it is robust to outliers. The outliers will gain low importance for the prediction. A drawback with the SVM becoming computationally expensive when having a high dimensional space [4].

3.3.4 Boosting algorithms

Boosting algorithms in general combines several classifiers with individually low accuracy into a meta classifier with higher accuracy [61].

Adaboost: In 1997 the AdaBoost algorithm was put forth by Freund and Schapire [85]. The AdaBoost starts by fitting one weak classifier and then checks its performance. The weight or importance of the misclassified samples are then increased and passed into the next classifier. Again, the misclassified samples receive a higher weight. All classifiers compute a score value. The final meta classifier is computed as a linear combination of all weak classifiers [86]. The boosting algorithm has gotten especially high performance on two-class problems, when used in multiclass problems the performance has a tendency to obtain lower values [87]. Adaboost is special case of the boosting algorithm gradient boosting that aim to minimize some error function [61].

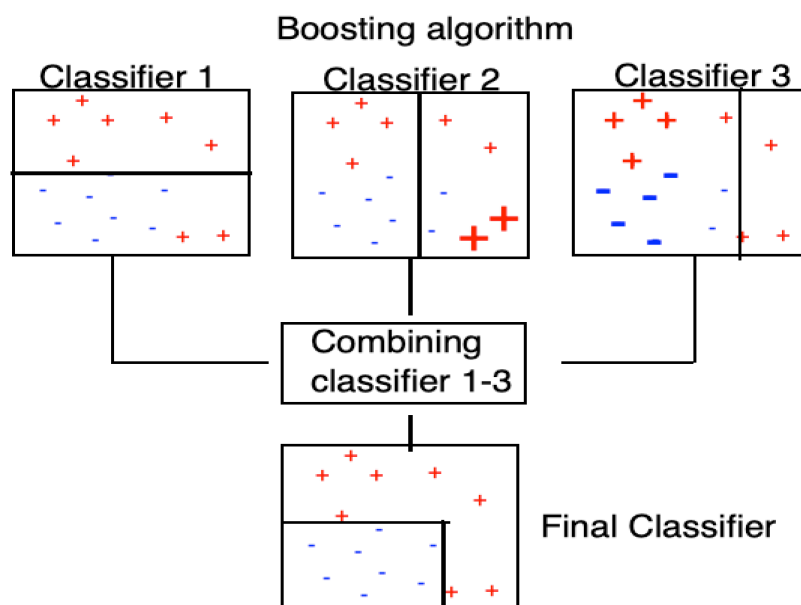


Figure 15: working principle of a boosting algorithm. The results from each individual classifier is combined to a final classifier. Modified from[61]

3.4 ARTIFICIAL NEURAL NETWORKS

One of the primary goals in this thesis is to develop a neural network with the ability to predict concentration after the initial classification. This section will introduce the branch of machine learning called deep learning that includes neural networks.

3.4.1 Artificial Neurons

Artificial neurons are the building block of all neural networks. The workings of an artificial neuron are derived from studies done on the human brain and tries to mimic its behavior. Neurons are often referred to as nodes in neural networks [88].

The input variables are weighted before they are forwarded into core of the neuron. In the core the inputs and weights are combined according to an activation function. Depending on the calculations of this activation function the network decides if the information obtained should be forwarded further into the network or not. For example, if the values from the activation function is higher than a certain threshold the information is passed further into the system. The activation function in the network may be both linear and nonlinear. Figure shows the workings of a neuron [88].

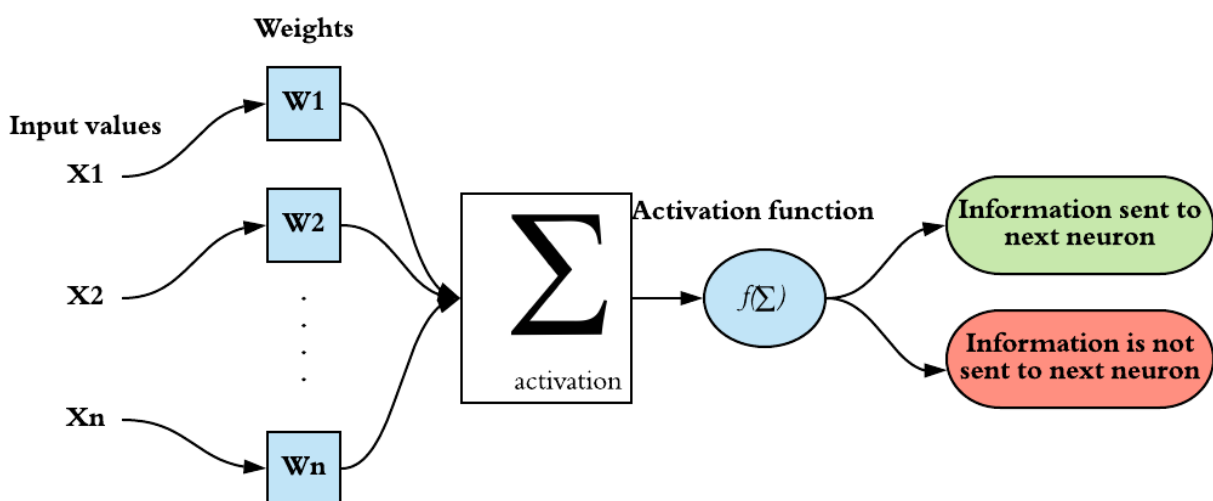


Figure 16: The Figure shows how the information flows through an artificial neuron. Modified from [84]

In Figure 16 x_n are the input values, w_n are the weights, the transfer function sums all the products between x_n and w_n and the activation function decides whether the information should be forwarded further.

Whether a neuron is activated or not is decided by the activation function. Using different activations enables the possibility to discover both linear and unlinear relationships in the data. The activation function calculates some value based on the weights and biases. This value can activate a neuron [61]. In Appendix A, an overview of different activation functions is given.

3.4.2 Neural Networks

Earlier the information through an artificial neuron was explained. Next, the organization of the neurons into a network is discussed. Neural networks try to combine the advantages of both computers and the human brain. The computer is far better at crushing numbers at high speed while the brain quickly recognizes a face from different angles and position or recognize a sound. The goal with neural networks is to combine the number crushing capacity of the computer with the brain ability to easily recognize patterns [4].

Neural networks consist of three main layers:

1. **Input layer:** All the inputs are received and normalized within the range of the activation function.
2. **Hidden layers:** There can be one layer for a simple network or several hidden layers for deep neural networks. Hidden layers do most of the processing of data and aims to extract patterns from the data.
3. **Output layer:** The output values are presented. In classification problem, the final activation function must give a discrete value and consist of one node for each class. Regression problems must have a continuous value as output [4].

The architecture of neural networks can be divided into feedforward networks, recurrent network and mesh networks. Feedforward networks has as its only criteria that the information flows from the input layer to the output layers without any loops. Feedforward are generally simpler to train, but not sensitive to the history of the network. Recurrent networks can send information in both directions using loops. Normally used if values for previous inputs are of importance [89]. In Figure 17 the difference between feedforward and recurrent are showed.

As effects of chemical degradation in the sensing layer are outside the scope of the thesis only feed-forward networks was implemented in the Auto CMUT.

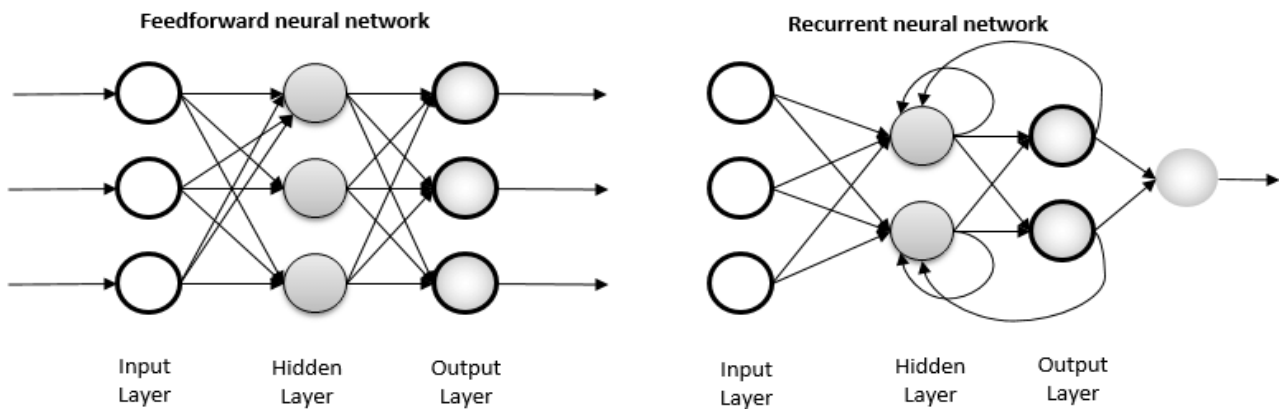


Figure 17: Shows the difference in information flow between feed-forward networks and recurrent networks. Reused from [113]

There are several different activation functions being used in neural networks. If only using linear function in the hidden layers of the model, the resulting network will never be more than a simple linear regression model. By adding non-linear functions the network can learn more complex relations. Within the same layer all neurons use the same activations function, but between different layers the activation function can vary [4]. In Appendix A an overview of different activation functions is given.

3.4.3 Multi-layer perceptron

A fully-connected feed-forward network with multiple hidden layers is in deep learning called a Multi-Layer Perceptron(MLP)[61]. In this thesis MLP will be used as both a classifier and regression method. Figure 18 shows the architecture of a MLP

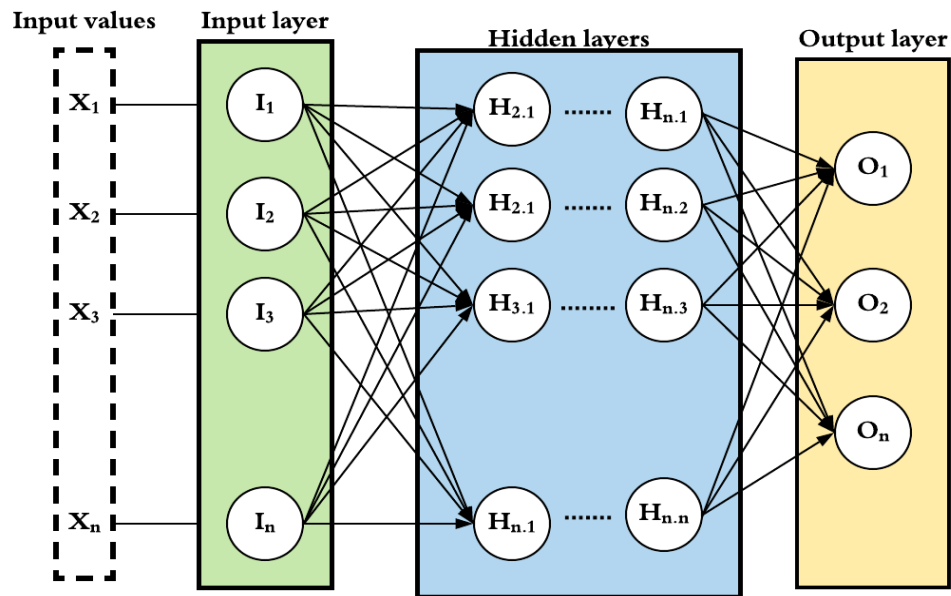


Figure 18: Shows how a MLP is organized. If used on a classification task the number of output nodes matches the number of classes. If used on a regression task the output is a continuous value. Modified from [90]

4 METHOD

In this section, the four main steps in the Auto-CMUT are described. The steps are illustrated in Figure 19. In step 1 the datasets used in the thesis are presented. For the CMUT dataset the test setup for data collection is also described and illustrated. In step 2 the pre-processing steps in the algorithm are described. The pre-processing step aims to prepare the data for prediction by dealing with missing values and select the best set of features. Step 3 puts forth the classifiers, the neural networks used and present the structure of the algorithm. The classifiers are used to determine which gas each sample belongs to. In the last step, the evaluation techniques and structure are presented. By using different evaluation metrics, the performance of the algorithms is evaluated.

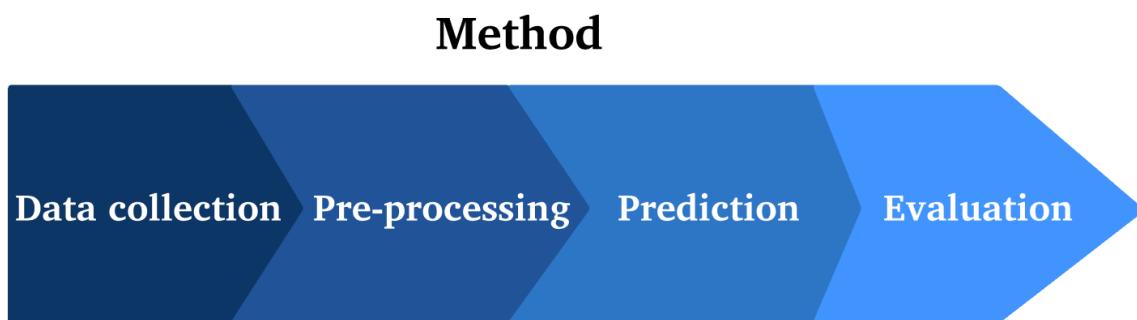


Figure 19: Organization of the Auto-CMUT.

It is important to note that due to the high number of machine learning techniques and algorithms available in Python, it is out of the scope of this thesis to test them all. The selection of techniques and algorithms covered in this thesis is based on the literature study done in January.

4.1 LITERATURE STUDY

The main part of the study was performed during the first 3 weeks of January 2019. The methods implemented in this thesis are methods that frequently appeared in relevant literature. The literature used was mostly found using the scientific databases Google Scholar, Web of science and Oria.

4.2 AIR QUALITY DATASET

To obtain a benchmark for the CMUT to be tested against the Auto-CMUT is applied to a dataset from a MOX sensor. This test will also work as a Proof of concept on how the algorithm performs on a larger dataset.

The air quality dataset contains 9358 measurements of 6 gases. CO, NMHC, NO_x and NO₂ has one measurement from the MOF sensor and one measurement from a reference sensor. The reference measurement will be used as the ground truth in this work. O₃ and C₆H₆ has no reference measurement and will therefore be removed from the dataset. Missing values in the dataset are tagged with the value -200. The device consists of 5 metal oxides sensors embedded into one device. The measurements are performed on an hourly basis from February 2003 to March 2004 to in addition to gas measurements the device measures temperature, relative and absolute humidity.

4.3 SOFTWARE

The algorithm developed during this work is implemented in Python 3.6. In addition, parts of the libraries *matplotlib*, *talos*, *scikit-learn*, *missingno*, *numpy*, *pandas*, *keras* and *seaborn* are used [91-97]. Both the classification algorithm and regression step of the Auto-CMUT outputs tables in excel. The regression figures are also plotted in excel.

4.4 PRE-PROCESSING

Before forwarding data into the prediction algorithms the dataset was visualized and processed. To ensure that the best feature set was used, the original features were compared to transformations by evaluating the feature importance. Missing values were removed to eliminate noise from the dataset. Figure 20 show the structure of the pre-processing.

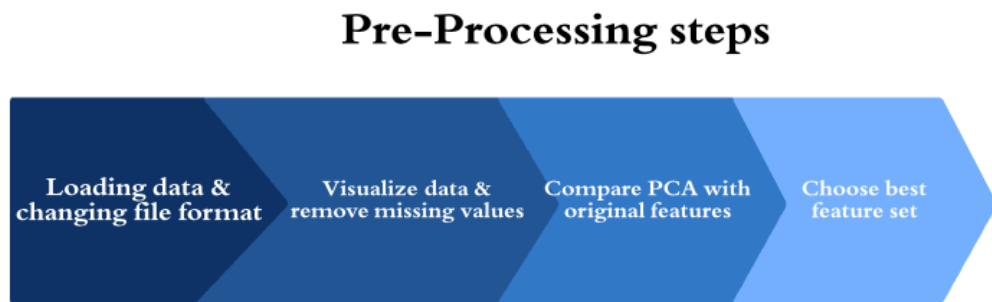


Figure 20: Pre-processing steps for the Auto CMUT.

For both datasets missing values were located and visualized by using matrix and bar plots from the *missingno* package in Python . The values were all removed from the dataset.

The final feature set was selected based on comparing feature importance's for original features and principal components. By using pair plots the difference between the feature sets was visualized.

All features and target values were scaled to prevent some of the features or the target to dominate the prediction during the classification and prediction part.

4.5 PREDICTION

In this section methods for both classification and regression are presented. During classification, the algorithm aims to recognise the gas type of each measurement. After performing a classification regression is used to make a prediction of the concentration of the sample. A flowchart for the prediction step is shown in Figure 21.

Prediction steps

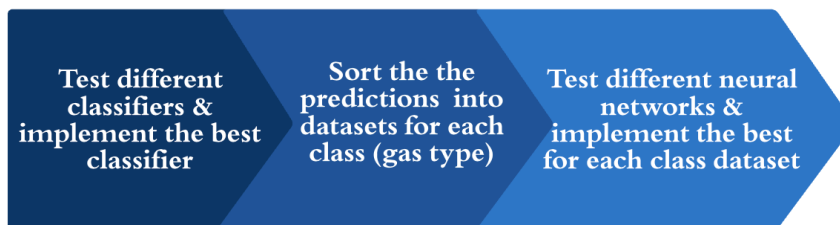


Figure 21: The different prediction steps.

4.6 CLASSIFICATION

The classification algorithms aim is to determine a samples class membership. In this thesis, the classes consist of different gases. The higher selectivity against a specific gas, the better the results. In this thesis *SVC*, *logistic regression*, *Random Forest*, *AdaBoost*, *GradientBoosting* and *Multilayer-perceptron* was implemented. By using *gridsearchCV()* from *sklearn.model.selection* all parameters combination was calculated with values decided by the programmer. The results from the different classifiers was summarized in a table.

During the tuning process, 70% dataset was used for training and 30% for testing. The testing data is forwarded into the model after the training and tuning of parameter. This ensures an unbiased result of how the model will perform on unseen data. During training, random subsets of the training data is used for training and test. By testing and training on random subsets several times, the model gives an overall performance based on many combinations of training and test subsets.

4.6.1 Implementing different classifiers

Random forest was implemented by using the `sklearn.ensemble.RandomForestClassifier()` in Python. Without restriction, the algorithm continues to split the nodes until each sample have their own node, this will lead to overfitting which is prevented by setting a maximum number of splits. The number of splits is restricted by the `depth` parameter. To prevent high variance from individual trees, the performance from many trees are averaged. The number of trees is set by the parameter `number of trees`. The `impurity` parameter defines the function used for splitting data [98].

Support vector machines The method was implemented using the package `sklearn.svm.SVC()`. Using the grid search function in python the polynomial, radial basis function and linear kernel was tried with a selection of values for the parameter C and γ . Low C values mean a low penalty for misclassification and therefore a wider margin, while a high C leads to large misclassification penalty and a narrower margin. The γ parameter is used for the rbf kernel. A high value means a that the samples have a high influence and often a bumpier decision border between classes [99].

Logistic regression was implemented using the function `Sklearn.linear_model.LogisticRegression()`. Grid search is used for tuning the C value. The C parameter defines how strictly misclassifications are penalized. A high C value gives a high penalty for misclassifications [100].

Adaboost was implemented using `sklearn.ensemble.AdaBoostClassifier()`. The number of classifiers was set by the `n_estimators` parameter. To restrict the influence of the misclassified, sample a `learning rate` is set. The `learning rate` ensures that the global minima are not overshoot. By using Grid search different combinations of values was tested [101].

Multi-Layer Perceptron was implemented using `sklearn.neural_network.MLPClassifier()` function. The MLP classifier include a high number of parameters. To prevent overfitting and too complex models a selection of parameters was chosen. The `activation` parameter was used to determine the activation being used in the hidden layers. The `learning_rate` regulates how much the weight of the activation function are updated per epoch. The number of hidden layers in the network is set by the `hidden_layers` parameter [102].

4.6.2 Development of multiple classifier model

The algorithm developed in this thesis implements a multiple classifier selector. The model evaluates multiple classifiers and performs a grid search over several parameter values, which are manually decided before running the algorithm. The results are automatically sorted by the highest mean test score. The results from all the tested classifiers with their parameter values and scores are summarized and converted into an excel file. The finalized model then implements the classifier with the highest mean accuracy score with the best choice of parameters and trains on the whole training set. The finalized model predicts the test data and sort them into one dataset for the NO₂ measurements and one for the CO measurements. These datasets are forwarded into the Neural Network.

For the thesis, the classifiers: Random Forest, Gradient Boosting, Ada Boost, Logistic Regression, SVC and Multi-Layer Perceptron are all from the *sklearn* package in python. For future use the classifiers can easily be changed by a programmer with basic knowledge of machine learning and python.

4.6.3 Development of neural network for regression

The concentration is predicted by tuned neural networks. To implement the model the *Keras*-package was used [103]. Keras is python library specialized on neural networks. The Keras package is run on top on Tensorflow and is easier in use [4]. To perform a randomized grid search Talos was used. Talos is a Python-package for hyperparameter optimization with Keras [104].

The regression step aims to try a selection of simple models. By obtaining an error estimate the model with the lowest scores are selected and applied to the test data. Error estimates calculates how much the predicted concentration differs from the true concentration. In this thesis, the mean squared error(MSE) and R² score are used.

Because of the wide range of possible tuning choices only the ones used in the best grid search are presented in this thesis. For further documentation on other methods and possibilities in Talos, check the Talos user manual [105]. In Table 5 and overview of the different parameters and layers included in the grid search is presented and the values tried. Information for the Table was found in Keras user documentation [106].

Table 5: Presents the parameters used in the final neural network along with its grid of parameters.

Layer hyperparameter:	Explanation	Values / functions
Dropout	Sets a fraction of the nodes in the NN to zeros. Prevents overfitting.	0,0.2,0.4,0.5,0.1
Hidden layers	Decides the number of layers between the input and output layer.	1,9,100
Activation function	The value calculated from the activation function decides if a neuron/node should be activated or not. In keras several activation functions is supported	Relu, Elu, Sigmoid, Lineaer
Loss functions	The function used to calculate the error between the predicted	Mean-Squared Error and R ² -score
Batch size	The number of samples being trained before updating the node values.	20, 50
Epochs	Number of times the samples are used during training	40,60
Optimizer	Optimizers update the model in response to the output from the loss function.	Adam
Learning rate	The learning rate regulates how much the weight of the activation function is updated per epoch	0.1,0.2,0.3

4.6.4 Automated Pipeline

Most end users and buyers have both limited experience and knowledge in implementing machine learning models and dealing with sensor output. An automated pipeline for the CMUT should therefore be included in the product package. An automated pipeline for handling sensor data will make chemical micro sensors more suitable for commercial purposes and reduce the manual labor and the costs associated this work.

First the pre-processing step, classification step and regression step were implemented in separate scripts and was run chronologically. Further work was put in to automating the whole process so all steps could be run in one single script. An overview of the pipeline is presented in Figure 22.

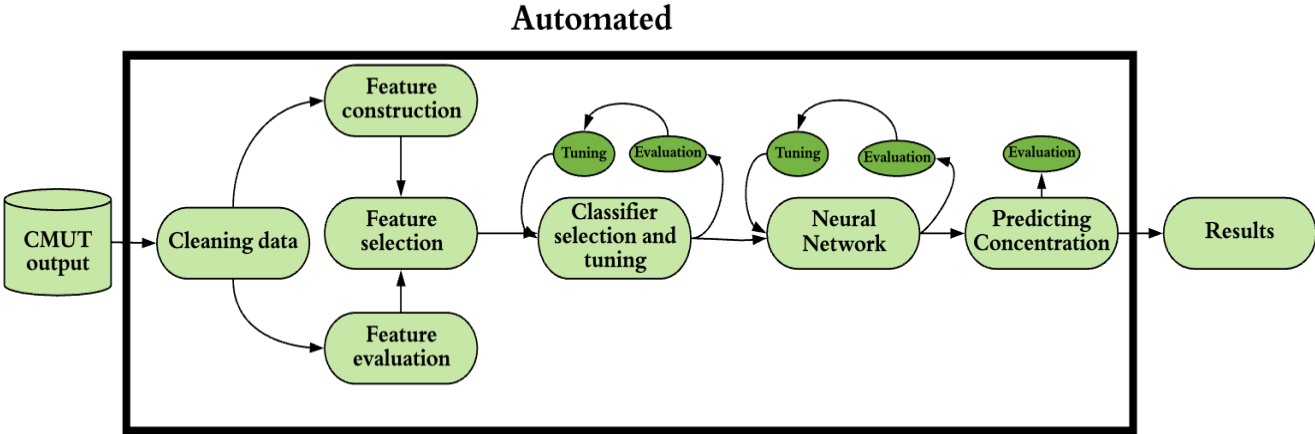


Figure 22 Overview over the steps included in the Auto CMUT

4.6.5 Similar work

Historically manual machine learning approach has dominated. In the manual ML approach, lots of tuning and manual work is put in when training the models. AutoML is a new subfield of machine learning and there is limited systems or implementation routines available in literature. In this section, similar work is presented in Figure 22.

WEKA:

The WEKA workbench was first launched in 1994 [107]. Since then it has been improved and extended. The WEKA workbench proved a collection of different machine learning algorithms. The workbench quickly tries different algorithms for the user. It contains algorithms for classification, regression and clustering [108]. The workbench is a stand-alone system and is not a module that can be run from python.

Auto-WEKA:

In 2013 an automated pipeline for pre-processing and classification was published [109]. It was the first algorithm to tackle the CASH problem. The CASH problem addresses 1) That no method work best on all datasets and 2) Some methods are sensitive to the choice of hyperparameter values [75]. The Auto-WEKA package is available open source and version 2.0 was published in 2017 [110]. The system is to be used together with the WEKA workbench.

Auto-sklearn:

In 2015 an algorithm for pre-processing and classification was presented [75]. The algorithm is based on scikit-learn in contrast to WEKA and Auto-WEKA. The algorithm is also intended for a broad selection of datasets with a high generalization performance. The algorithm aimed to improve the efficiency and robustness of Auto-WEKA and WEKA [75].

4.6.6 The Auto-CMUT

The Auto-CMUT algorithm developed in this thesis is based on scikit-learn, but differs on several points from the Auto-sklearn. In contrast to the Auto-sklearn, it is specifically intended for use on data from CMUT sensors, but should be easy to use on similar datasets. In addition to pre-processing and classification, the Auto-CMUT algorithm includes a regression step by using Neural Networks in Keras in contrast to the Auto-sklearn.

4.7 EVALUATION

During evaluation, the models with the best training performances was calculated with use of the test set. The results were then compared to the true answer, and finally summarized using different evaluation techniques. For the classification task accuracy and confusion matrix were used. For the regression MSE was used during the training of the network while R^2 was used to evaluate the predicted test set values. R^2 scores used on the test set were used to perform a benchmark test against the commercial sensors in section 2.5.

5 RESULTS

In this section, the results performed from the Auto-CMUT are presented. The results from the classification and regression part will be presented separately as the two parts give different information and are based on different grid searches. In addition, some visualization of the data is provided.

5.1 VISUALIZATION

The automated pipeline is meant to provide information of class presence and gas concentration. As the user of the pipeline should not need a computer and evaluating data plots, any visualisation of the data is excluded from the pipeline. However, during the work writing the algorithm the data was studied and visualized to decide on elements like feature importance as well as getting to know the data at hand. These visualisations are presented in the following section.

Principal Components vs original features:

To decide whether a transformation of the features combined with a dimension reduction was preferred, the principal component and original features were presented in pair plots. The results are shown in Figure 23.

The pair-plots show how the classes are distributed between two features. From Figure 23 the feature pairs giving some difference between classes are PC3- PC1, PC3- PC2 and PC3-PC4. For the original features, some separability can be seen between the MOX sensor and all the other features (RH, AH and T).

By looking at the pair plots it seems like there is little difference when using the PCs or original features. Due to the low number of original features most algorithms won't have problems dealing with the dimensionality of the MOX and CMUT datasets. As the dimension is low, and there is a small difference between PCA and original features, only the original features were used in this thesis.

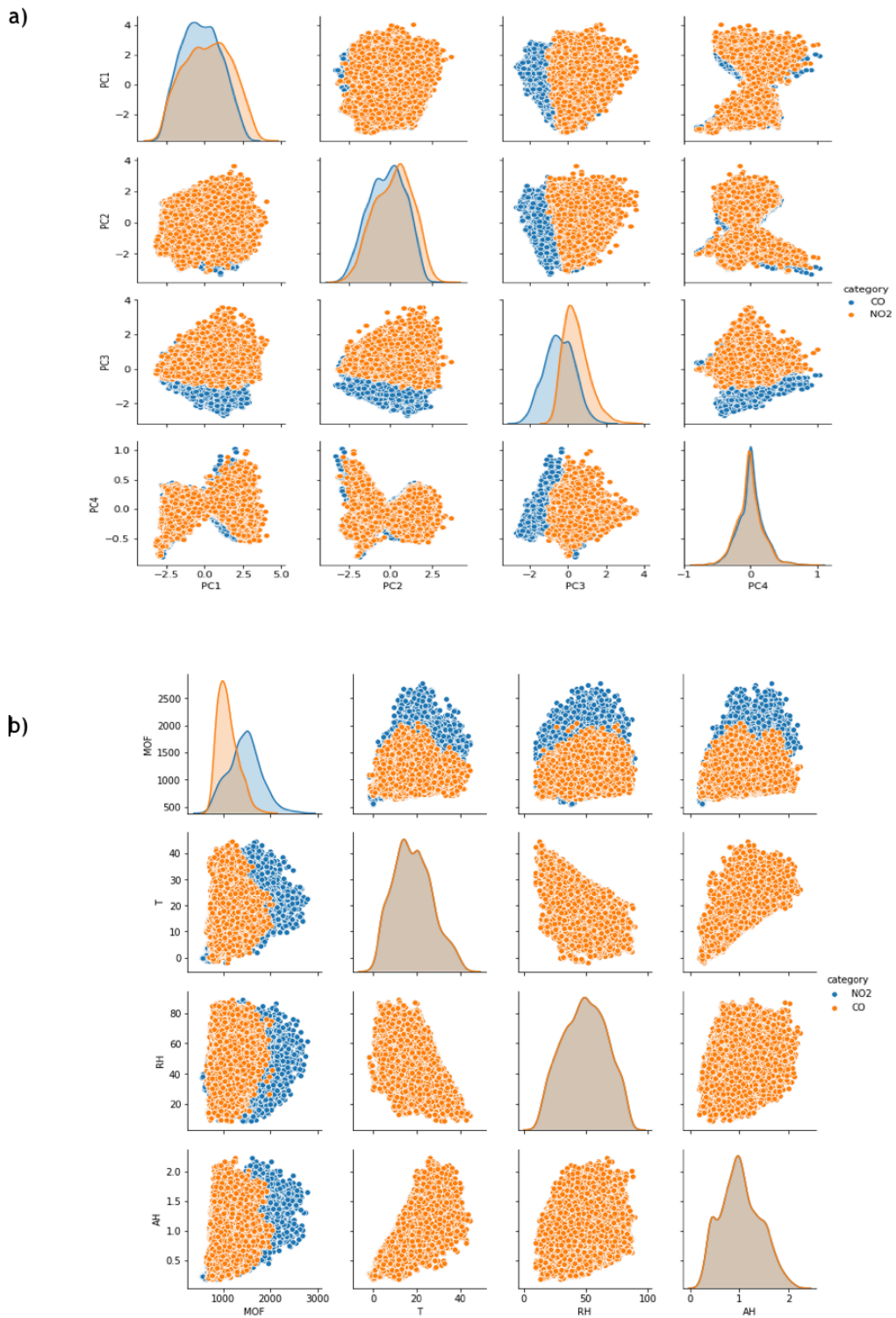


Figure 23: Shows the separability of classes when plotting two features against each other

5.2 PRE-PROCESSING

Missing values

In the thesis, the most common strategy of removing missing values was used. To investigate the source of missing values some plots from the *missingno* package in python are presented in Figure 24

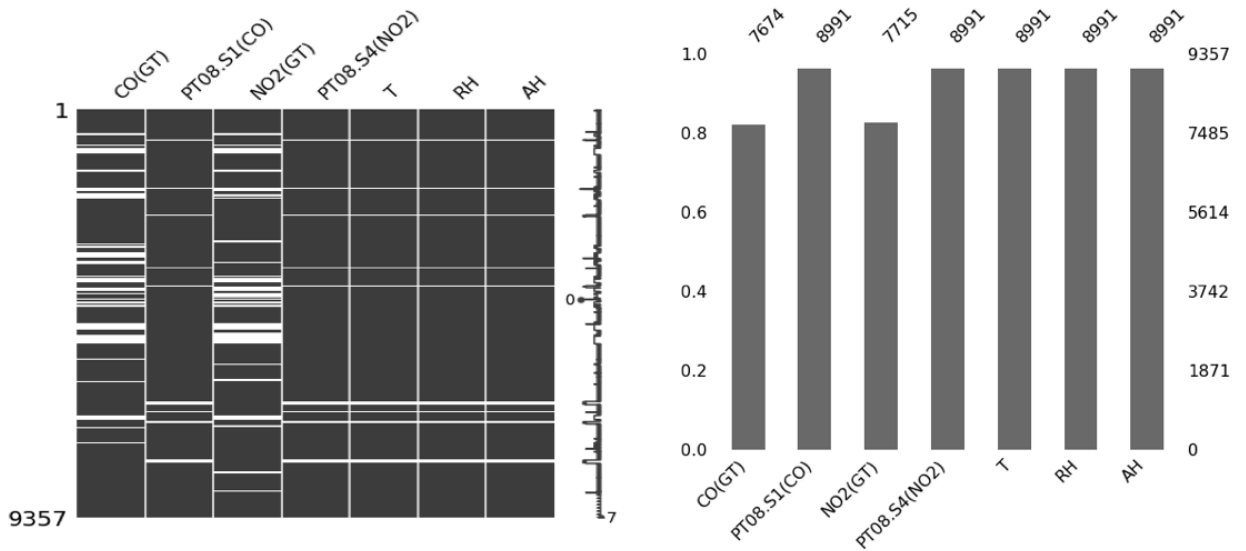


Figure 24: The first figure is showing that most of the missing values appear in in the reference sensors, and not the MOX sensor. The second figure shows the total number of measurements for each of the features and targets.

Figure 24 illustrates that most of the missing value come from the reference sensor for both NO_2 and CO measurements and not the MOX sensor. After performing a counting, 2416 values are removed. 7674 CO measurements and 7715 NO_2 measurements remain for the classification.

Before forwarding the data into the classification and regression task, all values were scaled to prevent dominating features and the features being in the same range as the target.

5.3 CLASSIFICATION

In the classification step grid search on multiple classifiers was performed. The models were ranked after the highest mean score. Classification step was performed several times with different parameter grids, the results presented is from the run with highest scores. The excel file with all models is provided in Appendix B. In Table 6 the best model for each classifier is presented.

Table 6: Shows the best model from each classifier with its parameter values and scores.

Classifier	Parameters:	Max_score	Mean score	Standard score
Multi-layer Perceptron	Activation=tanh Solver=adam Learning_rate_init=0.01 Hidden_layer_size= 8	0.782	0.777	0.0032
Random Forest	Maxdepth=8 Criterion=entropy n_estimators= 200	0.772	0.768	0.0042
SVC	C=0.005 Kernel= linear	0.764	0.762	0.0013
AdaBoost	n_estimators = 32	0.759	0.757	0.00082
Gradient Boosting	Learningrate=0.005 n_estimators = 100	0.757	0.757	0.0011
ExtraTree Classifier	n_estimators =100	0.738	0.732	0.0056

According to Table 6, the MLP Classifier performed be with a mean accuracy of 78% while the extra tree classifier performed worst with 73%. It should be noted that the MLP classifier takes longer time to compute.

In Figure 25 a confusion matrix was implemented to identify how the misclassification were distributed. According to the matrix, 1380 measurements were correctly classified as CO and 702 of the CO measurements were classified as NO₂. For NO₂ 1661 measurements were correctly classified as NO₂, while 422 were wrongly classified as CO.

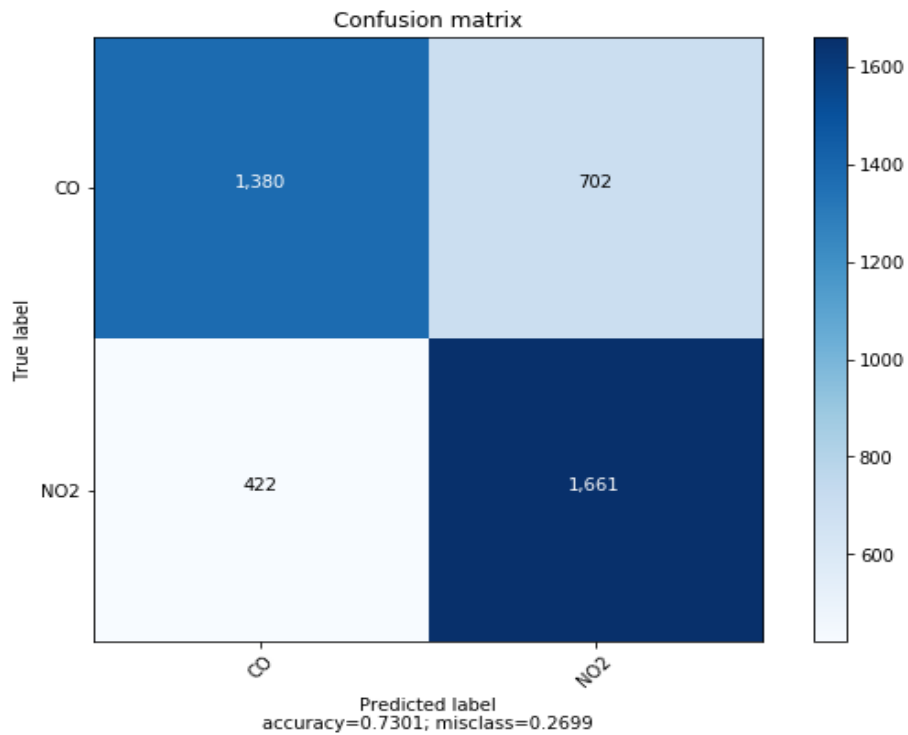


Figure 25: Shows how many measurements were correctly and wrongly classified.

5.4 REGRESSION

During the regression step a grid search for various parameters in a neural network was tested. The MSE loss function was used to rank models. The algorithm was run successfully multiple times. In Table 7 results from some models are shown for both NO₂ and CO. For a full overview of the results, the excel file with all the models are given in appendix C. The best model was later used on the test set and the R²-score was calculated.

Table 7: Present regression results and the models rank.

Rank of model	CO dataset		NO ₂ dataset	
	Validation MSE	Training MSE	Validation MSE	Training MSE
1	0.228	0.187	0.550	0.540
2	0.230	0.223	0.562	0.557
10	0.233	0.192	0.612	0.644
100	0.248	0.246	0.748	0.759

Table 7 shows a substantial difference in performance for the NO₂ and CO measurements. The CO measurements obtained a lower error with 0.205 compared to the NO₂ with 0.637. To analyse if the difference was due to the algorithm or the selective layer, a scatterplot of the MOX values against the reference sensor is computed in Figure 26.

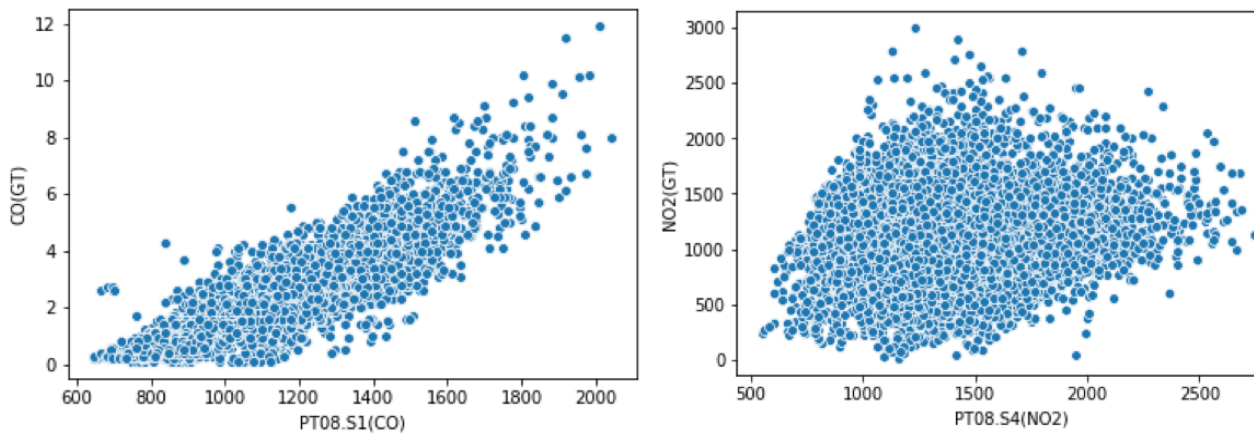


Figure 26: Scatterplot of raw output from the MOX value on the X axis and reference values on the Y axis.

The plot shows a clear linear relationship between the MOX value and the reference value for CO. However, between the MOX sensor and reference for the NO₂ measurements no linear relation is clear. In Figure 27 correlation plots were calculated to further investigate the relation between features and reference values.

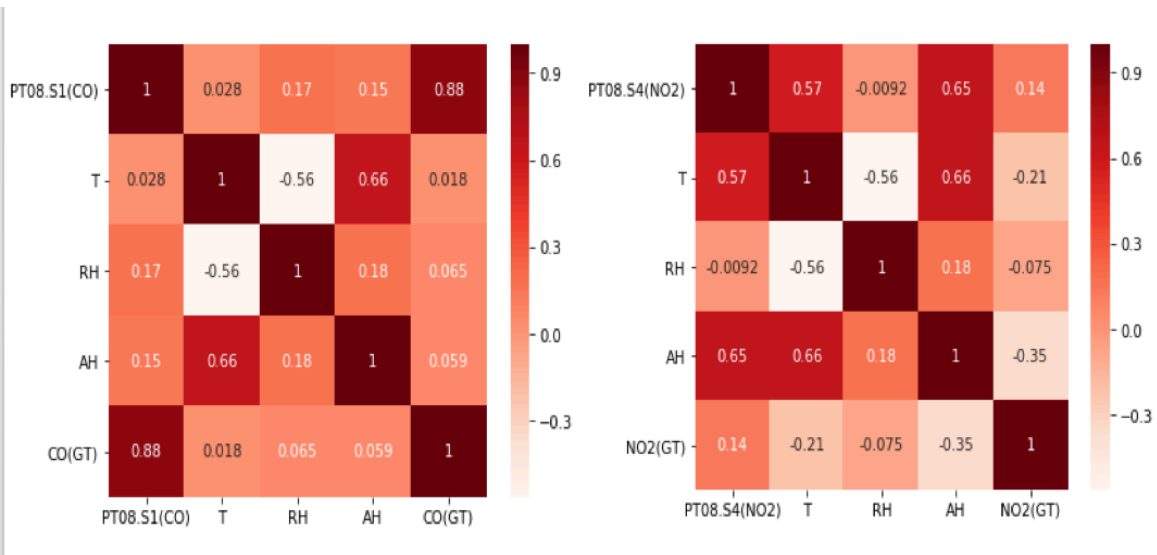


Figure 27: the correlation values between the features and reference sensor. CO(GT) and NO₂(GT) are the reference values while PT08.S4(NO₂) and PT08.S1(CO) are the MOX values.

From Figure 27 a strong positive correlation of 0.88 between the MOX value for CO and the reference value is shown. The correlation between MOX and the NO₂ reference is substantially smaller with a positive correlation of 0.14. However, for the NO₂ measurements there is a clear negative correlation between the temperature and absolute humidity features against the reference value of -0.21 and -0.35. This implies that for NO₂ measurements information about temperature and humidity can lead to important information about the reference values.

5.4.1 Comparing Auto-CMUT and commercial sensors

After training 216 models with different parameter values the best model was used to predict values based on the test-set. The predicted test values were compared to the target value by using the R^2 score. The results from the test set are shown in Table 8 together with the R^2 score for commercial sensors found in section 2.5.

Table 8: Compares the R^2 score of CO and NO_2 from the MOX data used in the Auto-CMUT against the R^2 score from the commercial sensors presented in section 2.5 The average R^2 score is also calculated.

Sensor	R^2 score CO	R^2 score NO_2	$R^2_{Average}$
Auto-CMUT	0.80	0.43	$\frac{0.80 + 0.43}{2} = 0.62$
AQ Mesh	0.80	0.46	$\frac{0.80 + 0.46}{2} = 0.62$
UniTec	0.43	0.62	$\frac{0.43 + 0.62}{2} = 0.53$
Air Quality Egg	0.0	0.03	$\frac{0.0 + 0.03}{2} = 0.015$
CairPol	0.94	0.12	$\frac{0.94 + 0.12}{2} = 0.53$
Vaisala	0.80	0.61	$\frac{0.80 + 0.61}{2} = 0.71$

The results from the Auto-CMUT combined with the MOX sensor studied in this thesis shows higher performance than the UniTec, Air Quality Egg and the CairPol sensors and the same as the AQ Mesh if looking at the average score. The sensor from Vaisala performs slightly better. It should also be noted that the Vaisala has a price of \$3700 and AQ Mesh \$10 000. It should also be noted that the MOX dataset forwarded into the Auto-CMUT was obtained over a period of 13 months, while the commercial sensor was tested for a 2 month period.

5.4.2 Change of R^2 due to regression

In the previous section predictions from the Auto-CMUT showed good results for both CO and NO₂ measurements compared to the commercial sensors. Furthermore, the results on how the regression part of the Auto-CMUT contributed to the R^2 score are presented. In Figure 28 the regression line between the scaled output from the MOX sensor against the reference is plotted for both CO and NO₂. The Figure shows a linear trend for the CO measurements, but with clear variance between many samples, the obtained R^2 -score is 0.53. For the NO₂ measurements no linear trend between the MOX value and reference value is visible, giving a R^2 score of 0.002. There are also several sample on the left that clearly deviates from the other measurements.

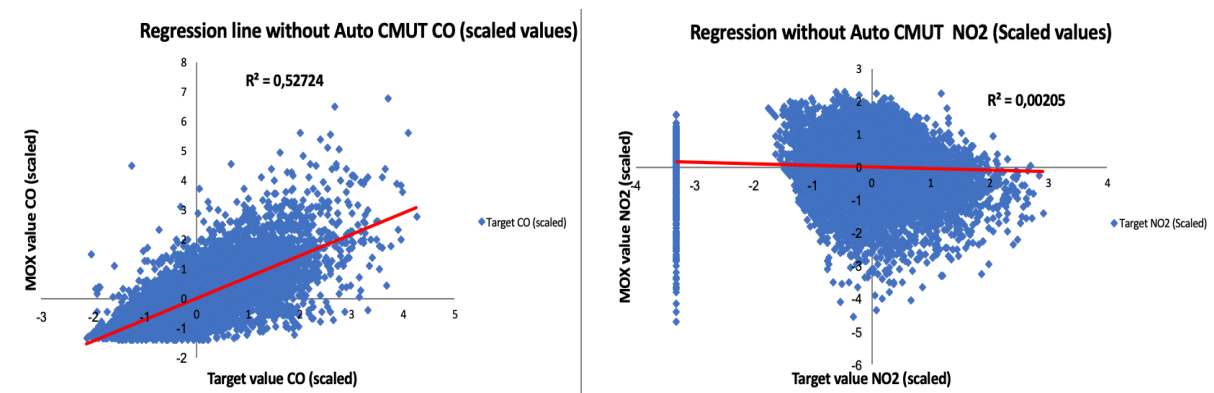


Figure 28: Regression line and R^2 score for CO and NO₂ measurements before the regression step of the Auto-CMUT.

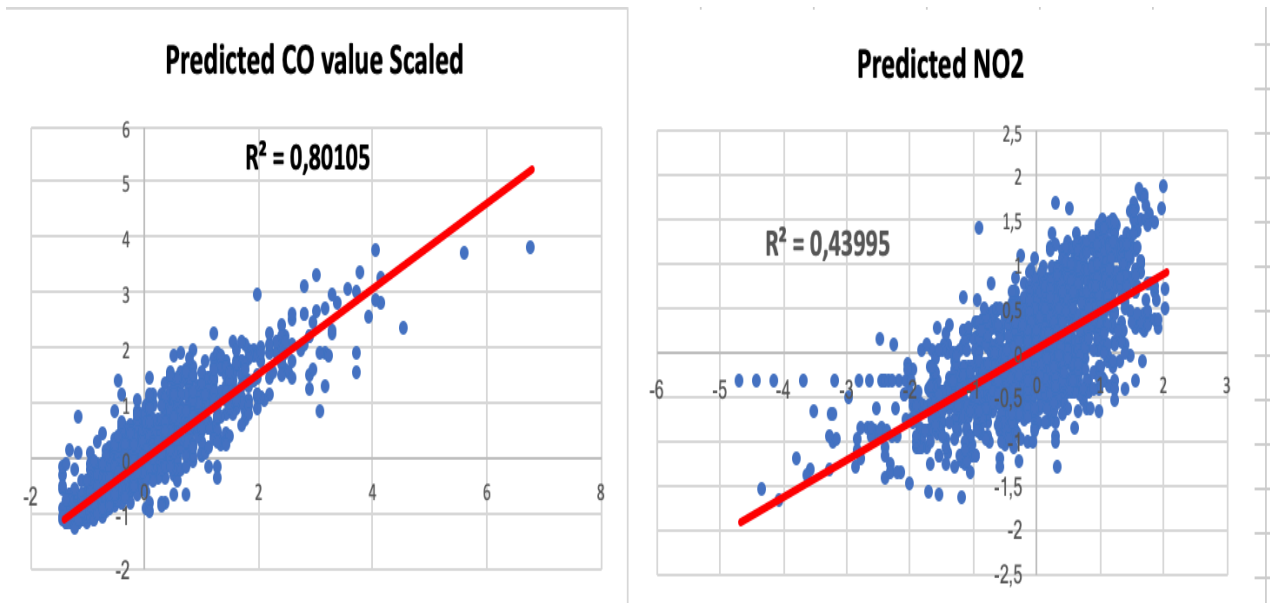


Figure 29 Shows the fitted regression line and R^2 score between the values predicted by the Auto-CMUT values and reference sensor.

Figure 29 shows the fitted regression line between the predicted values from the Auto-CMUT against the reference values. Compared to the regression lines plotted in Figure 26, the linear trend for both the CO and NO₂ is stronger. For CO, the R^2 score increased from 0.53 \rightarrow 0.80 and for NO₂ t from 0.002 \rightarrow 0.43. Applying the regression part of the Auto-CMUT clearly increased the R^2 score for both gases.

5.5 RESULT SUMMARY

To visualize the data pair plots, correlation plots and scatterplots were used to get an overview of the features and the relation between them before applying the Auto CMUT.

The Auto-CMUT was run successfully several times and results from the pre-processing, classification and regression was obtained. Visualizing missing values showed that the lack in measurements mainly originated from the reference sensor and not the MOX. The classification task obtained a max accuracy of 78% percent with the Multilayer Perceptron. Appendix B showed that MLP and Random Forest dominates among the models with highest performance. However, 60 models obtained an accuracy between 74% – 78%.

The regression task showed a clear difference in results for the CO and NO₂ measurements. To further investigate the reason for this difference scatter plots and correlation plots were calculated. The plots showed a clear linear relation between MOX values and reference value for the CO measurements and a more chaotic relation when looking at the NO₂ relation. Correlation plots validated the finding from the scatterplots and showed a substantial stronger correlation between the MOX value and reference value for CO measurements compared to the NO₂ measurements. It should be noted that temperature and humidity had significant correlation values against the NO₂ measurements. The R²-score for the MOX data increased from 0.002 to 0.43 for the NO₂ and from 0.053 to 0.80 for the CO after running the regression part of the Auto-CMUT compared to the R²-score for the scaled values. The neural network model therefore strongly contributed to an increase in the R²-score. With an R²-score for 0.43 and 0.80 the results from the Auto-CMUT is better than the Air Quality Egg, CairPol and Unitec sensor presented in section 2.5 In addition, the sensors yield similar results as more expensive sensors like Vaisala and AQ Mesh sensors.

6 DISCUSSION

During this thesis, an algorithm for dealing with output from chemical micro sensors called Auto-CMUT was developed. Due to lab delays no data from the CMUT sensor was obtained and the algorithm was therefore tested on data from a MOX sensor. The relevance of testing the Auto-CMUT on data obtained from the MOX sensor and its results are discussed. Evaluating the algorithm developed in the thesis must be discussed due to lack of universal and approved standards in machine learning. A discussion of the CMUTs relevance and commercial possibilities are addressed based on the research done in part one of the thesis. At last, thoughts about the projects challenges and the thesis limitations are highlighted.

6.1 COMPARING CMUT TO OTHER E-NOSE TECHNOLOGIES

The technology comparison in the first part of the thesis compared the CMUT to other sensor technologies. Based on Table 4 the MOX stands out as the biggest competitor for the CMUT sensor. The MOX technology scores high on most E-nose requirements except sensitivity and selectivity. However, based on the literature study, development of selective layers with high sensitivity is a challenge for all E-nose technologies. The MOX sensors biggest advantage over the CMUT technology is the amount of research and number of sensors already on the commercial market.

The CMUT has documented excellent sensitivities. If a company can produce a CMUT product with a competing selectivity at a slightly lower price it is likely that the CMUT can take a substantial part of the commercial E-nose market. To obtain quality assurance, the CMUT should also undergo an objective evaluation from AQ-SPEC as described in section 2.5

Many sensors available are sold without software that can translate the information from the sensor into meaningful information. The CMUT will be a more interesting product for all types of buyers if sold with software to answer the questions of which gases that are present and their concentration. In this thesis results show that the R^2 score increased substantially with the use of the Auto-CMUT.

6.2 COMMERCIAL SENSORS

Findings in section 2.6 give an objective evaluation of several commercial sensors on the market. The results showed big variation in performance between different commercial sensors. Both EPA and AQ-SPEC evaluate sensors based on the R^2 -score between the tested sensors output and the reference value. Based on these findings the R^2 -score should be used to evaluate the final performance of the Auto-CMUT. It should be noted that the AQ Mesh with a price of \$10 000 obtained the same overall R^2 -score as the predicted values from the Auto-CMUT. If the CMUT sensor succeeds in obtaining similar selectivity as the MOX sensor, it is likely that the CMUT will obtain similar results with the Auto-CMUT due to the similarities between the two sensors.

6.3 DATA SCIENTIST IN A BOX

Future smart cities will generate a vast amount of data and a substantial part of this data will come from sensors. For most people this data is nothing more than an overwhelming chaos of numbers. Implementing strategies and systems that can deal with and translate the information is crucial.

The Auto-CMUT algorithm developed in this thesis aims to translate data obtained from chemical micro sensors into information the average person can understand. The algorithm translates information about frequency shifts, temperature and humidity to answer two questions; which gasses are present in the air right now and what is the concentration of these gases. By answering these questions the algorithm and sensor together work like a specialized data scientist. In time the Auto-CMUT can reduce labour cost and number of human induced errors.

6.4 LACK OF CMUT DATA

This thesis developed an algorithm for the CMUT sensor. Due to unexpected delays in lab, no data from the CMUT sensor was obtained. Instead, results were obtained by testing the Auto-CMUT on the Air Quality dataset obtained from a MOX sensor. The Air Quality dataset has several similarities with the CMUT sensor. The main difference between the MOX sensor and CMUT is that the MOX gives values representing conductivity shift and the CMUT frequency shifts. Machine Learning only care about finding relation between features and does not care if features represent frequency or conductivity. Therefore, the algorithm is likely to give similar results when applied to the CMUT sensor if it obtains a similar selectivity.

In addition, results from the MOX sensor give an indication of competing technologies performance. These results can be used as a benchmark for the CMUT. The aim of the Auto-CMUT is to be specialized for the CMUT, but also could be used on similar technologies. By testing the algorithm on the Air Quality dataset, the generalization aspect of the algorithm is shown and a Proof of Concept (PoC) for the algorithm is conducted. Only minor details must be applied in the beginning of the algorithm in order to use the algorithm for data from the CMUT.

6.5 DATA QUALITY

The algorithm is important for obtaining high performances, but in the end a good algorithm cannot perform well if the data quality is neglected. Research has also been done to define a benchmark to define high data quality. A common definition is [111]:

“Data is of high quality when it satisfies the requirements of its intended use”

When using this definition for high data quality the argument for finding a universal benchmark model for data quality disappears, as different data in most cases are intended for different use. This fact can be combined this fact with a common expression in machine learning: “A model can only be as good as the data it is trained on”. Better data will in other words yield better models. Instead of focusing on defining a general score for data quality, the focus should therefore be to always aim towards better data quality for your specific problem. In machine learning, there are several methods and strategies to boost the quality of data:

- 1) **Knowing the data:** Storing data with unnecessary information leads to noise and is computationally expensive. Being aware of the content in the data is important.
- 2) **Automatically validation:** Validating data includes dealing with missing values and removing unnecessary data automatically.
- 3) **Data from a sensor will only be as good as the reference data it is evaluated towards.** Sticking a micro sensor on a stationary gas monitor with high accuracy will yield good data. Comparing the data to another low-cost sensor can prove to be useless.
- 4) The more data the algorithm deals with, the better. Therefore, testing the sensor in different applications and increase the number of measurements should be prioritized.

In machine learning a common misconception is that machine learning is a magical box that solves all problems. This misconception rarely discusses the quality of the data, but simply focuses on the complexity of the algorithm. The truth is that the quality is of great importance, the best algorithms in the world cannot give useful information if the data forwarded to the

algorithm is useless. The fact that the CO measurements obtained higher results than the NO₂ was clearly due to a difference in data quality.

6.6 MULTIPLE CLASSIFIER

The Auto-CMUT was successfully applied to the Air Quality dataset. Multiple classifiers with a wide range of parameter values are tested and the results are converted to an excel file. For the classification task the algorithm managed to correctly classify 78% of the CO and NO₂ measurements. The misclassification may be due to several things:

- 1) The MOX sensor may utilize similar values for the CO and NO₂ measurements, making it hard for any algorithm to separate the two gasses.
- 2) The reference used as the target value is not of high enough quality and therefore provide an inaccurate target.
- 3) The number of measurements are too low. The algorithm still provides high accuracy, but with further work on obtaining several measurements and with a quality assured reference the accuracy is likely to increase.

It should also be mentioned that 60 models obtained a mean accuracy between 74 -75,6%. Therefore, many classifiers with different parameter values can obtain high results. Appendix B shows that Random Forest classifier and the Multilayer Perceptron on average obtains higher performances on this dataset than other classifiers. To avoid error propagation in the regression step it is recommended that the classification step and regression step are run individually until a classification accuracy of 90% is reached. If over 10% of the measurements are classified wrong regression step will likely get confused by the misclassified measurements and result in a lower performance.

6.7 REGRESSION

During the regression step a grid search for multiple parameters in neural networks was successfully performed. The CMUT automatically provides an excel file with loss function values along with values for all parameters in each model. Table 7 and Appendix shows that Auto-CMUT performs substantially better on the CO measurements with a validation loss of 0.22 compared to a validation loss of 0.77 for the NO₂ measurements. For R² -score CO obtained a score of 0.80 while the NO₂ had a score of 0.43. Based on the correlation plots there is a higher correlation between the CO measurements and the reference than for the NO₂ measurements. Therefore it is likely that the difference between the NO₂ and CO measurements are due to a less selective layer for the NO₂ compared to the CO layer. In order to increase the score for NO₂ the focus should be on the selective layer rather than tuning the Auto-CMUT. The results from the Auto-CMUT also show that the MOX sensor combined with the Auto-CMUT obtain good results compared to a selection of commercial sensor.

6.8 PRE-TRAINING CMUT FOR DIFFERENT APPLICATIONS

Based on the work in this thesis a two-step training process for the CMUT is suggested. To obtain high performance the CMUTs should be sold for pre-trained applications. For instance, if the end-user intends to use the sensor in a greenhouse, the algorithm would likely perform better if pre-trained on a dataset obtained from a greenhouse and not an office. During pre-training the performance and not the computation time should be in focus, allowing the algorithm to run many models with some complexity. The Auto-CMUT developed is intended for step one in the suggested two-step training process.

Step two in the training process should be an embedded solution. The embedded solution should provide a continuous adjustment in parameters when used by the buyer. This step will perform the fine-tuning of the algorithm. The sensor will then be tuned for that specific location. For the embedded solution, the ranking criteria should be implemented changed: Simpler models with lower computational time should be preferred compared to the pre-training algorithm.

6.9 NO FREE LUNCH THEOREM

No free lunch theorem states that no model will perform perfectly on all problems and all datasets. The results in this thesis confirm this theorem. Both the classification and regression step of the Auto-CMUT was run several times, and the ranking among the possible models varied. It should be emphasized that these variations appeared when applying the algorithm on the same dataset.

The various rankings show that an algorithm with several simple models should be preferred over one complex algorithm. In addition, it is likely that different models will perform well on different applications (office, greenhouse, etc). Complex models will also likely suffer from overfitting as the algorithm starts finding patterns that only exist between a small subset of the measurements.

6.10 CHALLENGES AND LIMITATIONS

No CMUT measurements: Due to delays at Fraunhofer no data from the CMUT was obtained and the Auto-CMUT was not tested on CMUT data. It is therefore hard to estimate how well the algorithm will perform on data from the CMUT compared to the MOX sensor. Still, the only difference between the sensors will be that CMUT measures frequency change and MOX measures change in conductivity, the number of features will be the same.

Few features: The output from both the Air Quality dataset and the CMUT sensor will give few features to work with. Each CMUT array will measure the frequency shift for up to 7 different sensing layers in addition to temperature and humidity. This leads to one frequency value per gas in addition to temperature and humidity. To avoid overfitting this limits the depth and complexity of the NN's.

Few samples: The Air Quality dataset contains a limited number of measurements. As mentioned in the thesis the more measurements the better the model.

Chemical degradation: In literature the drawbacks of chemical degradation in the sensing layer of E-noses is often mentioned. Therefore, the CMUT should be tested over a longer period, preferably a year. Furthermore, work should be put in to analyse the effect of chemical degradation, and an algorithm to compensate for sensor drift should be implemented.

No standards for micro sensors: As there are no standard criteria for testing micro sensors it will be hard to compare the CMUT to competing technologies. In the future, a standard for testing micro sensor may be a required. It is hard to know if the CMUT will pass such a standard test. In this thesis, the CMUT has been compared to recommended guidelines from EPA.

7 CONCLUSION

7.1 RECOMMENDED FUTURE WORK

7.1.1 Compensating for drift in CMUT

Developing an embedded algorithm for real-time compensation for first-order drift should be implemented. This work should be based on data collected over a long period, preferably one year.

7.1.2 Remove outliers

Based on the regression plots in Figure 28 and Figure 29 the plots indicate some outliers for both the NO₂ and CO. Due to lack of time outlier removal was not implemented in this thesis. Removal of these outliers might lead to a slight increase in the R²-score.

7.1.3 Predefined applications for different CMUT configurations

The CMUT product should be recommended for specific applications. Giving the buyer the possibility to order a CMUT specialized for a specific application. Suggested applications are gas measurement inside homes/offices, outside and greenhouses. Before shipping the product to the buyer, the CMUT is recommended trained on a large dataset from its chosen application. Example, greenhouse CMUTs are trained on dataset obtained from greenhouse

Specializing the CMUT on specific applications will lead to a higher performance as it limits the expected range of values from the CMUT. As the CMUT is expected to have a low cost it would be possible to buy one for a greenhouse and one for measuring gas concentrations outside.

7.1.4 Field test inside and outside lab

The CMUT is suggested to be tested over a longer period inside a controlled environment. Field-test in lab will indicate how the sensor will perform under ideal conditions.

To test the CMUT in a real-life application the CMUT should be tested outside the lab. In the finalized commercial product, the CMUT should be trained on data obtained in a similar application and environment as the buyer intend to use it in.

By applying the sensor in a similar environment, it will recognize the patterns faster and will obtain a higher accuracy compared to a sensor only trained on an idealized dataset.

Testing the sensor in a real-life application over a longer period will also give indications of how the electronics and chemical sensing layer perform over time.

7.1.5 Embedded Machine Learning

The possibility of embedding a machine learning algorithm into the device itself should be researched. For future use, it is suggested that all CMUT are trained for a specialized application on a larger dataset and more time-consuming script. When shipped to the buyer the CMUTs also include an embedded algorithm. The embedded algorithm should have the ability to further tune the algorithm to aim for a higher performance.

7.1.6 Managing data from a network of sensors

Due to the low price of the CMUT's it is likely that some buyers will buy a network of sensor. Organizing the data from a network of sensor and logging the information in an organized and effective manner is crucial.

Extracting and combining data from many sensors is a complex task. Having a strategy and software to extract and combine information from a network of sensors would be positive purchase argument compared to others low-cost sensors. Further work on this could be done through a master thesis.

7.2 CONCLUSION SUMMARY

In the first part of this thesis a comparison between the CMUT technology and other E-nose technologies for use in portable E-nose applications was conducted. Based on the comparison in this thesis the CMUT obtained the highest average ranking along with the MOX technology. If the CMUT sensor succeeds in obtaining a competing lifetime and a slightly better selectivity to a lower price than the MOX sensors, the CMUT is likely to take a substantial part of the commercial E-nose market. Results from fieldtests performed by AQ-SPEC on a selection of commercial E-noses was also presented.

In the second part of the thesis an algorithm for handling output from chemical sensors was developed, the Auto-CMUT. Due to lack of data from the CMUT a similar dataset from a MOX sensor was forwarded to the algorithm. Due to the similarities between dataset from MOX and CMUT, only minor changes must be applied to the code before using the Auto-CMUT on CMUT data. The algorithm removes missing values and translates information about humidity, temperature, frequency or conductivity shifts to identify and quantify the concentration of different gasses. The results from the Auto-CMUT gave competing results with the commercial E-nose sensors evaluated in the thesis. It was also showed that the regression step of the Auto-CMUT increased the R^2 -score between the MOX output and the reference sensors with 27% for CO and 43% for NO₂.

BIBLIOGRAPHY

1. Mahmood, Z., *Smart Cities : Development and Governance Frameworks*. 2018, Springer International Publishing : Imprint: Springer: Cham. Available from: <https://link.springer.com/book/10.1007%2F978-3-319-76669-0>.
2. Mølgaard, M.J., *Capacitive Micromachined Ultrasonic Transducers for Gas Sensing*. 2018. Available from: [http://orbit.dtu.dk/portal/en/publications/id\(516ff999-3001-4a0a-88d3-5b53897926f8\).html](http://orbit.dtu.dk/portal/en/publications/id(516ff999-3001-4a0a-88d3-5b53897926f8).html).
3. Albino, V., U. Berardi, and R.M. Dangelico, *Smart cities: Definitions, dimensions, performance, and initiatives*. *Journal of urban technology*, 2015. **22**(1): p. 3-21. Available from: <https://www.tandfonline.com/doi/abs/10.1080/10630732.2014.942092>.
4. Vasilev, I.S., Daniel; Gianmario, Spacagna; Roelants, Peter; Zocca, Valentino, *Python Deep Learning* Second ed. 2019, Birmingham: Packt Publishing. 373.
5. Ghosal, A. and S. Halder, *Smart Cities : Development and Governance Frameworks*, Z. Mahmood, Editor. 2018, Springer International Publishing : Imprint: Springer: Cham. p. 107-125. Available from: <https://link.springer.com/book/10.1007%2F978-3-319-76669-0>.
6. WHO. *Ambient (outdoor) air quality and health*. 2018; Available from: [https://www.who.int/en/news-room/fact-sheets/detail/ambient-\(outdoor\)-air-quality-and-health](https://www.who.int/en/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health).
7. Castell, N., F.R. Dauge, P. Schneider, M. Vogt, U. Lerner, B. Fishbain, D. Broday, and A. Bartonova, *Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?* *Environment International*, 2017. **99**: p. 293 - 302. Available from: <http://www.sciencedirect.com/science/article/pii/S0160412016309989>.
8. Jordan, M.I. and T.M. Mitchell, *Machine learning: Trends, perspectives, and prospects*. *Science*, 2015. **349**(6245): p. 255-260. Available from: <https://science.sciencemag.org/content/349/6245/255>.
9. Fluenta. *About Fluenta* 2019 [cited 2019 5 May]; Available from: <http://fluenta.com/about/>.
10. Landrigan, P.J., *Air pollution and health*. *The Lancet Public Health*, 2017. **2**(1): p. e4-e5. Available from: [https://www.thelancet.com/journals/lanpub/article/PIIS2468-2667\(16\)30023-8/fulltext](https://www.thelancet.com/journals/lanpub/article/PIIS2468-2667(16)30023-8/fulltext).
11. NIPH, N.i.f.p.h., *Air pollution in Norway* 2017: <https://www.fhi.no/en/op/hin/environment/air-pollution-in-norway---public-he/>. Available from: <https://www.fhi.no/en/op/hin/environment/air-pollution-in-norway---public-he/>.
12. WHO *Air Quality Guidelines Global Update Particulate matter, ozone, nitrogen dioxide and sulfur dioxide*. 2005. **Available from:** http://www.euro.who.int/_data/assets/pdf_file/0005/78638/E90038.pdf?ua=1.
13. Carvalho, V., J.G. Lopes, H.G. Ramos, and F.C. Alegria. *City-wide mobile air quality measurement system*. in *SENSORS, 2009 IEEE*. 2009. IEEE. Available from: <https://ieeexplore.ieee.org/abstract/document/5398299>.
14. Folkehelseinstituttet, *Luftkvalitetskriterier: Virkninger av luftforurensning på helse*. 2013. Available from: <https://www.fhi.no/globalassets/dokumenterfiler/rapporter/2013/luftkvalitetskriterier---virkninger-av-luftforurensning-pa-helse-pdf.pdf>.

15. Raub, J.A., M. Mathieu-Nolf, N.B. Hampson, and S.R. Thom, *Carbon monoxide poisoning—a public health perspective*. Toxicology, 2000. **145**(1): p. 1-14. Available from: <https://www.sciencedirect.com/science/article/pii/S0300483X99002176>.
16. Organization, W.H., *WHO guidelines for indoor air quality: selected pollutants*. 2010. Available from: <https://apps.who.int/iris/handle/10665/260127>.
17. Olivier, J.G., K. Schure, and J. Peters, *Trends in global CO2 and total greenhouse gas emissions*. PBL Netherlands Environmental Assessment Agency, 2017: p. 5.
18. Persaud, K. and G. Dodd, *Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose*. Nature, 1982. **299**(5881): p. 352.
19. Gardner, J.W. and P.N. Bartlett, *A brief history of electronic noses*. Sensors and Actuators B: Chemical, 1994. **18**(1-3): p. 210-211.
20. Röck, F., N. Barsan, and U. Weimar, *Electronic nose: current status and future trends*. Chemical reviews, 2008. **108**(2): p. 705-725. Available from: <https://pubs.acs.org/doi/pdf/10.1021/cr068121q>.
21. Wilson, A. and M. Baietto, *Applications and advances in electronic-nose technologies*. Sensors, 2009. **9**(7): p. 5099-5148. Available from: <https://www.mdpi.com/1424-8220/9/7/5099>.
22. Strle, D., B. Štefane, M. Trifkovič, M. Van Miden, I. Kvasić, E. Zupanič, and I. Muševič, *Chemical selectivity and sensitivity of a 16-channel electronic nose for trace vapour detection*. Sensors, 2017. **17**(12): p. 2845. Available from: <https://www.mdpi.com/1424-8220/17/12/2845>.
23. Laref, R., E. Losson, A. Sava, and M. Siadat, *Calibrating chemical multisensory devices for real world applications: An in-depth comparison of quantitative machine learning approaches*. Sensors and Actuators B: Chemical, 2018. **255**: p. 1191 - 1210. Available from: <http://www.sciencedirect.com/science/article/pii/S0925400517313692>.
24. Peris, M. and L. Escuder-Gilabert, *A 21st century technique for food control: Electronic noses*. Analytica Chimica Acta, 2009. **638**(1): p. 1 - 15. Available from: <http://www.sciencedirect.com/science/article/pii/S0003267009002268>.
25. Chen, J., J. Gu, R. Zhang, Y. Mao, and S. Tian, *Freshness Evaluation of Three Kinds of Meats Based on the Electronic Nose*. Sensors, 2019. **19**(3): p. 605.
26. Gasparri, R., G. Sedda, and L. Spaggiari, *The Electronic Nose's Emerging Role in Respiratory Medicine*. Sensors, 2018. **18**(9): p. 3029.
27. Thaler, E.R. and C.W. Hanson, *Medical applications of electronic nose technology*. Expert review of medical devices, 2005. **2**(5): p. 559-566.
28. Wilson, A., *Diverse applications of electronic-nose technologies in agriculture and forestry*. Sensors, 2013. **13**(2): p. 2295-2348.
29. Castell, N., M. Viana, M.C. Minguillón, C. Guerreiro, and X. Querol, *Real-world application of new sensor technologies for air quality monitoring*. ETC/ACM Technical Paper, 2013. **16**.
30. Castell, N., F.R. Dauge, P. Schneider, M. Vogt, U. Lerner, B. Fishbain, D. Broday, and A. Bartonova, *Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates?* Environment international, 2017. **99**: p. 293-302. Available from: <https://www.sciencedirect.com/science/article/pii/S0160412016309989>.

31. Snyder, E.G., T.H. Watkins, P.A. Solomon, E.D. Thoma, R.W. Williams, G.S.W. Hagler, D. Shelow, D.A. Hindin, V.J. Kilaru, and P.W. Preuss, *The Changing Paradigm of Air Pollution Monitoring*. Environmental Science & Technology, 2013. **47**(20): p. 11369-11377.
32. Gerboles, M. *Developments and Applications of Sensor Technologies for Ambient Air Monitoring*. in *Workshop "Current and Future Air Quality Monitoring"*, Barcelona. 2012.
33. Williams, R., V. Kilaru, E. Snyder, A. Kaufman, T. Dye, A. Rutter, A. Russell, and H. Hafner, *Air sensor guidebook*. US Environmental Protection Agency, 2014.
34. McGill, R.A., V.K. Nguyen, R. Chung, R.E. Shaffer, D. DiLella, J.L. Stepnowski, T.E. Mlsna, D.L. Venezky, and D. Dominguez, *The "NRL-SAWRHINO": A nose for toxic gases*. Sensors and Actuators B: Chemical, 2000. **65**(1-3): p. 10-13. Available from: <https://www.sciencedirect.com/science/article/pii/S0925400599003524>.
35. Fanget, S., S. Hentz, P. Puget, J. Arcamone, M. Matheron, E. Colinet, P. Andreucci, L. Duraffourg, E. Myers, and M. Roukes, *Gas sensors based on gravimetric detection—A review*. Sensors and Actuators B: Chemical, 2011. **160**(1): p. 804-821. Available from: <https://www.sciencedirect.com/science/article/pii/S0925400511007891>.
36. Grzymala-Busse, J.W. and M. Hu. https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/MVDM/grzymala_busse_hu01.pdf. in *International Conference on Rough Sets and Current Trends in Computing*. 2000. Springer.
37. Wang, C., L. Yin, L. Zhang, D. Xiang, and R. Gao, *Metal oxide gas sensors: sensitivity and influencing factors*. Sensors, 2010. **10**(3): p. 2088-2106. Available from: <https://www.mdpi.com/1424-8220/10/3/2088/htm>.
38. Eranna, G., *Metal oxide nanostructures as gas sensing devices*. 2016: CRC Press. Available from: <https://www.taylorfrancis.com/books/9780429106606>.
39. M. Voinova, M.J., *Chemical sensors : comprehensive sensors technologies. : Volume 4, : Solid-state devices*. 1st ed. Sensors technology series. Vol. Volume 4,. 2011.
40. Korotcenkov, G., *Metal oxides for solid-state gas sensors: What determines our choice?* Materials Science and Engineering: B, 2007. **139**(1): p. 1 - 23. Available from: <http://www.sciencedirect.com/science/article/pii/S0921510707000700>.
41. Teramura, Y. and M. Takai, *Quartz Crystal Microbalance*. 2018: p. 509-520. Available from: https://www.researchgate.net/publication/323263269_Quartz_Crystal_Microbalance.
42. Arshak, K., E. Moore, G. Lyons, J. Harris, and S. Clifford, *A review of gas sensors employed in electronic nose applications*. Sensor review, 2004. **24**(2): p. 181-198. Available from: <https://www.emeraldinsight.com/doi/abs/10.1108/02602280410525977>.
43. Haller, M.I. and B.T. Khuri-Yakub, *A surface micromachined electrostatic ultrasonic air transducer*. IEEE transactions on ultrasonics, ferroelectrics, and frequency control, 1996. **43**(1): p. 1-6. Available from: <https://ieeexplore.ieee.org/abstract/document/484456>.
44. Park, K., H.J. Lee, G. Yaralioglu, A. Ergun, Ö. Oralkan, M. Kupnik, C. Quate, B. Khuri-Yakub, T. Braun, and J.-P. Ramseyer, *Capacitive micromachined ultrasonic transducers for chemical detection in nitrogen*. Applied Physics Letters, 2007. **91**(9): p. 094102. Available from: <https://www.sciencedirect.com/science/article/pii/S0925400506006204>.

45. Lee, H.J., K.K. Park, O. Oralkan, M. Kupnik, and B.T. Khuri-Yakub. *CMUT as a chemical sensor for DMMP detection*. in *2008 IEEE international frequency control symposium*. 2008. IEEE. Available from: <https://ieeexplore.ieee.org/abstract/document/4623034>.
46. Lee, H.J., K.K. Park, M. Kupnik, O. Oralkan, and B.T. Khuri-Yakub, *Chemical vapor detection using a capacitive micromachined ultrasonic transducer*. *Analytical chemistry*, 2011. **83**(24): p. 9314-9320. Available from: https://www.researchgate.net/publication/224405911_Capacitive_micromachined_ultrasonic_transducers_for_chemical_detection_in_nitrogen.
47. Microsystems, F.I.f.P., *Micromachined Ultrasonic Transponders in Post-CMOS Technology*. 2019. Available from: <https://www.ipms.fraunhofer.de/en/research-development/cmud.html>.
48. Ergun, A.S., G.G. Yaralioglu, and B.T. Khuri-Yakub, *Capacitive micromachined ultrasonic transducers: Theory and technology*. *Journal of aerospace engineering*, 2003. **16**(2): p. 76-84. Available from: [https://ascelibrary.org/doi/abs/10.1061/\(ASCE\)0893-1321\(2003\)16:2\(76\)](https://ascelibrary.org/doi/abs/10.1061/(ASCE)0893-1321(2003)16:2(76)).
49. EPA. *Evaluation of Emerging Air Pollution Sensor Performance*. 2019 [cited 2019 18 april]; Available from: https://www.epa.gov/air-sensor-toolbox/evaluation-emerging-air-pollution-sensor-performance?fbclid=IwAR0OtdGs2vbthpgkZtTZ6HmUTFKJm_IRx05N0FwXCH_2rPz94rXl34YL_T8.
50. iScape. *About*. 2019 [cited 2019 4 May]; Available from: <https://www.iscapeproject.eu/about/>.
51. Polidori, A., B. Feenstra, V. Papapostolou, and H. Zhang, *Field Evaluation of Low-Cost Air Quality Sensors: Field Setup and Testing Protocol*. 2017, AQ-SPEC. Available from: <http://www.aqmd.gov/docs/default-source/aq-spec/protocols/sensors-field-testing-protocol.pdf?sfvrsn=0>.
52. Williams, R., R. Long, M. Beaver, A. Kaufman, F. Zeiger, M. Heimbinder, I. Hang, R. Yap, B. Acharya, B. Ginwald, K. Kupcho, S. Robinson, O. Zaouak, B. Aubert, M. Hannigan, R. Piedrahita, N. Masson, B. Moran, M. Rook, P. Heppner, C. Cogar, N. Nikzad, AND W. Griswold. U.S. , *Sensor Evaluation Report EPA/Environmental Protection Agency*, Editor. 2014: Washington, DC. Available from: https://cfpub.epa.gov/si/si_public_record_report.cfm?Lab=NERL&dirEntryId=277270&simpleSearch=1&searchAll=sensor+evaluation+report.
53. Papapostolou, V., H. Zhang, B. J.Feenstra, and A. Polidori, *Development of an environmental chamber for evaluating the performance of low-cost air quality sensors under controlled conditions*. *Atmospheric Environment*, 2017. **171**: p. 82 - 90. Available from: <http://www.sciencedirect.com/science/article/pii/S1352231017306647>.
54. Alexander, D., A. Tropsha, and D.A. Winkler, *Beware of R 2: simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models*. *Journal of chemical information and modeling*, 2015. **55**(7): p. 1316-1322.
55. Egg, A.Q. *The Egg*. 2019 [cited 2019 4 May]; Available from: <https://airqualityegg.com/egg>.
56. AQ-SPEC. *Air Quality Egg (Version 1)*. 2019 [cited 2019 4 May]; Available from: <http://www.aqmd.gov/aq-spec/sensordetail/air-quality-egg>.
57. AQ-SPEC.

- Vaisala - AQT410*. 2019 [cited 2019 4 May]; Available from: <http://www.aqmd.gov/aq-spec/sensordetail/vaisala---aqt410>.
58. AQ-SPEC. *AQMesh (v.4.0)*. 2019; Available from: [http://www.aqmd.gov/aq-spec/sensordetail/aqmesh-\(v.4.0\)](http://www.aqmd.gov/aq-spec/sensordetail/aqmesh-(v.4.0)).
59. AQ-SPEC. *CairPol Cairsens*. [cited 2019 4 May]; Available from: <http://www.aqmd.gov/aq-spec/sensordetail/cairpol-cairsens>.
60. AQ-SPEC. *Unitec - SENS-IT*. [cited 2019 4 May]; Available from: <http://www.aqmd.gov/aq-spec/sensordetail/unitec---sens-it>.
61. Raschka, S. and V. Mirjalili, *Python machine learning*. 2017: Packt Publishing Ltd.
62. Da Silva, I.N., D.H. Spatti, R.A. Flauzino, L.H.B. Liboni, and S.F. dos Reis Alves, *Artificial neural networks*. Cham: Springer International Publishing, 2017. Available from: <https://link.springer.com/content/pdf/10.1007/978-3-319-43162-8.pdf>.
63. Finn, E., *What algorithms want: Imagination in the age of computing*. 2017: MIT Press. Available from: https://books.google.no/books?hl=no&lr=&id=TwJHDgAAQBAJ&oi=fnd&pg=PP7&dq=What+algorithms+want:+Imagination+in+the+age+of+computing&ots=a4qQxmhemn&sig=Ae59_mW3yIBaxECNuCBSdQO_rAU&redir_esc=y_v=onepage&q=What%20algorithms%20want%3A%20Imagination%20in%20the%20age%20of%20computing&f=false.
64. Mohammed, M., M.B. Khan, and E.B.M. Bashier, *Machine learning: algorithms and applications*. 2016: Crc Press.
65. Boden, M.A., *AI: Its nature and future*. 2016: Oxford University Press. Available from: https://books.google.no/books?hl=no&lr=&id=yDQTDAAAQBAJ&oi=fnd&pg=PP1&dq=AI:+Its+nature+and+future&ots=T0ip2z-aJd&sig=6tW81aDcMmO6Xcl6bd3X297T96U&redir_esc=y_v=onepage&q=AI%3A%20Its%20nature%20and%20future&f=false.
66. Smith, A. and J. Anderson, *AI, Robotics, and the Future of Jobs*. Pew Research Center, 2014. 6. Available from: <http://www.fusbp.com/wp-content/uploads/2010/07/AI-and-Robotics-Impact-on-Future-Pew-Survey.pdf>.
67. Nag, A. and S.C. Mukhopadhyay, *Wearable Electronics Sensors : For Safe and Healthy Living*, in *Smart Sensors, Measurement and Instrumentation*, S.C. Mukhopadhyay, Editor. 2015. p. 1-35.
68. Cucker and Smale, *Best Choices for Regularization Parameters in Learning Theory: On the Bias—Variance Problem*. Foundations of Computational Mathematics, 2002. 2(4): p. 413-428. Available from: <https://doi.org/10.1007/s102080010030>.
69. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for optimization*. IEEE transactions on evolutionary computation, 1997. 1(1): p. 67-82. Available from: http://georgemaciunas.com/wp-content/uploads/2012/07/Wolpert_NLFoptimization-1.pdf.
70. Bengio, Y. and Y. Grandvalet, *No unbiased estimator of the variance of k-fold cross-validation*. Journal of machine learning research, 2004. 5(Sep): p. 1089-1105. Available from: <http://www.jmlr.org/papers/v5/grandvalet04a.html?92f58540>.
71. Hossin, M. and M. Sulaiman, *A review on evaluation metrics for data classification evaluations*. International Journal of Data Mining & Knowledge Management Process, 2015. 5(2): p. 1. Available from:

https://s3.amazonaws.com/academia.edu.documents/37219940/5215jldkp01.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1558358711&Signature=6222TQ7dHVqKAvEO5JhCqu5gE2Y%3D&response-content-disposition=inline%3B%20filename%3DA_REVIEW_ON_EVALUATION_METRICS_FOR_DATA.pdf

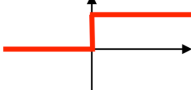
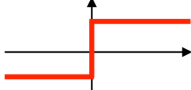
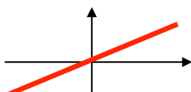
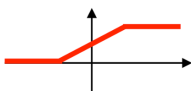
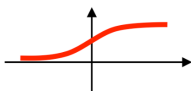
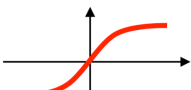
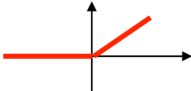

72. Patil, T.R. and S. Sherekar, *Performance analysis of Naive Bayes and J48 classification algorithm for data classification*. International journal of computer science and applications, 2013. 6(2): p. 256-261. Available from: <http://keddiyan.com/files/AHCI/week2/9.pdf>.
73. Feurer, M. and F. Hutter, *Automatic machine learning: methods, systems, challenges*, in *Challenges in Machine Learning*, F. Hutter, L. Kotthoff, and J. Vanschoren, Editors. 2019. p. 3-38.
74. Ho, Y.-C. and D.L. Pepyne, *Simple explanation of the no-free-lunch theorem and its implications*. Journal of optimization theory and applications, 2002. 115(3): p. 549-570. Available from: <https://link.springer.com/article/10.1023/A:1021251113462>.
75. Feurer, M., A. Klein, K. Eggenberger, J. Springenberg, M. Blum, and F. Hutter. *Efficient and robust automated machine learning*. in *Advances in neural information processing systems*. 2015. Available from: <http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf>.
76. Bergstra, J. and Y. Bengio, *Random search for hyper-parameter optimization*. Journal of Machine Learning Research, 2012. 13(Feb): p. 281-305. Available from: <http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>.
77. Elsken, T., J.H. Metzen, and F. Hutter, *Automatic machine learning: methods, systems, challenges*, in *Challenges in Machine Learning*, F. Hutter, L. Kotthoff, and J. Vanschoren, Editors. 2019. p. 69-86.
78. Gharamani, Z., *Automatic machine learning: methods, systems, challenges*, in *Challenges in Machine Learning*, F. Hutter, L. Kotthoff, and J. Vanschoren, Editors. 2019.
79. Kotsiantis, S., D. Kanellopoulos, and P. Pintelas, *Data preprocessing for supervised learning*. International Journal of Computer Science, 2006. 1(2): p. 111-117. Available from: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.8413&rep=rep1&type=pdf>.
80. Rahm, E. and H.H. Do, *Data cleaning: Problems and current approaches*. IEEE Data Eng. Bull., 2000. 23(4): p. 3-13. Available from: https://s3.amazonaws.com/academia.edu.documents/41858217/A00DEC-CD.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1558349414&Signature=xtaTRcgKfwdvHvuEg75gmaWA7C8%3D&response-content-disposition=inline%3B%20filename%3DAutomatically_extracting_structure_from.pdf_page=5.
81. Acock, A.C., *Working with missing values*. Journal of Marriage and family, 2005. 67(4): p. 1012-1028. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1741-3737.2005.00191.x>.
82. Ben-Gal, I., *Outlier detection*. Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers. 2005. Available from: https://link.springer.com/chapter/10.1007/0-387-25465-X_7.
83. Wolfson School of Mechanical and Manufacturing Engineering, *An introduction to MEMS*. 2002, Loughborough: PRIME Faraday Partnership. Available from: https://www.lboro.ac.uk/microsites/mechman/research/ipm-ktn/pdf/Technology_review/an-introduction-to-mems.pdf.
84. Meyer, D. and F.T. Wien, *Support vector machines*. The Interface to libsvm in package e1071, 2015: p. 28.

85. Freund, Y. and R.E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of computer and system sciences, 1997. **55**(1): p. 119-139. Available from: <https://www.sciencedirect.com/science/article/pii/S002200009791504X>.
86. Freund, Y., R. Schapire, and N. Abe, *A short introduction to boosting*. Journal-Japanese Society For Artificial Intelligence, 1999. **14**(771-780): p. 1612. Available from: <http://www.yorku.ca/gisweb/eats4400/boost.pdf>.
87. Hastie, T., S. Rosset, J. Zhu, and H. Zou, *Multi-class adaboost*. Statistics and its Interface, 2009. **2**(3): p. 349-360. Available from: https://www.intlpress.com/site/pub/files/_fulltext/journals/sii/2009/0002/0003/SII-2009-0002-0003-a008.pdf.
88. Krenker, A., J. Bester, and A. Kos, *Artificial neural networks: methodological advances and biomedical applications*, K. Suzuki, Editor. 2011, InTech: Croatia. p. 3-18.
89. Smith, L.S., *Neural Networks*, in *International Encyclopedia of Statistical Science*, M. Lovric, Editor. 2011, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 942-945.
90. Vasilev, I., D. Slater, S. Gianmario, P. Roelants, and V. Zocca, *Python Deep Learning 2019*, Packt: Birmingham, UK.
91. scikit-learn. *scikit-learn, Machine Learning in Python*. 2019 [cited 2019 5 May]; Available from: <https://scikit-learn.org/stable/>.
92. Pandas. *Python Data Analysis Library*. 2019 [cited 2019 5 May]; Available from: <https://pandas.pydata.org/>.
93. Numpy. *Numpy*. 2019 [cited 2019 5 May]; Available from: <http://www.numpy.org/>.
94. matplotlib. *matplotlib*. 2019 [cited 2019 5 May]; Available from: <https://matplotlib.org/>.
95. Keras. *Keras*. 2019 [cited 2019 5 May]; Available from: <https://keras.io/>.
96. Waskom, M. *seaborn: statistical data visualization*. 2019 [cited 2019 5 May]; Available from: <https://seaborn.pydata.org/>.
97. Kotila, M. *Talos user manual*. 2019 [cited 2019 5 May]; Available from: https://autonomio.github.io/docs_talos/-introduction.
98. scikit-learn. *3.2.4.3.1. sklearn.ensemble.RandomForestClassifier¶*
. 2019 [cited 2019 5 May]; Available from: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.
99. scikit-learn. *sklearn.svm.SVC*. 2019 [cited 2019 5 May]; Available from: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.
100. scikit-learn. *sklearn.linear_model.LogisticRegression*. 2019 [cited 2019 5 May]; Available from: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
101. scikit-learn. *sklearn.ensemble.AdaBoostClassifier*. 2019 [cited 2019 5 May]; Available from: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>.

102. scikit-learn. *sklearn.neural_network.MLPClassifier*. 2019 [cited 2019 5 May]; Available from: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.
103. McKinney, W. *Pandas*. 2010; Available from: <https://pandas.pydata.org/>.
104. kotila, M. *Autonomio Talos [Computer software]*. 2018; Available from: <http://github.com/autonomio/talos>.
105. Talos. *Talos: User Manual*. 2018; Available from: https://autonomio.github.io/docs_talos/-introduction.
106. Chollet, F. *Keras Documentation*. 2015 [cited 2019 15 April]; Available from: <https://keras.io/>.
107. Holmes, G., A. Donkin, and I.H. Witten, *Weka: A machine learning workbench*. 1994. Available from: <https://researchcommons.waikato.ac.nz/handle/10289/1138>.
108. Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, *The WEKA data mining software: an update*. ACM SIGKDD explorations newsletter, 2009. **11**(1): p. 10-18. Available from: <https://dl.acm.org/citation.cfm?id=1656278>.
109. Thornton, C., F. Hutter, H.H. Hoos, and K. Leyton-Brown. *Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms*. in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013. ACM. Available from: <https://dl.acm.org/citation.cfm?id=2487629>.
110. Kotthoff, L., C. Thornton, H.H. Hoos, F. Hutter, and K. Leyton-Brown, *Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA*. The Journal of Machine Learning Research, 2017. **18**(1): p. 826-830. Available from: <http://www.jmlr.org/papers/volume18/16-261/16-261.pdf>.
111. Ziemba, E., *Information Technology for Management. Ongoing Research and Development: 15th Conference, AITM 2017, and 12th Conference, ISM 2017, Held as Part of FedCSIS, Prague, Czech Republic, September 3-6, 2017, Extended Selected Papers*. Vol. 311. 2018: Springer.
112. Raschka, S., *Activation function figure* 2016. Available from: <https://sebastianraschka.com/blog/index.html>.
113. Engin. Pikel 2017 Available from : https://www.researchgate.net/publication/315111480_A_COMPREHENSIVE_REVIEW_FOR_ARTIFICIAL_NEURAL_NETWORK_K_APPLICATION_TO_PUBLIC_TRANSPORTATION/figures?lo=1

APPENDIX

APPENDIX A: OVERVIEW OF DIFFERENT ACTIVATION FUNCTIONS

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016
(<http://sebastianraschka.com>)

Figure 30: An overview of different activation functions commonly used in neural networks or classifiers. Obtained with permission from [112]

APPENDIX B: RESULTS FROM THE MULTIPLE CLASSIFIER IN THE AUTO-CMUT

Table 9: Results from the best run of multiple classifier. In the original excel file the values of different parameters are also included. Parameter values are excluded from appendix due to space.

Rank Nr	estimator	max_score	mean_score	min_score	std_score
1	MLPClassifier	0,782	0,777	0,774	0,00323
2	MLPClassifier	0,781	0,775	0,771	0,00395
3	MLPClassifier	0,779	0,774	0,770	0,00384
4	MLPClassifier	0,783	0,773	0,766	0,00697
5	MLPClassifier	0,778	0,772	0,768	0,00415
6	MLPClassifier	0,773	0,772	0,772	0,00068
7	MLPClassifier	0,774	0,772	0,768	0,00276
8	MLPClassifier	0,774	0,772	0,766	0,00368
9	MLPClassifier	0,780	0,772	0,766	0,00618
10	MLPClassifier	0,773	0,771	0,770	0,00131
11	MLPClassifier	0,774	0,771	0,769	0,00226
12	MLPClassifier	0,779	0,771	0,765	0,00613
13	MLPClassifier	0,773	0,770	0,768	0,00231
14	MLPClassifier	0,774	0,770	0,768	0,00240
15	MLPClassifier	0,773	0,770	0,765	0,00390
16	MLPClassifier	0,775	0,770	0,763	0,00501
17	MLPClassifier	0,772	0,770	0,768	0,00187
18	MLPClassifier	0,771	0,769	0,768	0,00149
19	MLPClassifier	0,774	0,769	0,766	0,00325
20	MLPClassifier	0,775	0,769	0,764	0,00460
21	MLPClassifier	0,773	0,769	0,764	0,00358
22	MLPClassifier	0,778	0,769	0,763	0,00662
23	MLPClassifier	0,772	0,769	0,767	0,00232
24	MLPClassifier	0,774	0,768	0,760	0,00613
25	MLPClassifier	0,772	0,768	0,765	0,00276
26	MLPClassifier	0,773	0,768	0,760	0,00601
27	MLPClassifier	0,777	0,768	0,762	0,00613
28	MLPClassifier	0,773	0,768	0,764	0,00349
29	MLPClassifier	0,772	0,768	0,765	0,00289
30	RandomForestClassifier	0,772	0,768	0,762	0,00423
31	MLPClassifier	0,775	0,768	0,761	0,00553
32	MLPClassifier	0,771	0,767	0,763	0,00331
33	RandomForestClassifier	0,774	0,767	0,762	0,00532
34	MLPClassifier	0,769	0,767	0,762	0,00297
35	MLPClassifier	0,773	0,766	0,761	0,00520
36	RandomForestClassifier	0,773	0,766	0,762	0,00494

37	RandomForestClassifier	0,771	0,766	0,762	0,00393
38	MLPClassifier	0,777	0,765	0,755	0,00875
39	RandomForestClassifier	0,770	0,765	0,763	0,00338
40	MLPClassifier	0,767	0,765	0,763	0,00192
41	MLPClassifier	0,771	0,765	0,762	0,00419
42	RandomForestClassifier	0,768	0,765	0,762	0,00238
43	MLPClassifier	0,770	0,765	0,762	0,00334
44	RandomForestClassifier	0,768	0,765	0,762	0,00257
45	MLPClassifier	0,765	0,765	0,764	0,00040
46	MLPClassifier	0,772	0,765	0,761	0,00518
47	RandomForestClassifier	0,768	0,765	0,759	0,00398
48	RandomForestClassifier	0,770	0,765	0,759	0,00479
49	MLPClassifier	0,766	0,764	0,763	0,00129
50	RandomForestClassifier	0,770	0,764	0,761	0,00437
51	MLPClassifier	0,767	0,764	0,761	0,00226
52	RandomForestClassifier	0,770	0,764	0,759	0,00476
53	MLPClassifier	0,767	0,764	0,760	0,00293
54	RandomForestClassifier	0,764	0,764	0,762	0,00094
55	RandomForestClassifier	0,768	0,764	0,762	0,00293
56	MLPClassifier	0,766	0,764	0,758	0,00370
57	MLPClassifier	0,764	0,764	0,762	0,00101
58	RandomForestClassifier	0,765	0,764	0,762	0,00122
59	RandomForestClassifier	0,765	0,764	0,762	0,00100
60	RandomForestClassifier	0,766	0,763	0,762	0,00170
61	RandomForestClassifier	0,765	0,763	0,762	0,00155
62	RandomForestClassifier	0,769	0,763	0,759	0,00413
63	RandomForestClassifier	0,771	0,763	0,754	0,00670
64	MLPClassifier	0,766	0,763	0,761	0,00205
65	RandomForestClassifier	0,764	0,763	0,761	0,00101
66	MLPClassifier	0,767	0,762	0,759	0,00345
67	RandomForestClassifier	0,767	0,762	0,757	0,00431
68	MLPClassifier	0,765	0,762	0,760	0,00218
69	SVC	0,764	0,762	0,760	0,00132
70	RandomForestClassifier	0,769	0,762	0,755	0,00586
71	MLPClassifier	0,768	0,762	0,756	0,00480
72	MLPClassifier	0,769	0,762	0,758	0,00522
73	RandomForestClassifier	0,768	0,762	0,757	0,00450
74	RandomForestClassifier	0,762	0,762	0,762	0,00022
75	MLPClassifier	0,764	0,762	0,760	0,00148
76	RandomForestClassifier	0,769	0,762	0,757	0,00546
77	RandomForestClassifier	0,766	0,762	0,755	0,00448
78	RandomForestClassifier	0,764	0,761	0,758	0,00240
79	RandomForestClassifier	0,766	0,761	0,756	0,00402
80	MLPClassifier	0,767	0,761	0,757	0,00432
81	MLPClassifier	0,768	0,761	0,757	0,00497

82	MLPClassifier	0,765	0,761	0,758	0,00288
83	RandomForestClassifier	0,766	0,761	0,758	0,00348
84	RandomForestClassifier	0,764	0,761	0,756	0,00369
85	MLPClassifier	0,761	0,761	0,760	0,00061
86	MLPClassifier	0,767	0,761	0,754	0,00541
87	RandomForestClassifier	0,768	0,761	0,752	0,00646
88	SVC	0,762	0,760	0,760	0,00087
89	SVC	0,761	0,760	0,759	0,00089
90	SVC	0,761	0,760	0,760	0,00047
91	SVC	0,765	0,760	0,755	0,00375
92	MLPClassifier	0,765	0,760	0,755	0,00399
93	MLPClassifier	0,762	0,760	0,758	0,00192
94	RandomForestClassifier	0,767	0,760	0,753	0,00576
95	MLPClassifier	0,761	0,760	0,758	0,00130
96	MLPClassifier	0,763	0,760	0,755	0,00338
97	MLPClassifier	0,766	0,760	0,756	0,00448
98	MLPClassifier	0,768	0,759	0,753	0,00604
99	MLPClassifier	0,763	0,759	0,756	0,00265
100	MLPClassifier	0,766	0,759	0,754	0,00525
101	MLPClassifier	0,765	0,759	0,753	0,00491
102	MLPClassifier	0,761	0,759	0,755	0,00285
103	MLPClassifier	0,761	0,759	0,755	0,00244
104	MLPClassifier	0,762	0,759	0,756	0,00246
105	MLPClassifier	0,760	0,759	0,757	0,00148
106	MLPClassifier	0,760	0,758	0,757	0,00102
107	MLPClassifier	0,760	0,758	0,755	0,00267
108	MLPClassifier	0,765	0,758	0,751	0,00581
109	MLPClassifier	0,760	0,758	0,757	0,00147
110	MLPClassifier	0,760	0,758	0,755	0,00227
111	MLPClassifier	0,759	0,758	0,757	0,00088
112	AdaBoostClassifier	0,759	0,757	0,757	0,00083
113	MLPClassifier	0,763	0,757	0,754	0,00401
114	MLPClassifier	0,760	0,757	0,754	0,00246
115	MLPClassifier	0,759	0,757	0,755	0,00178
116	MLPClassifier	0,759	0,757	0,756	0,00121
117	MLPClassifier	0,759	0,757	0,754	0,00230
118	MLPClassifier	0,763	0,757	0,754	0,00413
119	MLPClassifier	0,759	0,757	0,755	0,00156
120	MLPClassifier	0,758	0,756	0,755	0,00135
121	MLPClassifier	0,761	0,756	0,754	0,00305
122	MLPClassifier	0,759	0,756	0,753	0,00217
123	MLPClassifier	0,760	0,756	0,751	0,00368
124	MLPClassifier	0,762	0,756	0,753	0,00423
125	MLPClassifier	0,759	0,756	0,752	0,00275
126	MLPClassifier	0,760	0,756	0,754	0,00316

127	MLPClassifier	0,761	0,756	0,749	0,00514
128	MLPClassifier	0,757	0,756	0,753	0,00213
129	GradientBoostingClassifier	0,757	0,756	0,754	0,00112
130	GradientBoostingClassifier	0,757	0,756	0,754	0,00112
131	MLPClassifier	0,758	0,755	0,750	0,00346
132	MLPClassifier	0,759	0,755	0,750	0,00357
133	MLPClassifier	0,756	0,755	0,754	0,00121
134	SVC	0,758	0,755	0,751	0,00285
135	MLPClassifier	0,758	0,755	0,752	0,00268
136	MLPClassifier	0,765	0,754	0,747	0,00762
137	GradientBoostingClassifier	0,757	0,754	0,752	0,00228
138	GradientBoostingClassifier	0,757	0,754	0,752	0,00228
139	GradientBoostingClassifier	0,757	0,754	0,751	0,00259
140	MLPClassifier	0,759	0,753	0,746	0,00557
141	RandomForestClassifier	0,756	0,753	0,747	0,00458
142	GradientBoostingClassifier	0,757	0,753	0,749	0,00331
143	GradientBoostingClassifier	0,757	0,753	0,749	0,00331
144	RandomForestClassifier	0,755	0,753	0,749	0,00224
145	RandomForestClassifier	0,754	0,753	0,751	0,00149
146	RandomForestClassifier	0,756	0,753	0,747	0,00383
147	GradientBoostingClassifier	0,753	0,752	0,751	0,00070
148	GradientBoostingClassifier	0,753	0,752	0,751	0,00070
149	GradientBoostingClassifier	0,753	0,752	0,751	0,00068
150	GradientBoostingClassifier	0,753	0,752	0,751	0,00068
151	LogisticRegression	0,754	0,752	0,750	0,00193
152	GradientBoostingClassifier	0,753	0,752	0,752	0,00035
153	GradientBoostingClassifier	0,753	0,752	0,752	0,00035
154	GradientBoostingClassifier	0,753	0,752	0,752	0,00042
155	GradientBoostingClassifier	0,753	0,752	0,751	0,00061
156	GradientBoostingClassifier	0,753	0,752	0,751	0,00061
157	LogisticRegression	0,754	0,752	0,749	0,00231
158	RandomForestClassifier	0,754	0,752	0,748	0,00284
159	LogisticRegression	0,754	0,751	0,749	0,00231
160	LogisticRegression	0,754	0,751	0,749	0,00231
161	LogisticRegression	0,754	0,751	0,749	0,00231
162	LogisticRegression	0,754	0,751	0,749	0,00231
163	RandomForestClassifier	0,753	0,750	0,747	0,00245
164	AdaBoostClassifier	0,754	0,749	0,746	0,00327
165	RandomForestClassifier	0,755	0,749	0,745	0,00431
166	RandomForestClassifier	0,752	0,748	0,744	0,00353
167	ExtraTreesClassifier	0,742	0,736	0,725	0,00773
168	SVC	0,738	0,734	0,732	0,00256
169	ExtraTreesClassifier	0,738	0,733	0,725	0,00562
170	ExtraTreesClassifier	0,737	0,732	0,721	0,00728
171	SVC	0,737	0,705	0,689	0,02272

172	SVC	0,737	0,579	0,500	0,11175
173	SVC	0,605	0,535	0,500	0,04973
174	SVC	0,605	0,535	0,500	0,04973
175	SVC	0,605	0,535	0,500	0,04973

APPENDIX C: RESULTS FROM REGRESSION STEP IN THE AUTO-CMUT

Table 10: Results from the best regression for both NO₂ and CO. In the original excel file the parameters with their values are also given. Parameter values are excluded from this table due to lack of space.

	Validation MSE CO	Training MSE CO	Validation MSE NO ₂	Training MSE NO ₂
0	0,228	0,187	0,550	0,540
1	0,230	0,223	0,562	0,557
2	0,230	0,190	0,587	0,579
3	0,230	0,207	0,590	0,638
4	0,231	0,204	0,591	0,597
5	0,231	0,191	0,592	0,624
6	0,232	0,226	0,598	0,634
7	0,232	0,191	0,600	0,604
8	0,233	0,192	0,609	0,598
9	0,233	0,192	0,612	0,644
10	0,233	0,193	0,614	0,672
11	0,234	0,194	0,615	0,646
12	0,234	0,194	0,616	0,618
13	0,234	0,193	0,621	0,662
14	0,234	0,231	0,636	0,702
15	0,234	0,228	0,637	0,691
16	0,237	0,220	0,638	0,655
17	0,238	0,214	0,639	0,642
18	0,238	0,213	0,639	0,713
19	0,238	0,233	0,640	0,689
20	0,238	0,231	0,642	0,684
21	0,238	0,202	0,644	0,638
22	0,238	0,207	0,644	0,687
23	0,238	0,207	0,645	0,697
24	0,238	0,209	0,647	0,706
25	0,238	0,202	0,649	0,723
26	0,238	0,215	0,650	0,716
27	0,238	0,207	0,652	0,723
28	0,238	0,206	0,655	0,678
29	0,238	0,206	0,658	0,661
30	0,238	0,208	0,661	0,663
31	0,238	0,202	0,668	0,738
32	0,238	0,206	0,670	0,672
33	0,238	0,206	0,673	0,693
34	0,238	0,207	0,674	0,697

35	0,238	0,202	0,678	0,696
36	0,239	0,215	0,679	0,708
37	0,239	0,207	0,679	0,720
38	0,239	0,208	0,682	0,712
39	0,239	0,207	0,682	0,682
40	0,239	0,204	0,687	0,733
41	0,239	0,206	0,688	0,688
42	0,239	0,202	0,689	0,739
43	0,239	0,208	0,689	0,694
44	0,239	0,214	0,691	0,689
45	0,239	0,206	0,691	0,740
46	0,239	0,213	0,691	0,690
47	0,239	0,202	0,691	0,696
48	0,239	0,206	0,691	0,694
49	0,239	0,201	0,691	0,694
50	0,239	0,206	0,691	0,706
51	0,239	0,215	0,692	0,698
52	0,239	0,214	0,692	0,699
53	0,239	0,210	0,692	0,711
54	0,239	0,212	0,692	0,692
55	0,239	0,214	0,692	0,693
56	0,239	0,240	0,693	0,707
57	0,239	0,207	0,693	0,694
58	0,239	0,214	0,693	0,695
59	0,239	0,209	0,693	0,700
60	0,239	0,261	0,694	0,710
61	0,239	0,242	0,694	0,751
62	0,239	0,214	0,694	0,697
63	0,240	0,270	0,694	0,697
64	0,240	0,226	0,695	0,693
65	0,240	0,217	0,695	0,721
66	0,240	0,279	0,695	0,716
67	0,240	0,216	0,696	0,713
68	0,240	0,240	0,696	0,713
69	0,240	0,238	0,697	0,693
70	0,240	0,244	0,697	0,703
71	0,241	0,205	0,702	0,706
72	0,241	0,202	0,702	0,718
73	0,241	0,240	0,703	0,715
74	0,241	0,275	0,705	0,715
75	0,242	0,259	0,706	0,721
76	0,242	0,232	0,706	0,763

77	0,242	0,238	0,708	0,754
78	0,242	0,258	0,708	0,731
79	0,242	0,262	0,708	0,735
80	0,242	0,257	0,712	0,758
81	0,243	0,240	0,713	0,732
82	0,243	0,260	0,715	0,775
83	0,243	0,237	0,718	0,744
84	0,244	0,221	0,719	0,741
85	0,244	0,277	0,728	0,734
86	0,244	0,300	0,732	0,743
87	0,245	0,238	0,732	0,785
88	0,245	0,267	0,733	0,755
89	0,245	0,235	0,734	0,744
90	0,245	0,245	0,735	0,745
91	0,246	0,286	0,736	0,802
92	0,246	0,210	0,737	0,797
93	0,247	0,211	0,739	0,766
94	0,247	0,263	0,739	0,751
95	0,247	0,249	0,741	0,801
96	0,247	0,208	0,746	0,760
97	0,248	0,249	0,746	0,773
98	0,248	0,210	0,747	0,784
99	0,248	0,246	0,748	0,759
100	0,248	0,279	0,749	0,808
101	0,249	0,252	0,751	0,762
102	0,249	0,255	0,751	0,762
103	0,250	0,287	0,751	0,763
104	0,250	0,249	0,752	0,764
105	0,251	0,307	0,759	0,772
106	0,251	0,215	0,759	0,799
107	0,251	0,307	0,767	0,779
108	0,252	0,262	0,769	0,828
109	0,252	0,260	0,770	0,823
110	0,253	0,274	0,771	0,818
111	0,254	0,218	0,772	0,784
112	0,254	0,284	0,772	0,829
113	0,254	0,292	0,775	0,802
114	0,255	0,302	0,778	0,828
115	0,256	0,298	0,778	0,820
116	0,256	0,273	0,788	0,837
117	0,256	0,262	0,789	0,807
118	0,256	0,299	0,790	0,830

119	0,259	0,324	0,794	0,808
120	0,259	0,330	0,796	0,841
121	0,259	0,289	0,799	0,815
122	0,260	0,303	0,801	0,822
123	0,260	0,276	0,803	0,814
124	0,261	0,309	0,803	0,817
125	0,263	0,226	0,805	0,844
126	0,263	0,311	0,806	0,821
127	0,264	0,309	0,807	0,828
128	0,264	0,321	0,809	0,869
129	0,265	0,319	0,810	0,864
130	0,265	0,318	0,812	0,826
131	0,265	0,229	0,814	0,869
132	0,266	0,229	0,815	0,834
133	0,266	0,277	0,816	0,831
134	0,270	0,306	0,817	0,854
135	0,275	0,342	0,823	0,848
136	0,275	0,344	0,826	0,841
137	0,277	0,355	0,827	0,877
138	0,277	0,268	0,833	0,880
139	0,278	0,347	0,835	0,896
140	0,278	0,331	0,836	0,859
141	0,279	0,357	0,840	0,886
142	0,279	0,339	0,840	0,881
143	0,280	0,298	0,844	0,881
144	0,280	0,362	0,845	0,888
145	0,280	0,277	0,847	0,871
146	0,281	0,244	0,848	0,869
147	0,281	0,274	0,851	0,894
148	0,282	0,246	0,854	0,898
149	0,284	0,248	0,855	0,881
150	0,287	0,378	0,855	0,871
151	0,290	0,255	0,857	0,904
152	0,291	0,378	0,858	0,876
153	0,292	0,364	0,863	0,903
154	0,293	0,395	0,867	0,893
155	0,294	0,292	0,868	0,893
156	0,295	0,324	0,872	0,914
157	0,296	0,392	0,873	0,906
158	0,299	0,411	0,875	0,898
159	0,299	0,403	0,877	0,895
160	0,302	0,330	0,879	0,921

161	0,304	0,405	0,881	0,909
162	0,306	0,410	0,881	0,900
163	0,307	0,333	0,882	0,901
164	0,310	0,374	0,883	0,903
165	0,311	0,378	0,884	0,908
166	0,321	0,289	0,886	0,920
167	0,323	0,390	0,888	0,917
168	0,325	0,292	0,889	0,908
169	0,332	0,418	0,889	0,927
170	0,333	0,424	0,890	0,918
171	0,334	0,396	0,890	0,914
172	0,334	0,459	0,891	0,921
173	0,334	0,399	0,892	0,912
174	0,335	0,423	0,892	0,929
175	0,335	0,425	0,894	0,927
176	0,337	0,329	0,894	0,916
177	0,339	0,330	0,899	0,925
178	0,341	0,336	0,900	0,921
179	0,342	0,467	0,900	0,927
180	0,345	0,432	0,900	0,924
181	0,346	0,436	0,904	0,930
182	0,350	0,363	0,905	0,932
183	0,357	0,444	0,906	0,933
184	0,358	0,438	0,908	0,936
185	0,362	0,372	0,911	0,939
186	0,373	0,350	0,914	0,940
187	0,374	0,351	0,916	0,941
188	0,399	0,377	0,916	0,943
189	0,405	0,449	0,921	0,947
190	0,406	0,416	0,922	0,937
191	0,408	0,451	0,922	0,937
192	0,412	0,421	0,928	0,945
193	0,423	0,492	0,928	0,946
194	0,424	0,476	0,928	0,946
195	0,429	0,488	0,930	0,952
196	0,434	0,413	0,931	0,949
197	0,439	0,514	0,931	0,946
198	0,444	0,477	0,931	0,956
199	0,465	0,468	0,931	0,949
200	0,465	0,485	0,931	0,958
201	0,547	0,597	0,934	0,955
202	0,594	0,643	0,937	0,951

203	0,612	0,589	0,937	0,957
204	0,625	0,612	0,938	0,965
205	0,625	0,611	0,940	0,962
206	0,637	0,624	0,941	0,963
207	0,638	0,627	0,943	0,963
208	0,657	0,658	0,953	0,969
209	0,663	0,719	0,954	0,968
210	0,674	0,673	0,956	0,972
211	0,683	0,691	0,958	0,977
212	0,698	0,708	0,960	0,976
213	0,698	0,698	0,960	0,974
214	0,711	0,709	0,961	0,977
215	0,718	0,712	0,961	0,977

APPENDIX D: CODE PREPROCESSING AND MULTIPLE CLASSIFIER FROM AUTO-CMUT

Script for performing classification in python, returns an excel file with all models tested and their scores, the score of the final model which is applied to the test. Lastly a confusion matrix with the result from the final model is computed.

Python code:

```
1. # Load General Modules
2. import pandas as pd
3. import numpy as np
4. import matplotlib.pyplot as plt
5. import seaborn as sns
6.
7. #import classifiers
8. from sklearn.ensemble import ExtraTreesClassifier
9. from sklearn.neural_network import MLPClassifier
10. from sklearn.ensemble import RandomForestClassifier
11. from sklearn.ensemble import AdaBoostClassifier
12. from sklearn.ensemble import GradientBoostingClassifier
13. from sklearn.svm import SVC
14.
15. from sklearn import linear_model, decomposition, datasets
16.
17. from sklearn.linear_model import Perceptron, LogisticRegression
18.
19. #Preprocessing
20. from sklearn.preprocessing import StandardScaler
21. from sklearn.decomposition import PCA
22. from sklearn.model_selection import train_test_split
23. import missingno as msno
24. import missingno
25.
26. # Other imports
27. from statistics import mean
28. from sklearn.preprocessing import LabelEncoder
29. from sklearn.utils import shuffle
30. from sklearn.metrics import accuracy_score as accuracy
31. from tabulate import tabulate
32.
33. from sklearn.model_selection import GridSearchCV
34.
35. # Import dataset
36. All_data = pd.read_excel('AirQuality.xlsx')
37.
38. # Change all Mox vlues to microgram/m3
39. All_data.iloc[:,9] *= 10^-3
40.
41.
42. #Divide dataset to featuresets and targetsets for both NO2 and CO
43. y_CO = pd.DataFrame(All_data.iloc[:,2])
44. X_CO = pd.DataFrame(All_data.iloc[:,[3,12,13,14]])
45. y_NO2 = pd.DataFrame(All_data.iloc[:,9])
46. X_NO2 = pd.DataFrame(All_data.iloc[:,[10,12,13,14]])
47.
48.
49. # Change column names
50. X_CO.rename(columns = {'PT08.S1(CO)': 'MOF'}, inplace = True)
51. X_NO2.rename(columns = {'PT08.S4(NO2)': 'MOF'}, inplace = True)
52. X_CO['category'] = 'CO'
53. X_NO2['category'] = 'NO2'
54.
```



```

55.
56.
57. index = pd.DataFrame(list(range(0, len(X_CO))))
58. index.rename(columns = {0:'index'}, inplace = True)
59. frames= [X_CO, X_NO2]
60.
61. #Defining missing values as nan
62. result = pd.concat(frames)
63. result = result.replace(-200, np.nan)
64.
65. #Visualize missing values
66. msno.matrix(result, figsize =(8,7))
67. msno.bar(result.sample(len(result)), figsize=(8,7))
68.
69. result = result.dropna(axis=0)
70. print( ' Number of samples after removing Nan', len(result))
71.
72.
73. #Shuffle the dataset and reset index
74. result = shuffle(result)
75. result = result.reset_index(drop=True)
76.
77.
78.
79. # Separating features and targets
80. features = pd.DataFrame(result.iloc[:, :4])
81. target = pd.DataFrame(result.iloc[:,4])
82. target_values = target
83.
84. # Give NO2 and CO a value instead of categorical variable
85. labelencoder = LabelEncoder()
86. target.iloc[:, 0] = labelencoder.fit_transform(target.iloc[:, 0])
87.
88. print('CO is given the class label :', labelencoder.transform(['CO']))
89. print('NO2 is given the class label:', labelencoder.transform(['NO2']))
90.
91.
92. #Scale all feature values
93. scaler = StandardScaler()
94. features_scaled = scaler.fit_transform(features)
95.
96.
97.
98. #Divivde datasets into trainin and testing
99. X_train , X_test, y_train, y_test = train_test_split(
100.                                     features_scaled, target, test_size=0.3
101.                                     ,
102.                                     random_state=1, stratify=target)
103.     y_train = y_train.iloc[:,0].values
104.
105.     #Checking correlaion between the target and the different features.
106.     plt.figure(figsize=(6,5))
107.     cor = result.corr()
108.     sns.heatmap(cor, annot=True, cmap=plt.cm.Red)
109.     plt.show()
110.
111.
112.
113.     # The multiple classifier
114.     class EstimatorSelectionHelper:
115.
116.         def __init__(self, models, params):
117.             if not set(models.keys()).issubset(set(params.keys())):
118.                 missing_params = list(set(models.keys()) - set(params.keys()))
119.                 raise ValueError("Some estimators are missing parameters: %s"

```

```

120.                                     % missing_params)
121.     self.models = models
122.     self.params = params
123.     self.keys = models.keys()
124.     self.grid_searches = {}
125.
126.     def fit(self, X, y, cv=3, n_jobs=3, verbose=1, scoring=None, refit=False)
127.     :
128.         for key in self.keys:
129.             print("Running GridSearchCV for %s." % key)
130.             model = self.models[key]
131.             params = self.params[key]
132.             gs = GridSearchCV(model, params, cv=cv, n_jobs=n_jobs,
133.                               verbose=verbose, scoring=scoring, refit=refit,
134.                               return_train_score=True)
135.             gs.fit(X,y)
136.             self.grid_searches[key] = gs
137.
138.     def score_summary(self, sort_by='mean_score'):
139.         def row(key, scores, params):
140.             d = {
141.                 'estimator': key,
142.                 'min_score': min(scores),
143.                 'max_score': max(scores),
144.                 'mean_score': np.mean(scores),
145.                 'std_score': np.std(scores),
146.             }
147.             return pd.Series(**params, **d)
148.
149.         rows = []
150.         for k in self.grid_searches:
151.             print(k)
152.             params = self.grid_searches[k].cv_results_['params']
153.             scores = []
154.             for i in range(self.grid_searches[k].cv):
155.                 key = "split{}_test_score".format(i)
156.                 r = self.grid_searches[k].cv_results_[key]
157.                 scores.append(r.reshape(len(params),1))
158.
159.             all_scores = np.hstack(scores)
160.             for p, s in zip(params,all_scores):
161.                 rows.append((row(k, s, p)))
162.
163.         df = pd.concat(rows, axis=1, sort=True).T.sort_values([sort_by],
164.                                                             ascending=False,)
165.
166.         columns=['estimator', 'min_score', 'mean_score', 'max_score', 'std_score
167.         ' ]
168.         columns = columns + [c for c in df.columns if c not in columns]
169.
170.         return df #[columns]
171.
172. #Defining the classifiers to be used in gridsearch
173. models = {
174.     'ExtraTreesClassifier': ExtraTreesClassifier(),
175.     'RandomForestClassifier': RandomForestClassifier(),
176.     'AdaBoostClassifier': AdaBoostClassifier(),
177.     'GradientBoostingClassifier': GradientBoostingClassifier(),
178.     'LogisticRegression': LogisticRegression(solver='liblinear'),
179.     'SVC': SVC(),
180.     'MLPClassifier': MLPClassifier()
181. }
182.
183. #Defining the parameter grid for each classifier
184. params = {

```

```

183.         'ExtraTreesClassifier': { 'n_estimators': [80, 100, 150] },
184.         'RandomForestClassifier' : { 'n_estimators': [80, 100, 150, 200],
185.                                     'max_depth': [5, 8,9, 10, 15],
186.                                     'criterion':['gini','entropy']},
187.         'AdaBoostClassifier': { 'n_estimators': [16, 32] },
188.         'GradientBoostingClassifier':{ 'n_estimators': [16, 32, 50 ,100],
189.                                         'learning_rate': [0.001, 0.005,0.005, 0.0001 ,]},

190.         'LogisticRegression' : {'C':[ 0.1, 1, 10, 50, 100, 200]},
191.         'SVC':[
192.             {'kernel': ['linear'], 'C': [0.001,0.005, 0.1, 0.5,]},
193.             {'kernel': ['rbf'], 'C': [0.00001, 0.005, 0.001, 0.1, ],
194.              'gamma': [ 0.001, 0.1, ]}],
195.         'MLPClassifier': {'hidden_layer_sizes':[6,8],'activation':['relu', 'tanh'
196. ],
197.                             'solver':['adam', 'sgd'], 'alpha':[0.001, 0.01, 0.1],
198.                             'max_iter':[1000], 'batch_size':[50,100],
199.                             'learning_rate': ['adaptive'], 'learning_rate_init': [ 0.0001, 0.01]}
200.     }
201.     # Calliga and fitting the model
202.     model = EstimatorSelectionHelper(models, params)
203.     model.fit(X_train, y_train)
204.     summary = model.score_summary()
205.     summary = pd.DataFrame(summary)
206.
207.
208.     #Create excel file with all the results
209.     print(tabulate(summary, headers='keys', tablefmt='psql'))
210.     summary.to_excel("Results_Classifier_1.xlsx")
211.     summary = summary.reset_index(drop=True)
212.
213.
214.     # Picking out the best model and run it on the test set
215.     models2 = {
216.         'ExtraTreesClassifier': ExtraTreesClassifier(),
217.         'RandomForestClassifier': RandomForestClassifier(),
218.         'AdaBoostClassifier': AdaBoostClassifier(),
219.         'GradientBoostingClassifier': GradientBoostingClassifier(),
220.         'LogisticRegression' : LogisticRegression(),
221.         'SVC': SVC(),
222.         'MLPClassifier': MLPClassifier()
223.     }
224.
225.
226.     for key in models2:
227.         if key == summary.iloc[0,1]:
228.             print('The best estimator is:', key)
229.             print('With a max_score of', summary.iloc[0,6])
230.             classifier = models2.get(key)
231.
232.             if key == 'ExtraTreesClassifier' :
233.                 n_estimators = summary.loc[ 0 , 'n_estimators' ]
234.                 clf = ExtraTreesClassifier(n_estimators = n_estimators)
235.
236.
237.             elif key == 'RandomForestClassifier':
238.                 max__depth = summary.loc[ 0 , 'max_depth' ]
239.                 n_estimators = summary.loc[ 0 , 'n_estimators' ]
240.                 criterion_ = summary.loc[ 0 , 'criterion' ]
241.                 clf = RandomForestClassifier(max_depth = max__depth,
242.                                             n_estimators = n_estimators,
243.                                             criterion=criterion_)
244.
245.             elif key == 'LogisticRegression':
246.                 c_ = summary.loc[ 0 , 'c' ]

```

```

247.         clf = LogisticRegression(C=c_)
248.
249.
250.         elif key == 'GradientBoostingClassifier':
251.             learning_rate = summary.loc[ 0 , 'learning_rate' ]
252.             n_estimators = summary.loc[ 0 , 'n_estimators' ]
253.             clf = GradientBoostingClassifier(learning_rate=learning_rate,
254.                                             n_estimators= n_estimators)
255.
256.
257.         elif key == 'AdaBoostClassifier':
258.             n_estimators = summary.loc[ 0 , 'n_estimators' ]
259.             clf= AdaBoostClassifier(n_estimators=n_estimators)
260.
261.
262.         elif key == 'SVC':
263.             c = summary.loc[ 0 , 'c' ]
264.             kernel_ = summary.loc[ 0 , 'kernel' ]
265.             gamma_ = summary.loc[ 0 , 'gamma' ]
266.             clf = SVC(C=c, gamma=gamma_,kernel=kernel_)
267.
268.         elif key == 'MLPClassifier':
269.             activation_ = summary.loc[ 0 , 'activation' ]
270.             hidden_layer_sizes_ = summary.loc[ 0 , 'hidden_layer_sizes' ]
271.             solver_ = summary.loc[ 0 , 'solver' ]
272.             alpha_ = summary.loc[ 0 , 'alpha' ]
273.             max_iter_ = summary.loc[ 0 , 'max_iter' ]
274.             batch_size_ = summary.loc[0,'batch_size']
275.             learning_rate_ = summary.loc[ 0 , 'learning_rate' ]
276.             learning_rate_init_ = summary.loc[ 0 , 'learning_rate_init' ]
277.
278.             clf = MLPClassifier(hidden_layer_sizes=hidden_layer_sizes_,
279.                                activation=activation_,solver=solver_,
280.                                alpha=alpha_, max_iter=max_iter_,
281.                                batch_size=batch_size_,
282.                                learning_rate=learning_rate_,
283.                                learning_rate_init=learning_rate_init_)
284.
285.         else:
286.             print('key not found')
287.
288.
289.     y_train = pd.DataFrame(y_train)
290.     clf.fit(X_train, y_train)
291.     y_pred = pd.DataFrame(clf.predict(X_test))
292.     print('The accuracy for the final model is: ', accuracy(y_test,y_pred))
293.
294.
295.
296.     # Plottin a confusion matrix wit the results from the final Model
297.     cm= confusion_matrix(y_test, y_pred, labels=None, sample_weight=None)
298.     TargetNames = ['CO', 'NO2']
299.
300.
301.     def plot_confusion_matrix(cm,
302.                              TargetNames,
303.                              title='Confusion matrix',
304.                              cmap=None,
305.                              normalize=False):
306.
307.         import matplotlib.pyplot as plt
308.         import numpy as np
309.         import itertools
310.
311.         accuracy = np.trace(cm) / float(np.sum(cm))
312.         misclass = 1 - accuracy

```

```

313.
314.     if cmap is None:
315.         cmap = plt.get_cmap('Blues')
316.
317.     plt.figure(figsize=(8, 6))
318.     plt.imshow(cm, interpolation='nearest', cmap=cmap)
319.     plt.title(title)
320.     plt.colorbar()
321.
322.     if TargetNames is not None:
323.         tick_marks = np.arange(len(TargetNames))
324.         plt.xticks(tick_marks, TargetNames, rotation=45)
325.         plt.yticks(tick_marks, TargetNames)
326.
327.     if normalize:
328.         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
329.
330.
331.     thresh = cm.max() / 1.5 if normalize else cm.max() / 2
332.     for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
333.         if normalize:
334.             plt.text(j, i, "{:0.4f}".format(cm[i, j]),
335.                     horizontalalignment="center",
336.                     color="white" if cm[i, j] > thresh else "black")
337.         else:
338.             plt.text(j, i, "{:,}".format(cm[i, j]),
339.                     horizontalalignment="center",
340.                     color="white" if cm[i, j] > thresh else "black")
341.
342.
343.     plt.tight_layout()
344.     plt.ylabel('True label')
345.     plt.xlabel('Predicted label\naccuracy={:0.4f}; misclass={:0.4f}'.format
346.                (accuracy, misclass
347.                ))
348.     plt.show()
349.
350.     plot_confusion_matrix(cm, TargetNames, title='Confusion matrix')

```

APPENDIX E: CODE FOR REGRESSION FROM AUTO-CMUT

Script for performing the regression part of the Auto-CMUT in python, returns an excel file with all models tested and their MSE scores. The Auto-CMUT implements the best model on the test set. On the best model the R^2 -score is calculated between the predicted test values and reference sensor

Python code:

```
1. #General inputs
2. import numpy as np
3. import pandas as pd
4.
5.
6. #Imports from sklearn
7. from sklearn.preprocessing import StandardScaler
8. from sklearn.metrics import r2_score
9. from sklearn.model_selection import train_test_split
10.
11. #imports Keras
12. from keras.losses import mean_squared_error, mean_absolute_error
13. from keras.models import Sequential
14. from keras.layers import Dropout, Dense
15. from keras.optimizers import Adam, Nadam
16. from keras.activations import relu, elu, linear, sigmoid, linear, softmax
17. from keras.layers import Dropout
18. from keras.models import Sequential
19. from keras.layers import Dense
20.
21. #Talos imports
22. import talos
23. import talos as ta
24. import wrangle as wr
25. from talos.metrics.keras_metrics import fmeasure_acc
26. from talos.metrics.keras_metrics import root_mean_squared_error as rmse
27. from talos import live
28. from talos.model.normalizers import lr_normalizer
29. from talos import Reporting
30.
31.
32. #Load Air Quality dataset
33. All_data = pd.read_excel('AirQuality.xlsx')
34. All_data.iloc[:,9] *= 10^-3
35.
36.
37.
38. #Sort CO and NO2 data in separate dataframes
39. CO = pd.DataFrame(All_data.iloc[:,[2,3,12,13,14]])
40. NO2 = pd.DataFrame(All_data.iloc[:,[9,10,12,13,14]])
41.
42. #Change names
43. CO.rename(columns = {'PT08.S1(CO)': 'MOF'}, inplace = True)
44. NO2.rename(columns = {'PT08.S4(NO2)': 'MOF'}, inplace = True)
45.
46.
47. # Remove missing values
48. CO = CO.replace(-200, np.nan)
49. NO2 = NO2.replace(1800, np.nan)
50. CO = CO.dropna(axis=0)
51. NO2 = NO2.dropna(axis=0)
52. print( ' Number of CO samples after removing Nan', len(CO))
```

```

53. print( ' Number of NO2 samples after removing Nan', len(NO2))
54.
55.
56.
57. #Reset index
58. CO = CO.reset_index(drop=True)
59. NO2 = NO2.reset_index(drop=True)
60.
61.
62. #Separate features and target for both gases
63. y_CO = pd.DataFrame(CO.iloc[:,0])
64. X_CO = pd.DataFrame(CO.iloc[:,1:])
65. X_NO2 = pd.DataFrame(NO2.iloc[:,1:])
66. y_NO2 = pd.DataFrame(NO2.iloc[:,0])
67.
68. #Scale all values
69. scaler = StandardScaler()
70. X_CO_scaled = scaler.fit_transform(X_CO)
71. X_NO2_scaled = scaler.fit_transform(X_NO2)
72. y_CO_scaled = scaler.fit_transform(y_CO)
73. y_NO2_scaled = scaler.fit_transform(y_NO2)
74.
75. #scaler2 = scaler.inverse_transform(scaler)
76.
77. #Splitt in both the CO and NO2 data into train, test and validation sets
78.
79. def split_test_valid(X, y, test_ratio):
80.     X_train, X_test, y_train, y_test =
81.         train_test_split(X, y, test_size=test_ratio, random_state=1)
82.     return X_train, X_test, y_train, y_test
83.
84.
85. test_ratio= 0.2
86.
87. X_CO_train, X_CO_test,y_CO_train, y_CO_test= split_test_valid(X_CO, y_CO,
88.                                                                 test_ratio)
89.
90. X_CO_train, X_CO_valid,y_CO_train, y_CO_valid = split_test_valid(X_CO_train,
91.                                                                 y_CO_train,
92.                                                                 test_ratio)
93.
94. X_NO2_train, X_NO2_test,y_NO2_train,y_NO2_test = split_test_valid(X_NO2_scaled,
95.                                                                 y_NO2_scaled,
96.                                                                 test_ratio)
97.
98. X_NO2_train, X_NO2_valid,y_NO2_train,y_NO2_valid =split_test_valid(X_NO2_train,
99.                                                                 y_NO2_train,
100.                                                                 test_ratio
101. )
102.
103.     #Define the Neural Network
104.     def regression_model(X_train, y_train, X_valid, y_valid, params ):
105.         model = Sequential()
106.         model.add(Dense(10, input_dim= X_train.shape[1],
107.                         activation=params['activation'],
108.                         kernel_initializer='normal'))
109.
110.         model.add(Dropout(params['dropout']))
111.
112.         #hidden_layers(model,params, 1)
113.
114.         model.add(Dense(1, activation=params['last_activation'],
115.                         kernel_initializer=params['kernel_initializer']))
116.

```

```

117.         model.compile(optimizer=params['optimizer'](lr=lr_normalizer(params['lr']
118.             ,
119.                 params['optimizer'])),
120.             loss=['mse'],
121.             metrics=['mse'])
122.         history = model.fit(X_train, y_train,
123.             validation_data=[X_valid, y_valid],
124.             batch_size=params['batch_size'],
125.             callbacks=[live()],
126.             epochs=params['epochs'],
127.             verbose=0)
128.
129.         return history, model
130.
131.
132.     #Defining the parameter grid
133.     params = {'lr': [ 0.1,0.2,0.3],
134.         'first_neuron':[2,8, 100],
135.         'hidden_layers':[1,9,100,],
136.         'batch_size': [ 50],
137.         'epochs': [40],
138.         'dropout': [0.2,0.4,0.5,0.1],
139.         'kernel_initializer': ['normal'],
140.         'optimizer': [Nadam],
141.         # 'loss':[mse],
142.         'activation':[relu, linear, sigmoid, softmax],
143.         'last_activation':['linear']
144.     }
145.
146.     # Running the Neural Network with the chosen parameter grid
147.     t = ta.Scan(x=X_CO_train,
148.         y=y_CO_train,
149.         model= regression_model,
150.         params=params,
151.
152.         grid_downsample=0.2,
153.         #reduction_method='correlation',
154.         #reduction_metric= rmse(),
155.         dataset_name='Final_Results_CO',
156.         experiment_no='1',
157.         # dataset_name = 'HPO',
158.         print_params=True
159.     )
160.     r = Reporting('CO_not_scaled')
161.     #df = pd.read_csv('Final_NO2_1.csv')
162.
163.
164.     from talos import Evaluate
165.     e = Evaluate(r)
166.
167.
168.     #Deploying model, to save results for later use
169.     from talos import Deploy
170.     Deployed_CO = Deploy(t, model_name='Final_CO', metric='val_loss', asc=True)
171.
172.     report_CO = talos.Reporting('HPO_NO2_2.csv')
173.
174.     report_CO.plot_corr('val_loss')
175.
176.     correlation_values_CO = abs(report_CO.correlate('val_loss'))
177.
178.     typed_report_data_CO= report_CO.data.convert_objects(convert_numeric=True)
179.

```



```

180.         typed_report_data_CO = typed_report_data_CO.loc[typed_report_data_CO['loss']]
181.         ]
182.
183.         # Gettin id number for best model
184.         best_model_id_CO = typed_report_data_CO.iloc[0].name
185.
186.         #Saving the best model
187.         best_model_CO = talos.utils.best_model.activate_model(t, best_model_id_CO)
188.
189.         #Use best model on the testset
190.         y_CO_predict = pd.DataFrame(best_model_CO.predict(X_CO_test))
191.
192.
193.         #Computing the R2 score for the testset
194.         R2_test_CO = r2_score(y_CO_test, y_CO_predict, sample_weight=None,
195.                               multioutput='uniform_average')
196.
197.         e = Evaluate(t)
198.         e.evaluate(X_NO2_train,y_NO2_train,model_id=1,
199.                   folds=10,asc=True,metric='val_loss')

```




Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitenskapelige universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway