UNIVERSITY OF BERGEN

Department of Information Science and Media Studies

MASTER THESIS

Braluft: Forecasting air quality using incremental models and computer vision

Author:
Rune Myklevoll

Supervisor:
Bjørnar Tessem

May 28, 2019

Abstract

Air quality in urban areas is an issue of great concern as it affects public health and local environments. By forecasting the pollutant levels public administrations may be notified of periods with potentially bad air quality and can initiate strategic policies to limit the spreading of pollutants. One of the challenges associated with forecasting air quality is the fact that meteorological conditions and anthropogenic activities change as seasons passes. This thesis targets such issues by presenting Braluft, a distributed system designed to incrementally train forecasting models over time using machine learning. The thesis makes use of the program to evaluate: (a) which variables influence the levels of two important pollutants, NO_2 and PM_{10}, at Danmarksplass, Bergen, and (b) whether the incremental approach is well suited for making air quality forecasts by continuously adjusting to new observations. The program uses weather forecasts and traffic level as input data, and the latter is assessed by applying computer vision to a web camera overlooking the area. The most promising variables for NO_2 forecasting turned out to be wind speed and traffic levels by a wide margin. PM_{10} levels are seemingly a result of more complex processes where all the observed variables have an influence. The program delivers promising results for its intended purposes, namely register trends occurring in the air quality and subsequently make air quality forecasts based on these trends. This results in good air quality forecasts for most days where the pollutant levels are low. However, bad air quality is often a result of sudden changes and can hardly be considered a trend. The program is therefore struggling to foresee such events. The concept supporting the program might prove more valuable in areas where raises in pollutant levels are less abrupt.

Acknowledgements

I would like to especially thank my supervisor Bjørnar Tessem for valuable feedback, discussions, and keeping the door open over the last year. I would to express my gratitude to the rest of study room 638 for several insightful and academic talks over the last couple of months, and for contributing to Innsjekk. Thanks to my partner and friend Tori for the support! Finally, a big thank you to my parents for providing feedback, proofreading, and printing services!

Icons in the thesis are made by Freepik and Smashicons from www.flaticon.com

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The sheer amount of data sources keeps expanding in personal and commercial contexts, and the growth seems to have no end in sight. Components with data collection capabilities are everywhere and have become a part of everyday life. The great challenge for the technology industry is how to benefit from the immense collection of data which is being recorded in our surroundings [6]. Industries are for instance investing heavily in advanced monitoring technology and data archiving, in an attempt to construct intelligent software capable of performing routine tasks. This leads to opportunities related to optimizing maintenance using predictive models but also challenges in how to process the data [7]. The new technologies introduced have made way to a new era of digitalization resulting in a reformation in many business areas changing how we work.

1.1 Smart cities

Parallel with the growing amount of data sources there has been a pursuit for smarter environments in a city context. This has led to implementations of innovations such as smart grids, smart homes, smart transportation, and smart health care [8]. However, while the field seems to continue its growth in terms of popularity there is an absence of a universally agreed definition of smart cities. One attempt to characterize these smart environments has been done by Mark Weiser by referring to them as "a physical

world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network" [9]. In a more general sense one might say that the final aim of the smart city is to make better use of public resources by reducing costs and providing better, and perhaps even new services to the citizens [10].

The IoT & big data

Two important enablers of smart technology in a city context are *The Internet of Things (IoT)* and *big data* technology [9]. Furthermore, the pursuit of smart environments is what drives the growth of available data, which again is the core of the services rendered by the IoT. IoT is also a novel paradigm, meaning it still has not established best practices and a widely accepted business model which can attract investors for further improvements [10]. IoT consists of *things* that are able to communicate with one another and its neighbours [11].

ABI Research has previously estimated that there will be 30 billion connected devices by 2020 contributing to the IoT [12]. The growth in the number of data sources is posing some challenges in terms of efficient data storing and processing. This has led to a paradigm shift from traditional computing towards more sophisticated computing, such as big data analysis techniques. [8].

Big data is considered a revolution because of the potential for knowledge extraction and decision making support based on large amounts of data, therefore making an altering to how we live, work, and think [12]. The data obtained can provide value to the city by providing new insights by uncovering hidden patterns and correlations that can reduce costs and resource consumption [8]. Data collected can, in other words, serve as a bridge between the physical and digital world by shedding new light on already existing environments [11]. A collection of techniques that can be applied in order to obtain value from big data is machine learning, which The McKinsey Global Instituate consider the main driver for the big data revolution [12].

1.2 Machine learning in society

Machine learning thrives on efficient algorithms, large datasets, and powerful computation environments making it an essential part of the big data analytics kit [13]. The field of machine learning has become indispensable in regards to extracting information out of otherwise meaningless data, such as data generated by the IoT in a smart city context [14]. The information technology surrounding us has contributed to the growth of massive amounts of data across the globe, but 80% of this data is unstructured. The idea of transforming this unstructured data into knowledge has been circulating since early artificial intelligence research in the 1980's [13].

However, mainstream machine learning venues are usually focused on novel algorithms and sandbox studies on benchmark data sets rather than publishing studies targeting real-world problems, even though the latter is influencing the broader world through implementations in the form of various applications. This bias within the field of machine learning can lead to an *algorithmic echo chamber*, increasing the gap between theoretical and applied work [15]. Furthermore, many machine learning researchers are surprised to realize that the difference in performance between various algorithms diminishes in importance outside of a sandbox context. Success in applying machine learning algorithms in real applications is rather determined by how well the domain is understood. Machine learning experts are in other words not able to solve the world's problem in isolation [15].

1.3 Air quality forecasting

The idea of contributing to the smart city ecosystem using machine learning and already existing data sources, such as sensors and imagery, form the foundation of this thesis and serves as the key motivational factor. How may one use already existing resources to create value that benefits citizens with minimum added costs? A potential area in an urban context that might benefit from new solutions is within air quality management, such as forecasting and information services.

Routine air quality forecasts are of great importance for several reasons in a society, including public health, air quality management, and science [16]. A significant association between air pollution and health issues is well asserted through many studies showing the damaging effects of components forming air pollution [17][18][19]. Main contributors to polluted air in urban areas include components such as CO, NO_2, O_3, SO_2 and particle matters of varying size [20].

Strategic moves initiated by public administrations are not uncommon when trying to reduce the concentrations of pollutants by limiting vehicular traffic. An example is number plate circulation (odd / even numbers) [20]. Using predictive models can help assist planning and enforcing such strategies by providing forecasts supporting the decision making process [20]. The forecasts should ideally be available 24-48 hours in advance in order to implement such strategies in an efficient way [16]

The effect of bad air quality is of increasing public concern, which has led to the rise of air quality standards set to protect public health [17]. The European Union has for instance established air quality standards for NO_2 and PM_{10} with concentration limits on how many times per year the mean concentration can surpass the individual thresholds [18]. Norway was in 2015 found guilty by the EFTA court of exceeding threshold values and having insufficient assessments of measures for air quality regulation. One of the cities included in the decision was Bergen which is the subject of this thesis in terms of geographical location [2].

There are naturally already services publicly available for air quality forecasting for Bergen and the surrounding areas, but the ones discovered as a part of the research for this paper were either (a) just forecasting upcoming 24-48 hours and/or (b) lacking details, such as actual pollutant levels or what the forecasts are based on. Miljøstatus.no [21] is for instance as of today in early version providing forecasts for the following day. Another service is hosted by the municipality of Bergen on their official sites [22]. Their solution is for a longer time period but at the costs of providing very little detail in the form of a manual written message. Norwegian Meteorological Institute offers air quality forecasting through their public APIs, but is currently only in beta stages and is only

hosting forecasts for the following day [23].

Changes over time

There are many interacting factors having an impact on air quality through pollution levels. Air quality in urban areas depends on local and regional emissions, as well as the geographic and meteorological characteristics of the area. The forming and dispersion of pollution should therefore be studied locally [24][25]. Furthermore, several factors may change over time. A review of articles presented in this paper shows a lot of variation in terms of how the different seasons affect air pollution and to which degree. Observations from two different stations within the city of Athens emphasize this point, where one station recorded no variations of NO_X concentration when comparing the seasons while another station noticed a significant difference when performing the same comparisons [18]. Seasons do not only affect the air quality through meteorological changes, but also manipulate anthropological activities leading to changes that might have an impact on the air quality. The amount of cruise ships arriving in Bergen during the summer months is expected to grow in the future resulting in an increased contribution to NO_2. Similar effects are seen for PM_{10} during the winter due to domestic heating. [2].

Other factors may change over time as well, including changes in vehicle type distribution and other transportational factors. Electric cars have seen an increased popularity which may have an observable effect on the air quality. This growth is however sensitive to changes in benefits for choosing such vehicles, including exemptions on toll fees [2]. The amount of traffic using motorised vehicles is also expected to increase by 1.9% yearly in Bergen [2]. Several improvements in transportational infrastructure are also in motion, including light rail tracks to new locations and new opportunities for walking and cycling [2]. An addition to this meteorological conditions tend to change every year, and this may have a significant effect on the air pollutant levels and spatial location of the pollutants within the city. Annual averages of NO_2 and PM_{10} may vary between 3-5 $\mu g/m^3$ from one year to another [2].

1.4 Research questions

This thesis is based on an attempt to build a solution that forecasts air quality for several days in advance without making a compromise in regards to details and at the same time contribute to the overall knowledge base of air quality. The proposed solution addresses the issue of changing factors over time by being built on a foundation of online machine learning models. This leads to the following research questions:

1. Which variables are ideal for air quality forecasting when considering traffic levels and meteorological variables?

2. How well are online machine learning models performing when trying to forecast air quality?

The designed solution is a program that does not try to directly answer the research questions, but rather provides relevant data from external sources and data created by the program itself. This data are then the subject of further analysis that targets answering the research questions. The first research question is answered by looking at the relationship between independent variables, such as wind speed or traffic level, and the target variables (NO_2 and PM_{10}). Visualization of the data is the main contributor in this answering process along with correlation coefficients for a numerical measurement.

The second research question is answered by comparing the observed air quality values with the forecasted ones using common machine learning metrics for regression problems. The idea of treating it as a regression problem is to look at predictive capabilities of the machine learning models as a numerical value to see how low error rate it is possible to achieve using the selected methods. The comparison is also visualized in order to see how well the models fit to the observed data.

Chapter 2

Design science

Information systems are developed to improve the efficiency within an environment or an organization. Such creations are often of a complex nature and can be studied at several levels, such as knowledge about the development of applications, as well as information technology at a managerial level [26]. One might therefore argue that two different, but complementary, research paradigms are needed to grasp the complexity of information technology: behavioural science and design science [26].

Behavioural science has roots in natural science research methods and is about explaining how and why things are the way they are. The end goal of the research paradigm is *truth* [26] [27]. Behavioral science is revolving around developing and justifying *theories*, where progress is achieved when the theories provide more accurate explanations of phenomenons then past ones, and success can be measured by the theories predictive ability of future observations [27] [26]. In an information technology context this can result in theories related to a system's usage, usefulness, and impact within an organization [26].

Design science, on the other hand, has roots in the engineering field and is a problem-solving paradigm that seeks to build innovative artifacts by applying knowledge of tasks and situations to the building process [27]. Design can in other word be regarded as both a *process* and an *artifact* where the goal is *utility*. In the end knowledge and understanding of the problem domain are achieved through the development and usage

of the designed artifact [26].

Novelty While design science is based on creating artifacts, it should not be mixed with system development as a routine design. The latter is about applying existing knowledge to solve organizational problems using best practices. Design science is contrarily addressing unsolved problems in an innovate way, or solved problems in a more efficient way. Furthermore, design science research has a clearly identifiable contribution to a knowledge base [26].

It is difficult to build something really *new*, as most work is based on previously existing ideas or products. Innovation might, however, take several different shapes as seen in figure 2.1, such as improvement by implementing new solutions to existing problems, exaptation by extending known solutions to new problems, and invention with new solutions to new problems [1].

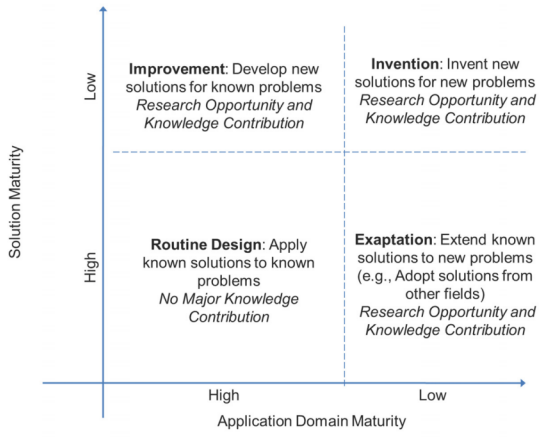


Figure 2.1: Contribution matrix [1]

Process

Similar to the development and justification of theories in behavioural science the design sciences process is mainly built on two stages, *building* and *evaluating* [27]. This loop between building and evaluating is usually performed several times before the final artifact is complete [26].

Building

Building artifacts as a part of design science research is a pursuit of an artifact with a specific purpose, proving in the process that such an artifact *can* be developed [27]. The end products of design science are generally described as either

Constructs

Constructs assist the composition of vocabularies, enabled knowledge sharing within a domain. Such conceptualizations include for instance entities, attributes, and consensuses [27].

Models

Models are built upon a set of constructs and their relationships in a formal manner, resulting in representations of the real world such as an Entity-Relationship Model (ER-model) [27][26].

Methods

Methods are a way to perform goal-directed activities [27]. They are in other words providing guidance on how to solve problems using, for instance, mathematical algorithms, textual descriptions of approaches, or a combination [26]

Instantiations

The instantiations are a realization of an artifact in its environment, capable of solving a specific task by operationalizing constructs, models, and methods [27]. Furthermore, the implementations prove the feasibility or effectiveness of the models and methods that are included in the artifact's implementation [27].

Evaluating

The evaluation phase is concerned with assessing the utility provided by an artifact in order to solve a given problem [26]. The evaluation results in more information and a better understanding of the problem space, highlighting the improvement potentials in terms of both the building processes and the artifact [26].

Performance is a relative term connected to the intended use since artifacts can potentially solve several different problems [27]. The evaluation metric is therefore dependent on the particular artifact's intended environment defining what it is trying to accomplish [27]. Such metrics might be based on functionality, completeness, reliability, usability, or how good the artifact is fitted to the organization [26]. The overall progress is achieved when old technology is surpassed by more efficient innovations [27].

Knowledge base

There are two types of scientific research in the information technology practice, descriptive and prescriptive. While the behavioural science field is generally based on descriptive knowledge, design science is corresponding to prescriptive research activities [27].

Incomplete understanding of the environment where the problem is originated can result in poorly designed artifacts or unforeseeable side-effects. The creation of artifacts is thus dependent on what is called *kernel theory* [27]. The kernel theory refers to any descriptive knowledge used to inform the artifact building process about the problem or its environment. This knowledge may have different forms, such as observations of a phenomenon, principles, and natural laws [1].

From the prescriptive knowledge base the researcher can in a design science study investigate similar known artifacts that have been used to solve a similar problem. This may assist the process of setting a knowledge baseline by indicating the level of novelty in the new artifact and by providing knowledge [1].

Knowledge from behavioural science and design science are accordingly both important as they provide the raw materials to a design science research project. This through foundations from historical research on either information systems or referenced disciplines, and methodologies providing guidelines used to justify theories and evaluate artifacts [26].

Chapter 3

The artifact

3.1 Braluft

This thesis proposes and demonstrates the use of an artifact made for air quality forecasting, and monitoring of how different parameters affect the air quality of urban areas. The proposed artifact goes by the alias *Braluft*, which is Norwegian and translates as *good air*. The name is inspired by the overall objective of this thesis, improving the city of Bergen's air quality by offering knowledge about the problem space and utility in the form of the artifact itself.

Braluft explores how a combination of traffic data and meteorological variables relate to the air quality on Danmarkplass in the city of Bergen. Additionally, it is attempting to forecast the air quality one week ahead using machine learning models. The constructed artifact is in other words mainly dealing with three data themes: Weather data, traffic data, and air quality data. Furthermore, all three data themes exist within the program as both observations and forecasts.

The artifact is a complete software stack running on a group of virtual machines in the cloud in a microservice-like architecture, including a single-page application (SPA) available at braluft.no providing observational data, forecasts, and statistical insights related to the performance of the machine learning models.

Figure 3.1: Air quality forecasts as a sum of traffic and weather data

3.2 Program design

Braluft is designed to take an incremental approach to solve the issue of changing factors over time. The general idea behind this is that the artifact has no data or knowledge about air quality upon initialization on day 0. When the first day has passed the observed data for that day are sent to train the machine learning models in the artifact and air quality forecasts are being made for the following seven days. This procedure is repeated every night resulting in a growing set of underlying data and potentially smarter models.

Online learning The incremental approach of the program is made possible to realize by using a concept called *online learning* in the machine learning field. Two different approaches to training a model using machine learning are batch learning and online learning. Batch learning is training models based on complete data sets and is the most common among those two [12][28]. However, in many applications, time is of the essence and a performed task is only valuable within a certain period, such as predicting stock prices and earthquakes. Online learning is a paradigm within machine learning that is based on learning one instance at the time and is therefore capable of making changes over time. This strategy is also a way to handle big data volumes as machines do not need to store large data sets in memory [29]. Online learning is, in other words, useful when it is problematic to fit entire datasets in memory or when the learning systems need to adapt to new patterns [28] .

In online learning training samples are being observed in a sequence. For every training sample a prediction is initially made. The correct, observed value is then presented to the algorithm. The algorithm may finally decide whether to change the parameters of the model or not, in an attempt to better fit to subsequent samples during training or prediction [30].

3.3 Daily routine

The underlying processes and architecture of the program that enables the incremental approach are in this paper presented twice. The intention of the following presentation is to provide a conceptual understanding of what the program is trying to achieve without discussing implementational details, but rather give an overview of the main steps included in the process. A more thorough explanation of the implementation of the program is provided in the Architecture chapter, and the Modelling chapter discusses how the machine learning models are constructed.

As mentioned above, the program is designed to perform a set of operations every night where data is gathered, models are updated, and forecasts are made. This process can be divided into roughly two steps, training, and forecasting. The first step in this process concerned with training starts with the gathering of observational data for the day that went by. Data included in this operation are observational weather data, observational air quality data, and observational traffic data. All the gathered data are then stored in a relational database before it is used to update the machine learning models in the program. Several machine learning models exist for both air quality and traffic forecasting, and all the models in each category are updated at the same time in a sequence. This step is illustrated in Figure 3.2. At the end of this step the overall data set of observed data has increased, and the underlying machine learning models should ideally perform better than before.

The next (and final) step of the nightly procedure is concerned with making air quality forecasts for the next seven days. The initial part of this step is gathering weather forecasts and creating traffic forecasts for the upcoming week which serves

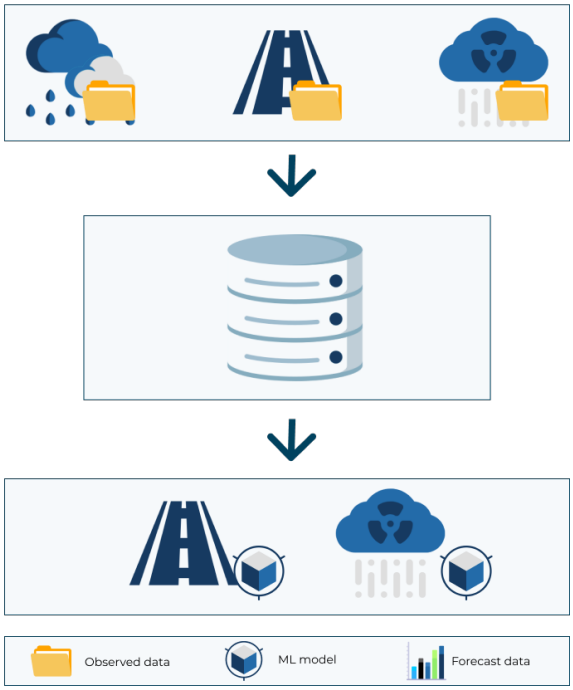


Figure 3.2: Training step

Every night observational data are collected for the previous day and stored in a relational database (weather, traffic, and pollutant levels). This data are finally used to train internal machine learning models in the program responsible of making traffic forecasts and air quality forecasts.

as input data for the air quality forecasting. The weather forecasts are originating from external sources and the traffic forecasts are made by the program itself using separate machine learning models. The weather and traffic forecasts are then sent to the machine learning models for air quality forecasting for the upcoming week before all the forecasting data are stored in the relation database alongside the observational data. Figure 3.3 illustrates this step at a conceptual level.

These two steps sum up the operations performed actively by the program in a simplified manner. A more detailed description of the system is as mentioned before the subject of later chapters.

3.4 Spatial location

There are a total of four stations measuring air quality in Bergen and each station represents a unique type of area based on the centrality of the area and whether it is close to heavy traffic or not, as seen in Table 3.1. Danmarksplass was evaluated to be the most promising location for this research work, due to its central position just south-east of downtown Bergen, high traffic, and general availability of relevant data sources. Weather observations are registered close to Danmarkplass at a weather station at Florida, weather forecasts can be accessed using latitude and longitude, and the location offers opportunities in regard to traffic assessment.

	Central	**Suburb**
Heavy traffic	Danmarkplass	Loddefjord
Little traffic	Rådhuset	Åsane

Table 3.1: Measurement stations in Bergen [2]

3.5 Intervals

The main building block of the braluft ecosystem is the interval. It can be considered both (a) an *interval* as a 6-hour long time period used at a conceptual level to divide days

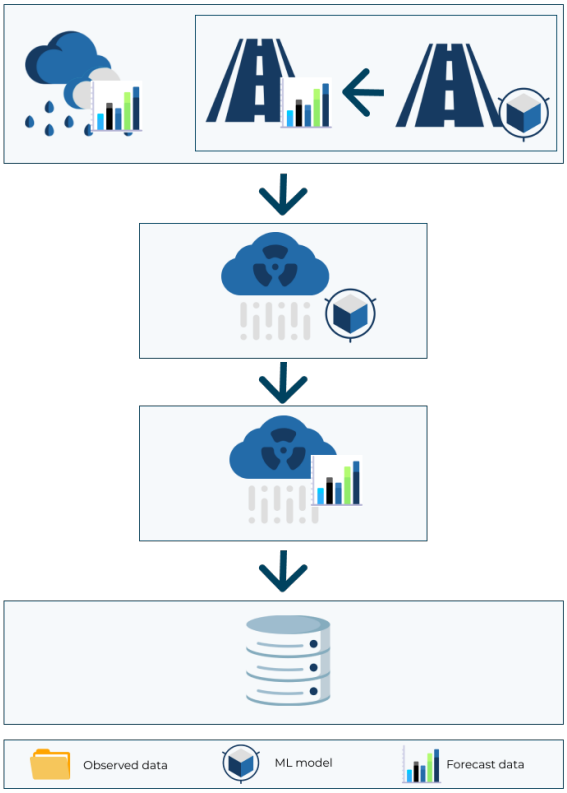


Figure 3.3: Forecasting step

The initial step of making air quality forecasts is to gather weather forecasts from external sources and a traffic forecast from the internal traffic forecasting models in the program. These data are sent to the internal air quality forecasting models in the program which returns forecasts that are stored in the relational database.

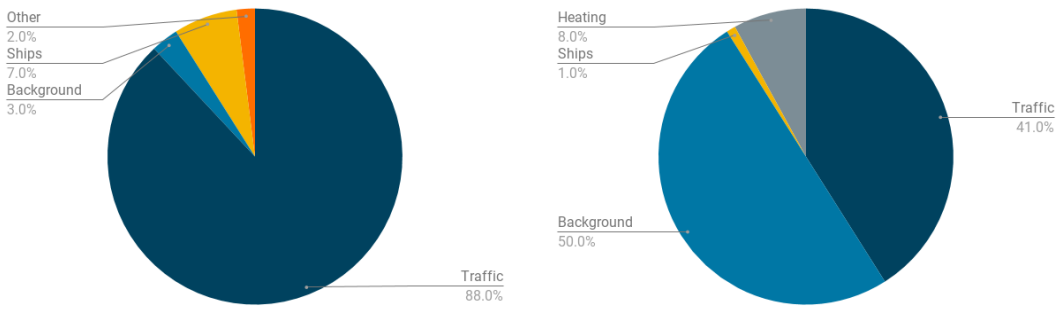
into shorter time ranges and (b) **interval** as data structure in the program responsible of storing many different data types which are related to the time period.

An *interval* is a time period within a day and the final objective of the program is to forecast air quality for each individual *interval*. Each day is divided into four *intervals*, 00-06, 06-12, 12-18, and 18-24. The 6-hour length was decided based on mainly two factors: The temporal dimensions of the data from external sources and selecting the approach that provided best generalization capabilities but still has enough details to provide usability. The intention of dividing days into *intervals* it is to get an understanding of how the air quality evolves during a day and how it relates to human factors, such as rush hour.

The **interval** as a data structure in the program may contain the following data:

- Weather observations
- Traffic observations
- Air quality observations
- Weather forecasts
- Traffic forecasts
- Air quality forecasts

Which data each **interval** actually possesses and the quantity of each data type depends on how the **interval** is related to the current date. Only **intervals** that belong to the past have observations, and future **intervals** generally obtain forecasts each day one week in advance, with the exception of the traffic forecasts that are only made once per **interval**. Both traffic and air quality forecasts are connected to potentially several different models. The ideal passed **interval** looks in other words like this, where n refers to the number of traffic models and m is the number of air quality models:



(a) NO_2 (b) PM_{10}

Figure 3.4: Pollutant sources at Danmarkplass [2]

Type	**Quantity**
Weather observation	1
Traffic observation	1
Air quality observation	1
Weather forecasts	7
Traffic forecasts	1 day $\times n$
Air quality forecasts	7 days $\times m$

3.6 Air quality

This thesis and the design of Braluft are targeting the air pollutants NO_2 and particle matters smaller than 10 µm (PM_{10}) for air quality assessment. The air quality data is delivered by NILU - Norwegian Institute for Air Research which is an independent, nonprofit institution concerned with raising awareness and increasing knowledge of climate change and environmental pollution through their research and services [31]. Among these services is an open API (api.nilu.no) serving historical observations for several pollutants across various stations in Norway. One of these stations is located at Danmarkplass, which serve as a data source for NO_2 and PM_{10} observations in the Braluft-program.

The role of the air quality observational data is twofold: (a) It is being used for training the underlying machine learning models in the program in order to make better predictions, and (b) to validate the predictions made by the artifact. In terms of representing the air quality numerical values are used for the NO_2 and PM_{10} levels. More specifically is the micrograms per cubic meter ($\mu g/m^3$) unit being used in the artifact for both the pollutants. No conversions are being made to the data collected from the API, but mean values are calculated for each **interval**.

3.7 Weather data

Unlike air quality and traffic data the artifact does not observe or try to predict any of the meteorological variables. Both forecasts and observations of the weather data are gathered from Meteorologisk Institutt, a public administrative body in Norway providing meteorological services for civilian and military usage [32]. Many of the services they provide are available through their public APIs, including those being used by Braluft.

The selected meteorological variables being used are essentially the intersection of the parameters provided by the APIs hosting the weather forecasts and observational data. These parameters are:

- Wind speed
- Wind direction
- Humidity
- Temperature
- Pressure
- Precipitation

3.8 Traffic

Traffic data plays a vital part in the program along with weather and air quality data since sufficient knowledge about traffic is a prerequisite for modeling air quality [2]. While some data and reports exist related to local traffic there are currently limited options in terms of open data solutions. The available sources found were deemed unfitted for the Braluft project because of the lack of details in either the spatial or temporal dimensions. Traffic data aggregation is therefore one of the processes included in Braluft. Traditionally the state of traffic congestion is conducted by using various types of sensors, such as piezoelectric sensors responding to pressure on the road [33] or inductive loops [34]. The increased usage of GPS-devices such as smartphones has also led to the emergence of network-wide traffic data capable of solving this problem [33]. Such data is unfortunately not available in the context of Braluft. Luckily, there is an available web camera overlooking the intersection on Danmarksplass and parts of Fjøsangerveien, mainly in the direction of the central parts of Bergen. Images from this camera are used as input to the application for assessing the traffic congestion in the intersection.

Traffic assessment through video camera footage can be grouped into three categories [33]:

- Detection-based methods identifying and counting vehicles
- Motion based methods tracking vehicle movement
- Holistic methods analyzing images a whole

The Braluft application takes a detection based approach, by downloading snapshots of the web camera images and counting vehicles observed using a pre-trained machine learning model. The **intervals** in the program are making use of this data by aggregating the vehicle counts in the same time span as the **interval**, resulting in one final numerical value representing the vehicle count for the six hours.

.

Chapter 4

Air quality

4.1 Pollutants

As mentioned before, air quality is in this thesis assessed by investigating the presence of NO_2 and PM_{10}, two of the most important pollutants concerning Norwegian cities along with $PM_{2.5}$ [2]. High concentrations of the components forming air pollution can arise based on several different sources and conditions [18][20], such as:

- Local sources (traffic, construction, industry, heating, etc)
- Natural particle sources (dust)
- Inefficient atmospheric dispersion conditions
- Weather conditions enabling long-range transport of pollution components

4.1.1 Nitrogen dioxide

Nitrogen dioxide (NO_2) is a toxic gas with great irritating power and is considered one of the main pollutants of concern in the matter of air quality [20][16]. The pollutant is part of a larger group of gases and components called nitrogen oxides (NO_X) [2] and is responsible for the yellowish color that can cover highly polluted cities [20]. It is considered a secondary pollutant, as it is derived from nitrogen monoxide oxidation

that occurs in the atmosphere, it is not produced by for instance a vehicle directly [20]. NO_2 is capable of having both short and long term health effects, particularly when exposed to sensitive people [20]. The gas may contribute to reduced lung function and worsening of respiratory diseases [2]. It is also contributing to the formation of acid rain leading to possible alterations in the ecosystem [20].

NO_2 is the most challenging pollutant in Bergen with regard to regulatory requirements, where the yearly levels exceeded threshold values in 2010, 2012, 2013, 2014, and 2016 [2]. However, levels are expected to drop significantly by 2021, mainly due to new vehicular technology including zero-emissions vehicles, but exceedings of hourly threshold values might still occur, especially during inversion of temperatures. This is a natural phenomenon where the temperature layers are reversed, meaning hot air is trapping cold air at ground level and not allowing pollutants to diminish [2].

4.1.2 Particulate matters

Particulate matters exist in a wide variety in terms of size, ranging from a few nanometers to about 100 micrometers. PM_{10} is particles smaller than 10µm and can be referred to as the inhalable fraction where exposure can lead to development and worsening of lung and cardiovascular diseases [20][2].

Particles are formed by a complex mixture of many different solid and liquid substances of various nature, including metals, carbon, nitrates, and sulfate [20]. These particles can be of a primary or secondary nature. Particles originating from primary sources are usually a result of anthropogenic activities including combustion of fossil fuels in vehicles, but also natural phenomenons such as wildfires. Secondary sources, on the other hand, are for instance chemical reactions, condensation, and coagulation in the atmosphere [19]. PM_{10} is at Danmarkplass mainly attributed to domestic heating and traffic through resuspension of dust from roads and tire wear [2].

There has not been registered any exceedings of threshold set by the official regulations for yearly PM_{10} values in Bergen since the measurement program was started in 2003 [2]. This trend seems to continue, as the risk of exceeding the threshold remains

low. However, there are several days per year with PM_{10} levels above what is recommended by The Norwegian Directorate of Health [2]. It is expected more emissions from road dust towards 2021 in Bergen, but direct emissions from vehicles should at the same time decrease due to new vehicular technology, meaning the total amount of PM_{10} should remain about the same with the exception of tunnel openings in the central part of the city [2].

4.2 Weather

There is a lot of uncertainty regarding the connection between meteorological variables and the impact they have on air quality [35]. The following section is therefore dedicated to reviewing the effect of meteorological variables in similar studies to extract which variables are worth exploring when designing air quality models for Braluft. *How* they are used should however be decided by the models during training.

The knowledge about each variable gained from the reviews and during data exploration can also facilitate the development of new similar tools, and planning of future activities in the area to improve on the air quality [18].

Wind speed and direction

Local breeze has been observed as a main influence on the air quality when considering meteorological variables in Barcelona. More specifically, the traffic/wind speed ratio. In other words, a positive change in air quality was observed when the wind speed increased and/or the amount of traffic decreased [24].

Similar observations are also seen in Cairo, where the prevailing presence of wind results in a significant negative correlation with particles in the air [25]. Furthermore, the same study indicated that the wind direction was able to affect the presence of NO_2. PM_{10} did not seem to be influenced by wind direction, suggesting that particles are of urban origin [25].

A combination of wind speed and direction were deemed the most important me-

teorological variables overall in Athens and Helsinki, but with some local variations. [18].

The same tendencies were shown in three different cities in China, where NO_2, $PM_{2.5}$, and PM_{10} diminished as the wind speed increased, except during the summer. Wind direction also had an impact in these cities, where the highest concentrations of polluted air were associated with certain wind directions, changing from city to city [35].

In regard to forecasting in Bergen, there is a reason for optimism when considering wind speed. Some concern should be raised towards wind direction as there seems to be a somewhat low variance of wind direction in Bergen, especially during the winter where wind mainly flows in from south/south-east. More variance is seen during the summer [2].

Humidity

Studies in Cairo show that high concentrations of NO_2 occurred when the humidity was less than or equal to 40%. Every other observed component were, on the other hand, peeking with humidity over 80%. Furthermore, the correlation was found to be stronger between NO_2 and the humidity than the other components [25]. A similar study surveyed three Chinese cities and concluded with an all over positive correlation between humidity and the components chosen in the Braluft application, NO_2 and PM_{10}. This was especially clear during the winter season [35]. Humidity was found to have little to no effect on the NO_2 levels at three different sites in Ireland [16].

Temperature

The same study from Cairo concluded that there was no significant association between temperature and the primary local pollutants. An increase was seen in NO_2 during warmer periods but was mainly attributed to other seasonal factors [25]. The correlation between temperature and air pollution in the previously mentioned Chinese cities shows varying results, both in terms of whether it had a positive or negative effect, and to

which degree. These variations seem to vary a lot from season to season with different results in each of the cities [35]. Just a little correlation was found between temperature and NO_2 levels in different sites in Ireland [16]

Based on these studies, it is difficult to conclude how much impact temperature has on air pollution in general, since seasons and local climate have to be considered. It does, however, justify further exploration in terms of what the effect it has in when developing predictive models based on Danmarkplass, a location with a different climate than previously mentioned studies.

Air pressure

The effect on air quality in regards to air pressure seems to be limited in similar historical research. However, the effect was found to vary when tested in four different sites in Ireland, from insignificant to significant [16].

Precipitation

Precipitation is rarely mentioned in the reviewed research on air quality forecasting so it is hard to anticipate how or how much the parameter will affect the forecasting capabilities. However, it may have a positive effect by washing pollutants of the roads so that resuspension is avoided [2].

4.3 Impact from traffic

Air quality in urban areas is strongly influenced by the level of road traffic emissions [24][36][20] and motor vehicles emit about 500 different compounds [25]. Local emissions from traffic have been shown to be the main source of NO_X and PM_{10} in the urban areas of both Helsinki and Athens. This despite their differences in terms of climate and human factors, such as population and cultural differences that might have an impact (for instance attitude towards public transportation) [18].

Rush hours Studies performed in Barcelona show an increase in all the observed air pollution components during the rush hours in the morning, including NO_2 and PM_{10} [24]. Similar observations are also seen in Helsinki and Athens, where NO_X and PM_{10} peaked during the rush hours [18].

There are two types of emission produced by vehicles, namely emissions produced by exhaust and non-exhaust. The former refers to pollutants directly emitted into the air from the vehicle formed during fuel combustion in the engine or formed during the emission itself when exhaust gases are mixed with the ambient air [24]. Non-exhaust emissions, on the other hand, are a result of resuspension of road dust from the degradation of tires, brakes, and pavement abrasion [24]. Studded tires are providing an extra contribution to the resuspension of such particles, which is why they often are regulated by fees. This has proven to be effective and provides an extra income to support policies for emission control [2]. Data gathered from several European cities suggest that emissions from the exhaust and non-exhaust sources contribute about the same amount of particulate matter. The percentage of emissions from non-exhaust sources can rise to up to 90% in northern European countries during the winter, with studded tires and measures for de-icing the roads [36].

Applied policies for reduced emissions are usually targeting exhaust as a source of the pollution through means such as extra toll during rush hours, park-ride-systems (parking areas in the outskirts of urban areas connected to public transport), and incentives for car sharing. While such measures can lead to a significant reduction of emissions caused by the exhaust, it seems to have very little effect on non-exhaust emissions [36].

Chapter 5

Machine learning

An essential part of the Braluft ecosystem is the air quality and traffic forecasting models. The chosen approach for creating these models is using machine learning. Machine learning is an interdisciplinary field that includes elements from a variety of sources, for instance artificial intelligence, cognitive science, statistics, and several others [29]. Machine learning is capable of handling tasks too complex for fixed programs written by humans because of its potential to create generalizations automatically from examples, and more complex tasks can be solved as the set of examples grow [3].

Machine learning algorithms can be categorized broadly into three categories, *supervised learning, unsupervised learning, and reinforcement learning* [28].

Supervised learning Supervised learning is concerned with mapping inputs to outputs in the form of labels [12]. The data sets used for training the models using supervised learning contain samples of input-output pairs [28]. Classification is usually a supervised task where the outputs are in the form of a discrete value. The learning algorithm is another words asked to produce a function $f : \mathbb{R}^n \rightarrow \{1, ..., k\}$ where $\{1, ..., k\}$ is the set of different possible categories. Another variant is producing a function that computes the probability distribution of the different categories [4]. Regression problems are another kind of supervised task that can be solved using machine learning, but the outputs are continuous unlike classification problems [12]. The algorithms are in

other words producing the function $f : \mathbb{R}^n \to \mathbb{R}$ [4].

Unsupervised learning Unsupervised learning is applied when the desired output is not known but we want to know the structure of the data [12]. Labels of the data are in other words not included in unsupervised learning algorithms [29]. Clustering, for instance, is the problem of finding partitions inside observed data, which can be used for creating rules for predicting the outcome of future data [37].

Semi-supervised is an option when a small amount of the samples in the data set contains the desired label, but missing in the majority. The models are then based upon both labeled and unlabeled data [28].

Reinforcement learning Reinforcement learning is based upon learning through feedback in the form of reward or punishment from an external environment [28]. While supervised and unsupervised learning focus mainly on data analysis, reinforcement learning is preferably used for decision-making problems [29].

5.1 Generalization

A key challenge in machine learning is making sure the models are capable of making good predictions to *new* data, not the data which the models are based upon. This ability is often called *generalization*. A strategy for measuring how well the model generalizes is to measure how well the model is making predictions on a separate test set containing samples that are not used to train the model. One might, in other words, distinguish between *training error* and *test error*, where the former is used to direct the training, and the latter is used to evaluate the model. In order to achieve generalization capabilities in a model, the training error must be small, and the gap between training error and test error should be narrow. Two central challenges in machine learning in this context are the issues of *over- and underfitting the model* [4].

The issue of overfitting An indication that overfitting is occurring is when a model predicts well on training data, but bad on a separate test set. The distance between training error and test error is in other words too wide and the models fail to generalize to unseen data [4]. Generalization errors can be divided into *variance* and *bias*. Variance refers to a model's ability to make consistent predictions and bias is the ability to learn the wrong thing. Both should be minimized for the most accurate predictions [12]. With this in mind, a powerful learner is not necessarily better than a less powerful one [3].

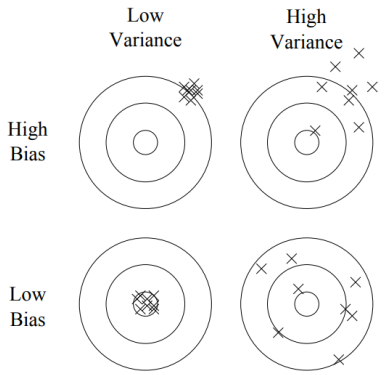


Figure 5.1: Bias and variance illustrated [3]

Regularization is a group of techniques attempting to reduce overfitting and improve generalization. The general idea is to apply a regularization term to an evaluation function [3]. Early stopping, Lasso, and Ridge are a few examples. However, the techniques introduce new parameters that need to be tuned in order to achieve a good fit to unseen data, resulting in additional processing time, for instance using cross-validation and grid search [7]. Regularization should, however, be used with caution, as *underfitting* may occur instead [3].

Underfitting Underfitting, on the other hand, is when the training error is higher than the accepted level, which can occur for instance if the training data is too complex for the chosen algorithm [4].

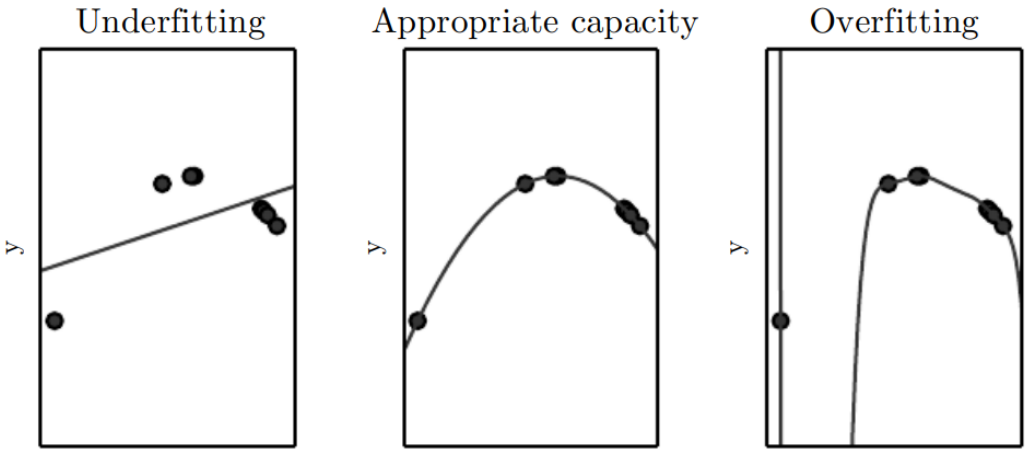


Figure 5.2: Overfitting and underfitting visualized [4]

A strategy for improving the model when it is under- or overfitting is to alter its capacity, which refers to its ability to fit a wide variety of functions. A low capacity can lead to underfitting because of its lacking capabilities of capturing complex structures in the data and high capacity can result in overfitting where the model is basically just memorizing the structure of the training data instead of trying to generalize. It is therefore important to consider this trade-off when building the model in order to obtain the best generalization capabilities [4].

5.2 Machine learning process

Machine learning algorithms are in general descriptions of how training examples should be processed [4]. A training example is a collection of features with quantitative data collected from the object or event we want the algorithm to process. The examples are usually represented as vector $[x_1, \ldots, x_n]$ where x_i is a feature, a piece of information included in the representation of the example. A set of examples with several features results in a matrix which is a common way to describe a data set in a machine learning context [14].

Many machine learning problems can be solved by designing the right set of features and process them using a simple machine learning algorithm. The ambitions behind

this designing process are to separate the factors of variation in the data set [4]. The performance of machine learning models are in other words heavily dependent on how the data is represented [28].

A typical machine learning process usually goes through preprocessing, learning, and evaluation phases [28].

5.2.1 Preprocessing

Preprocessing is the act of shaping raw data into a more suited form by removing unwanted noise and transforming the data into input suited for learning. [28].

Data cleaning

Missing data values are not uncommon due to problems such as malfunctioning data sources. This leads to implications with applying machine learning algorithms that are not capable of taking missing data into account resulting in less accurate models [29]. Data noise and outliers are therefore typically removed from the data set before machine learning algorithms start the training process [12]. However, noisy data can contain interesting patterns in itself, so deletion is not always the wisest choice. Missing or corrupt data can, for instance, be replaced using accurate predictive methods [28].

Features

A feature in a training example is "an individual measurable property of the process being observed" [38]. For instance, in a classification context is the objective of the feature to provide useful information about the classes in the data, or more specifically, we want the features to help us distinguish the classes. This means that a feature is irrelevant if it is conditionally independent of the class labels [38].

Feature engineering The process of defining new features is often referred to as feature engineering and may for instance be performed using domain knowledge [12]. The approach of using domain knowledge is, however, a costly one because of the

dependency on human labor [28]. Selecting the ideal features is one of the most time-consuming processes in regards to machine learning, and the tasks grow further in complexity with an increased vertical and horizontal size in the datasets [12]. A possible reason behind this is the fact that creating features often is a domain-specific task, while learning algorithms are often way more general-purpose [3].

Dimensionality reduction Data sets with very high dimensionality require a massive amount of capacity in terms of memory and a high computational cost for training, while simultaneously risking reduced generalization capabilities because of what is referred to as *The curse of dimensionality*. The term is meant to describe the phenomenon of algorithms performing well in low dimensions, but become hard to deal with given a higher dimension [3]. Furthermore, the Hughes effect states that the effectiveness and predictive ability of algorithms decrease after a certain point when the datasets grow in dimensionality. In other words, machine learning algorithms might lose accuracy as a result of too many features in datasets of static size. Even though it might seem obvious, it is worth mentioning that there is no universally ideal subset of features, meaning the feature selection process is individual for all task [14]. Dimensionality reduction is concerned with trying to decrease the number of features in the data without losing a significant amount of information [12]. Another way of looking at it is to divide the problem of concept learning into two subtasks: deciding which features to use and how to best combine them [39].

Feature selection While feature engineering is related to creating features, feature selection is the process of selecting the best features [12]. The goal is in other words to select a subset of variables capable of efficiently describing the original input data, while simultaneously reducing noise and removing irrelevant variables. The desired effect of this action is increased knowledge about the data, reduced computational complexity, and overall predictive performance on the dataset. [38].

Many datasets consist of highly correlated variables with lots of potential for feature selection. For instance, one feature is sufficient to describe two perfectly correlated vari-

ables, since the extra variable provides no additional information about the class. These excessive variables might even serve as noise for a algorithm, as it might introduce bias and therefore reducing the performance [38]. An important notion is that correlation does not imply causation [3], but it can serve as a guide to further investigate the effect of a feature.

One way of selecting relevant features is through filter methods, where the general idea is to give each variable a score and exclude any variable with a score beneath a certain threshold. Pearson correlation coefficient is an alternative for calculating such a score which is lightweight and avoids overfitting. A downside of such as an approach is that variables might be discarded due to a low score even though it could prove valuable in combination with other data [38]. The correlation coefficient can be a good indicator of the strength between two or more variables, but only when a linear relationship exists between the variables [29].

Instances selection

Instance selection is the process of selecting samples from a data set that are capable of resembling the entirety of the dataset but on a smaller scale. A new dataset containing representative samples will result in a reduction in height in regards to the data used for machine learning [12]. This is similar to dimensionality reduction as some instances are a better aid for the learning process than others. Blum and Langley [39] mentions the following reasons why this is so:

- Reduction of computational complexity
- Labels for the samples could be expensive (e.g. when manually constructed by experts)
- Focusing the learning process on informative examples

Possible approaches to this selection are random selection, genetic algorithm-based selection, progressive sampling, using domain knowledge, and cluster sampling [12]. The size of the re-sampled data sets has also to be put into consideration with a balance

between accuracy and computing time, and the selection approach should ensure that all output classes are included [12]. It is important to remember that there is a trade-off when it comes to data size. In other words, how much can the data set be reduced in the number of training samples before performance drops? Contrary, a simple, pragmatic solution to bad model performance is for instance possibly just getting more data [3].

Feature scaling

Many machine learning algorithms perform poorly if the features are using very different scales. Feature scaling is, therefore, one of the most important parts of the data preprocessing [5, p. 66]. For instance, in the context of Braluft pressure is usually around 1000 hPa and wind speed below 10 m/s resulting in potential struggles for the machine learning models due to the difference in scale. Scaling the label of the data is usually not required. Standardization is a commonly used feature scaling strategy, where the mean value of a feature in the data set is subtracted from the feature value in the training sample, and then divided by the variance [5, p. 66].

5.2.2 Learning

The learning phase includes selecting appropriate algorithms and tuning the learning parameters to create a model based on the preprocessed data.

There are many machine learning algorithms available with a great deal of diversity, something that reflects the different needs within the applications in regards to capturing the mathematical structures in the data, offering explanations, and providing alternatives for the trade-off between computational complexity and performance [37]. Selecting the appropriate algorithm is often considered more an art than a science since there is no single model that performs best on all problems [13]. In addition to this, models usually have the same fundamental strategy: grouping similar examples, where similar is the variance provided by the individual algorithms. Therefore, Domingos [3] suggests to start with the simplest algorithms. The learning parameters of the models may also affect the performance significantly meaning proper configuration is crucial.

Unfortunately, most machine learning systems are not providing assistance in this area [28].

Machine learning algorithms

There are way too many algorithms available to mention here, so only the three algorithms being tested by the Braluft-program are described in this section: *Passive-Aggressive Regressor (PAR)*, *Stochastic Gradient Descent (SGD)*, and *Neural Networks (NN)*.

PAR and SGD are variants of Linear Regression, meaning the models make predictions by computing a weighted sum of the input features plus a bias term. This is more formally written as: $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$ where y is the predicted value, θ is the vector of trained model parameters, x is the vector of feature values, and θ_0 is the bias term [5, p. 106].

Passive-Aggressive regressor The Passive-Aggressive Regressor is part of an online learning algorithms family for various prediction tasks, including classification, regression, and sequence prediction [30]. It is trained one instance at the time by initially making a prediction of the target value which is the dot product of an internal parameter vector and the feature values of the training instance. The true target value of the training instance is then revealed to the algorithm which suffers an instantaneous loss calculated by the chosen loss function. The learning parameter *epsilon* controls the sensitivity of prediction mistakes by considering the loss zero if the prediction mistake is smaller than *epsilon*. At the end of the training process for the instance is the weight vector updated using the loss function and the training instance [30].

Epsilon is in other words responsible for defining when to update the model. The weight vector will remain the same if the prediction error is less than *epsilon* meaning the algorithm remains *passive* for the given training instance. Contrary, if the prediction error is larger than *epsilon* the algorithm will be *aggressive* to change the weight vector [30].

The Passive-Aggressive algorithms have a few variations of the objective function for weight vector optimization. Some of these include a regularization term C which defines how large steps the algorithm may take upon updating the weight vector [30].

Stochastic Gradient Descent Gradient descent is an optimizing algorithm attempting to minimize a cost function in order to find an optimal solution. This is performed by updating a parameter vector by measuring the local gradient of the error function and moving towards the descending direction one step at the time [5, p. 117]. Once the gradient is zero the algorithm has reached a minimum. The size of each step can be determined by setting a *learning rate*.

The algorithm has various different implementations, including Batch Gradient Decent and Stochastic Gradient Descent. The former is based on using an entire data set to compute the gradients at every step. SGD on the other hand is only making use of one training sample when calculating the gradients per step. This result in a cost function bouncing up and down, but decreasing on average over time [5, p. 117].

Neural networks While neural networks are available in various types for different tasks this thesis is focusing on using a *Multi-Layer Perceptron (MLP)* architecture. A central building block for these neural networks are the *linear threshold unit (LTU)* which serves as an artificial neuron in the network. The LTU takes a set of numbers as input which are turned into a weighted sum. At the end a step function is applied to the sum which is the output of the unit [5, p. 257].

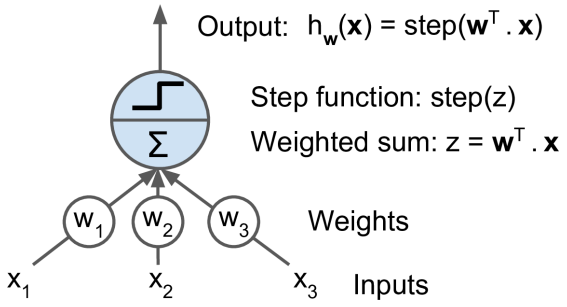


Figure 5.3: Linear threshold unit (LTU) [5, p. 257]

A MPL consists of one input layer, one or more *hidden layers* of LTUs, and a final layer of LTUs called output layer. Networks with two or more hidden layers are called deep neural networks [5, p. 261].

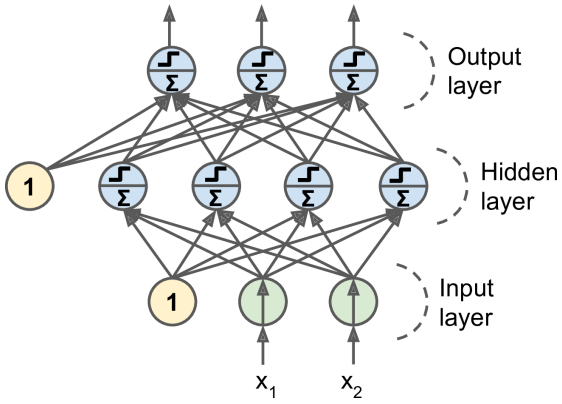


Figure 5.4: Multi-Layer Perceptron [5, p. 261]

The networks are trained using back-propagation, which includes making a prediction using the network, measure the error, then go through each layer of the network in reverse order to measure how much each of the connections in contributed to the overall error, and finally tweak these connection weights to reduce the error.

5.2.3 Evaluation

An evaluation of the model is the last step of the process, where the performance of the model is determined [28].

Regression metrics

The objective of the metrics for regression problems is to measure the distance between the predicted value and the actual target value, which may say something about how much error the models are making in their predictions [5, p. 37-39]. While several metrics could be included in this category, only four are presented as they are used to evaluate the machine learning models in the program. In the samples m is representing the number of training examples, $x^{(i)}$ is the vector of the values of the features of training

sample number i, and h represents the predictive function of a machine learning model.

Coefficient of determination (r^2) The coefficient of determination is a measurement of how likely it is that future samples are correctly predicted by a model. The best possible score is 1 and it can be negative [40].

Root mean square error (RMSE) The root mean square error emphasizes large errors by squaring the prediction error, which may be an undesired property if there are many outlying values that should be ignored [5, p. 37-39].

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h(x^{(i)}) - y^{(i)}\right)^2} \tag{5.1}$$

Mean absolute error The mean absolute error is a simpler metric and is the mean absolute distance between the predicted value and the observed value [5, p. 39].

$$\frac{1}{m}\sum_{i=1}^{m}|h(x^{(i)}) - y^{(i)}| \tag{5.2}$$

Median absolute error As the name suggests median absolute error is the median of a absolute errors made by the model when comparing predictions with the actual values. A potential advantage of using the median is that outlier data are ignored.

5.3 Machine learning & air quality forecasting

Stochastic multiple linear regression and neural networks have previously been used to predict the concentration of air pollutants with success [18].

Forecasting models based on multiple linear regression with meteorological variables have previously been developed and tested with data from Helsinki and Athens. The target for the study was to forecast the maximum hourly concentration of PM_{10} and NO_X, as well as the daily average, for the following day. The latter was concluded to be the easiest task to model out of the two, as anomalies were smoothed by more

predictable observations. The challenge of seasonal variations was handled by making separate models for cold and warm periods of the year [18].

A two-day pollutant forecast with good predictive abilities has previously been built using an Elman Model based on a recurrent neural network to forecast the occurrences of different components in the city of Palermo (Italy), including PM_{10} and NO_2. Variables used for the models were wind speed and wind direction, pressure, and temperature. [20].

Neural networks have previously been built for PM_{10} forecasting targeting the following day in urban areas in Belgium. The study concluded that meteorological conditions were the main influencer of the PM_{10} concentration, with boundary layer height as the most important variable. Anthropogenic activities, on the other hand, had a smaller effect. Contrary to much similar work wind speed did not provide a significant role in the accuracy of the model [19].

Neural networks and lazy learning have been tested for ozone and PM_{10} forecasting for the current data in the city of Milan using air quality and meteorological data with promising results. The best estimation parameter for PM_{10} was the previous observation, with less emphasis on meteorological variables [41]

A forecasting system for making NO_2 forecasts 24-48 hours in advance at four different locations in Ireland has been developed using a model based on multiple linear regression, historical NO_2 observations, and meteorological data. Wind speed and direction were found to be a significant associated with the emissions levels at the three different sites [16].

Chapter 6

Architecture

The Braluft system is based on a microservice architecture consisting of four microservices spread across three virtual machines. Each microservice is a stand-alone application sub-unit. The communication between each of them is based on HTTPS and REST-inspired APIs. This approach leads to flexibility as each service can be developed and redeployed separately without affecting the other services [42]. More traditional "monolithic" approaches are on the contrary dependent on a full redeployment of the entire code base for even small changes. The microservice architecture is consequently more lightweight which enables easier deployment during updates and is well suited for situations where it is difficult to anticipate all functionality in advance [43].

The virtual machines are hosted on UH-IaaS, a collaboration between the universities in Bergen and Oslo offering cloud computing to members of various research organizations including the University in Bergen [44]. They are running with identical specifications and operative system (1 VCPU, 4GB RAM, 20GB hard drive, Ubuntu 18.04).

This section will provide a more detailed description of the data sources and how they fit into the overall architecture of Braluft. The three virtual machines are each given an ambiguous name to make them distinguishable, *Wilhelm*, *Thorvald*, and *Ragnvald*. The intent behind the ambiguity is to enable easy moving of services if needed. The following list shows an overview of the architecture with a summary of the different

services hosted by the different machines.

- **Wilhelm**

  Main service

  Responsible for the communication between all the services through daily routine operations, data persistence, and serving as an API for the front end. Everything runs in other words through the main service.

  Source Service

  Enables gathering and formatting of data from external APIs and hosting these data so the main service can access them.

  PostgreSQL

  A relational database for data persistence.

  Front-end

  Static file hosting of front-end resources.

- **Thorvald**

  Model manager

  Responsible for handling the machine learning models that are a part of the application. This includes training models and making predictions using data provided by the the main service and hosting various utility functions.

- **Ragnvald**

  Image service

  The image service is responsible for downloading web camera images overlooking the intersection on Danmarkplass, detecting the number of vehicles in each image, and providing these data to the main service.

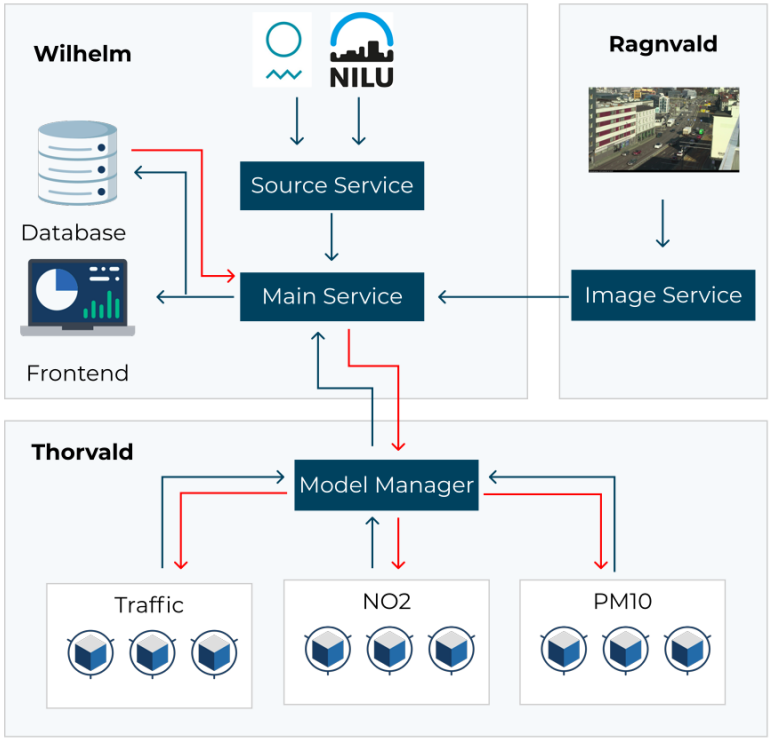


Figure 6.1: The architecture behind Braluft

As the figure illustrates everything runs through the main service. The Source Service hosts Weather forecasts / observations from the API's of Norwegian Meteorological Institute and air quality observations from NILU. Observational traffic data are created by the Image Service using web camera images. The main service collects these data and saves them in the database. The main service is also sending the observational data to the model manager to train the underlying machine learning models in the program, as well as making sure forecasts are made based on weather forecasts originating from the Source Service and traffic forecasts from the Model Manager.

6.1 Source Service

Source service is one of the four microservices that makes up the Braluft program. It is written in Python as a Flask application and is responsible for serving *weather observations, weather forecasts, and air quality observations* at source.braluft.no.

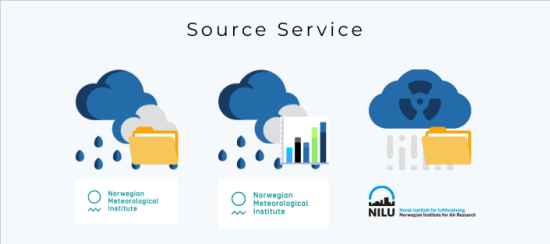


Figure 6.2: Responsibilities for the Source Service
Weather observations, weather forecasts, and air quality observations

Weather data

Both weather forecasting and weather observations for the application are delivered by Norwegian Meteorological Institute and their APIs. The meteorological variables included in the program are, as mentioned before, humidity, pressure, precipitation, temperature, wind speed, and wind direction.

Weather observations are gathered from a service called Frost hosted by Norwegian Meteorological Institute (frost.met.no). The service is an API providing access to historical weather and climate data from various weather stations. One of them is located at Florida approximately 750 meters away from Danmarkplass that the program makes use of. The requested time period, weather station, and meteorological variables are specified using GET-parameters to the Frost API and the response is formatted as hourly intervals. The data are then aggregated into mean values for the *intervals* used by Braluft (6-hour), with the exception of precipitation which is summed and wind direction which is using the median. These data are served by the Source Service in the format viewed by Listing 1.

```
[

    {
        "0": {
            "air_temperature": -0.55,
            "precipitation": 0.2,
            "relative_humidity": 77.67,
            "surface_air_pressure": 990.42,
            "wind_from_direction": 141,
            "wind_speed": 3.82
        },
        // Intervals starting 06, 12, and 18 omitted
}

]
```

Listing 1: Weather observations example

Weather Forecasts Weather forecasts are also delivered by Norwegian Meteorological Institute, using the Locationforecast 1.9 module available as a part of their standard API (api.met.no). Locationforecast does only include the current forecasts meaning no historical data are available on the endpoint.

To solve the need for historical forecast data for development, database restarts, documentation, and other purposes, the weather forecasts are saved as XML-files on the virtual machine hosting the Source Service every night at 1:30 AM CET. The location of the forecast is specified using latitude and longitude as GET-parameters and a nine-day forecast is returned in XML-format by the origin.

When the Source Service is queried for weather forecasts for a specific day a Python script is used to look through the folder containing the downloaded forecasts for relevant hits from the preceding seven days of the date in question. A query for the date 2019-01-10 will, for instance, look for forecasts with a date between 2019-01-03 and 2019-01-09 and returned in JSON-format as shown in Listing 2.

```
[
    {
        "forecastOrigin": "2019-01-09",
        "intervals": {
            "0": {
                "humidity": 95.82,
                "precipitation": 1.7,
                "pressure": 1024.33,
                "temperature": 1.47,
                "windDirection": 142.4,
                "windSpeed": 3.35
            },
            "6": {
                "humidity": 95.8,
                // Data omitted
            },
            "12": {
                // Data omitted
            },
            "18": {
                // Data omitted
            }
        }
    },
    // Forecasts between 2019-01-03 - 2019-01-08 omitted
]
```

Listing 2: Weather forecast example for 2019-01-10

The original XML-data are stored in various intervals by the API depending on how close the forecasted data are to the origin date of the forecast. The nearest dates have data in an hourly rate, while later dates come with data in six-hour intervals. Similarly as the weather observations the forecasted data are returned from the Source Service as mean values, with the exception of precipitation and wind direction, using the program's *intervals* of six hours.

Air quality observations

Air quality observations are the final data hosted by the Source Service. The data are delivered by Norwegian Institute for Air Research (NILU) through their publicly available APIs located at api.nilu.no. The Source Service sends requests to the API specifying requested time range and observation site using GET-parameters. The API sends a response containing data related to several observed pollutants, including NO_2 and PM_{10}, in hourly intervals for the requested time range. Finally, the Source Service is

aggregating the data into the 6-hour *intervals* used by the program as shown in Listing 3.

```
{
    "0": {
        "NO2": {
            "max": 12.55,
            "mean": 6.93,
            "median": 6.69,
            "min": 2.53
        }
        // Other pollutants for interval omitted
    }
    // Remaining three intervals omitted
}
```

Listing 3: NO_2 for the 00-06 interval for an arbitrary date

6.2 Image Service

Figure 6.3: Responsibility for the Image Service

The Image Service downloads web camera images overlooking local traffic and counts the number of vehicles in the images

Image service is a separate service responsible for getting the traffic observations used in the Braluft program. It runs on its own virtual machine for dedicated resources and scalability. Briefly explained the Image Service is responsible for measuring the traffic in images from a web camera hosted by Bergens Tidende overlooking the intersection on Danmarksplass. The measurement is based on how many vehicles are being detected by an object detector in the images. Figure 6.4 is an example image from the web camera stream.

Figure 6.4: Imaged captured from web camera on Danmarksplass

Image downloading strategy

The service makes no attempt to provide an exact number of passing vehicles in the intersection, but rather provide numbers that are able to represent the amount of traffic. It is therefore based on snapshots rather than the continuous video stream. This leads to a significant reduction in required computational power compared to trying detecting vehicles in a video stream. Additionally, the web camera stream is based on many short video clips resulting in challenges for the object detection approach since the vehicles are likely to appear in several clips. Using snapshots is therefore a lot more feasible and requires much less engineering.

Conveniently, snapshots from the stream are already being hosted publicly by the web camera provider in JPG-format which is updated every 10 minutes. Braluft makes use of these snapshots when making traffic observations. The virtual machine hosting the Image Service downloads the latest image every 10 minutes and saves them in a backlog folder.

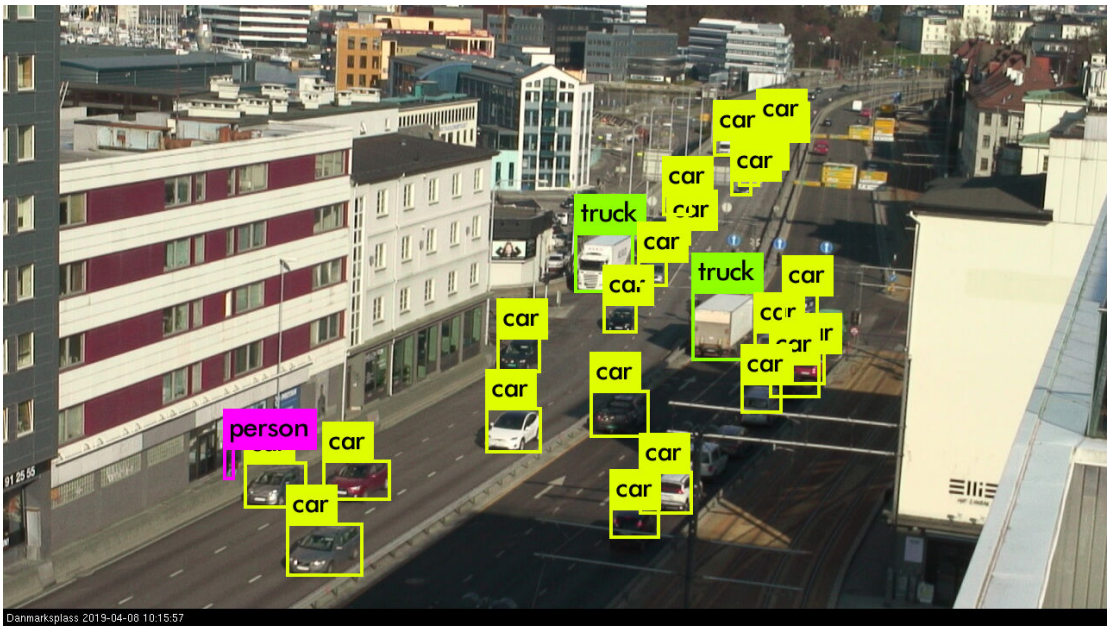


Figure 6.5: Processed web camera image by YOLOv3

Object detection strategy

The Image Service is taking a detection-based approach using *You Only Look Once version 3 (YOLOv3)* for detecting vehicles in the web camera images. YOLO is a pre-trained machine learning model based on a convolution network capable of predicting several objects simultaneously in an image.

The algorithm divides the image into a grid with several different bounding boxes per cell. Each of the boxes is assigned an individual confidence score representing how certain the model is that an object exists within the box and how accurate the perimeter of the box is [45]. The boxes are also predicting class probabilities describing how likely it is that various classes exist within the box [46]. The final prediction values are the product of the confidence scores for the bounding boxes and the class probabilities. The technique is very fast compared to more traditional classifier-based methods since YOLO only requires a single network evaluation [45]. The convolutional network used by YOLOv3 has 53 layers and is called Darknet-53 [46]

YOLO has previously proved to be capable of performing better on traffic congestion

classification than regular deep convolutional neural networks [33] while simultaneously being quick because of the *only look once* approach. In addition to traffic congestion, YOLO (or modified versions of YOLO) is also tested in a variety of other tasks within traffic contexts, such as traffic light detection [47], pedestrian detection [48], and road lane detection [49].

The object detection phase is initiated once per hour where the downloaded web camera images in the backlog folder are processed using YOLO. This is performed by a script that loads the object detector and iterates over the six images downloaded the past hour. The object detector returns a list of objects for each image which is reduced to an integer representing the sum of vehicles. The vehicle count and the filename are at the end saved locally as files in the format seen in Listing 4

```
[
    {
    "cars": 38,
    "file": "2018-12-20T13:46:23.jpg"
    },
    // Remaining files omitted
]
```

Listing 4: Traffic observations in JSON-format

Precision

Targeting 100% accuracy is not a priority for the object detection process in the Image Service. It is more important that the object detection is consistent for all the images and that the detector is capable of creating numbers suited for traffic representation. The YOLO object detector can be controlled by setting a threshold value defining how certain the detector has to be that a certain object exists in order to report it as detected. This threshold was configured in the Image Service through many experiments performed during development using images with different context, such as a varying amount of vehicles and time of day (day/night). The ideal threshold value was concluded to be 20% leading to the most detected vehicles without false positives and other errors.

Figure 6.6: Image Service process

Web camera images are downloaded every 10 minute into a backlog folder. Once per hour the images in the backlog folder are processed by counting the number of vehicles in the images. The results are saved as local files on the virtual machine hosting the service.

Accessing the processed data

The processed data are available through HTTPS using a Flask application written in Python. GET-parameters in the HTTPS-query are used to specify the time period, for instance, `https://images.braluft.no/?from=2018-12-20T14:00&to=2018-12-20T15:00` will get all data related to images between 14 PM and 15 PM, 20 December 2018. The service will then query the local files for file names that are in the requested time range before the data are returned in JSON-format as seen in listing 4.

6.3 Model Manager

A typical approach to making machine learning models is to make several variants and select the model with the best performance. However, the best performing model might vary as time passes and the problem space changes [16]. In addition to this there is as previously mentioned no model that performs best on all problems.

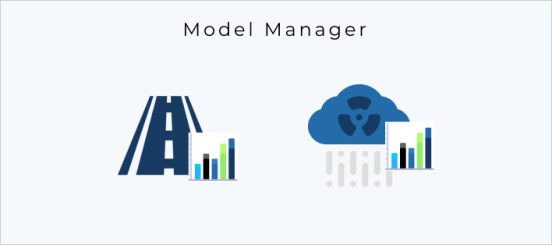


Figure 6.7: Responsibilities for the Model Manager

The Model Manager is handling all the machine learning models used by the program to create traffic forecasts and air quality forecasts. This includes initiating and training the models, and using the models to create forecasts. All the actions are performed upon request by the Main Service.

This challenge adds extra complexity in terms of building models and infrastructure for the Braluft program since:

- The incremental nature of Braluft means hardly any data are available to analyze at the initial period for the project. This makes it difficult to gain intuition on what a good model looks like for the project

- Even with this intuition it is difficult to foresee which model will perform best months or years ahead
- Several models increase the need for a uniform way of training models and making forecasts
- The program is dependent on regression models for three different tasks:
  - NO_2
  - PM_{10}
  - Traffic

The model manager is implemented to combat these issues by providing an interface for effortless model deployment, making it easier to explore new models in larger quantities.

Model structure

The general idea behind the Model Manager is that every model controlled by the model manager must conform to (a) a specific file structure where each model has a folder containing one script file and one file for the serialized model, and (b) the script file must implement a train and a predict function.

The file structure is illustrated by Figure 6.8 showing how models are grouped by task and identified by using the folder names as model names. This structure enables the Model Manager to find the correct model for the correct task.

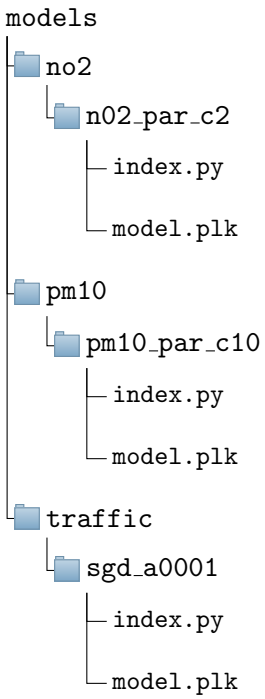


Figure 6.8: File structure for models

All the models are grouped into folders by task. Each model has a separate folder where the folder name is used for identification and the folder must contain an index.py file with the actual implementation of the model. The serialized models are stored as model.plk

The index.py file in the model folders allows the Model Manager to communicate with the models in a uniform way. These files hold the actual implementations of the models and must therefore implement a constructor creating the serialized model if it does not exist, a train function, and a predict function. The implementation of the functions is entirely up to each individual model, meaning they can take advantage of different frameworks and libraries.

Using the Model Manager

The model manager's role is to keep an updated list of available models and making sure data are sent to the right model for training or for making predictions. The model manager is not responsible for selecting which **intervals** should be sent to training or

HTTPS POST request -> https://mm.braluft.no/**train**


```
{
"data":{
"fromDate":"2019-01-01T18:00:00+01:00",
"toDate":"2019-01-02T00:00:00+01:00"
,"weather":{"precipitation":0.0,"wind_ms":9.2,"temperature":3.83,"max_wind_speed":12.
6,"humidity":61.17,"wind_direction":331.0,"pressure":1023.13},
"airHistoric":{"PM10":14.21,"NO2":16.05},
"traffic":142},
"model":{
        "id":105100,"name":"k_s_a001_20_mse","type":"NO2"
        }
}
```

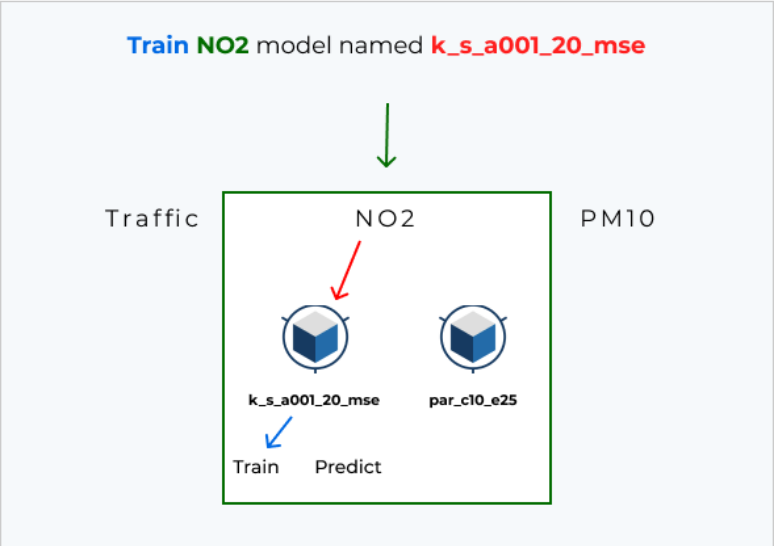



Figure 6.9: Training a model using the model manager

The Model Manager receives a POST-request to the train endpoint. The model type and name are specified as a part of the request body so that the Model Manager may find the correct model to train. The other part of the request is the interval data that are being sent to the model for training or predicting.

making predictions, nor keeping track of which **intervals** have been used for training of the individual models. The model manager knows, in other words, nothing about the general state of the program or its **intervals**.

The Model Manager is running on a separate virtual machine as a Flask application with two endpoints accessible through HTTPS using the POST-method, /train and /predict. These endpoints decide which action to perform. The message body of the request must contain two JSON-objects. The first one is a JSON-object with the name and type of the model. Upon receiving a request the model manager attempts to find the model with a matching name and type using the file hierarchy. The other object in the request is the **interval** data that include observational data if the action to perform is training, or forecasting data if new forecasts are to be made. These data are forwarded to the selected function (train / predict) in the specified model. This process is shown in Figure 6.9. If the performed action is to make a prediction, the response from the Service is the predicted value provided by the selected model.

6.4 Main Service

The last service is responsible for communication between the other microservices, a relational database, and the front end. Most of the active actions in the program are initiated or go through the Main Service. Key responsibilities of the service include:

- Defining a domain model
- Data Persistence
- Creating **intervals**
- Making sure upcoming **intervals** are provided with forecasts
- Fetching historic data for the past **intervals** and sending them to the model manager for model updates
- Providing endpoints used by the front-end and for statistics

The service is written in Java 1.8 using the framework Spring Boot, with the assistance of Maven for handling dependencies and building.

Domain model and data persistence

Spring Boot was chosen for this task for several reasons, including the Spring Data JPA (Java Persistence API), static types, and sensible default configuration. The domain model of the service is defined using annotated Java classes to describe entities and their relationships. The model is mapped to a relational database (PostgreSQL) hosted on the same virtual machine as the service. This process is made possible with some assistance from the Object-relational mapping tool Hibernate which has a lot of utility related to saving data and making queries, making the data persistence a lot simpler with reduced development time.

Daily routine

The Main Service's most important job is the daily routine performed 2 AM every night. All the data manipulation for the service is performed in a single run preparing for the next day. The procedure is performed in the following order:

1. Synchronize the list of available models with the Model Manager
2. Create **intervals**
3. Add data to the **intervals** (except traffic forecasts and air quality forecasts)
4. Train traffic models and make traffic forecasts for upcoming **intervals**
5. Review all the traffic models
6. Train air quality models and make air quality forecasts for upcoming **intervals**

Model synchronization The initial step of the nightly procedure is synchronizing available models. The Main Service is notified about any changes in the models hosted by the Model Manager and makes sure new models are saved in the database, and

models no longer hosted by the Model Manager are deleted. Models for all tasks are taken into account during this step (traffic, NO_2, and PM_{10}).

Creating intervals After the models are synchronized the Main Service is creating the four new **intervals** one week ahead relative to the current date. For instance, if the current date is 01.01.2019, the new **intervals** are dated 08.01.2019.

Getting data The next step is to add data of the following types to the **intervals** missing data:

- Observational weather data (Source Service)
- Observational air quality data (Source Service)
- Weather forecasts (Source Service)
- Observational traffic data (Image Service)

Which data to add are depending on how a given **interval** is related to the current date. Only passed dates can get observational data. Contrary, passed dates do not get any new weather forecasts.

Training traffic models and making traffic forecasts The **intervals** from the previous day are used for training the traffic models, and the newly created **intervals** one week ahead are receiving exactly one traffic forecast per traffic model

Traffic model review The next step is reviewing the traffic models, where the goal is to find the traffic model with the best performance for the past 30 days. The time range was set with the intention of trying to find the model capable of capturing recent traffic trends that might occur based on seasonal changes such as holidays. This model is stored in-memory so that the program know which traffic forecast to use when making air quality forecasts.

Training air quality models and making air quality forecasts The initial part of the step is using using the observational data from the previous date to train the air quality models in the program. Next up is making forecasts for the **intervals** in the following seven days. These forecasts are based on fresh weather forecasts and the traffic forecast made by the model found in the previous step.

6.5 The journey of an interval

To further illustrate how the different services within the program are communicating with each other, the following section is dedicated to describing the entire journey of a single **interval** object. All the steps in the process include saving data in the relational database and is therefore not explicitly stated as a part of each step. Table 6.1 describes the creation of an *interval* starting *08.01.2019 00:00*.

Day: 1 Date: 01.01.2019
1. The **interval** object is created by the Main Service. 2. The Main Service initiates the collection of weather forecasts for the **interval** from the Source Service. 3. The Main Service sends the **interval** to the Model Manager n times for traffic forecasts, where n is the number of traffic models available. 4. The Main Service has previously calculated which traffic forecasting model performs the best. The traffic forecast made by this model for the **interval** is sent, along with weather forecasting data, to the Model Manager $k+m$ times for air quality forecasting, where k is the number of NO_2 models, and m is the number of PM_{10} models.
Day: 2-7 Date: 02.01.2019 - 07.01.2019

Performed every day: 1. The Main Service initiates the collection of the latest weather forecast for the **interval** from the Source Service. 2. Step 4 from day one is repeated by sending in weather forecasts and traffic forecast to the Model Manager for air quality forecasting. The selected traffic forecast used may differ every day if the best performing traffic model has changed.
Day: 8 Date: 08.01.2019
No action performed related to the **interval**.
Day: 9 Date: 09.01.2019
1. The Main Service initiates the collection of observational weather and air quality data for the **interval** from the Source Service. 2. The Main Service initiates the collection of observational traffic data for the **interval** from the Image Service. 3. The Main Service sends the observational data for the **interval** to all models managed by the Model Manager (traffic, NO_2, and PM_{10}) for training.

Table 6.1: The journey of an interval

6.6 Front end - braluft.no

While the microservices form the fundamentals of the Braluft program, they offer little usability and insight by themselves. This is of course because the microservices are only available through HTTPS using, for the time being, undocumented endpoints in a JSON-format. A separate front end is therefore built to represent the processed data provided by the Main Service's endpoints. It is intended to show historical data, forecasts, and different types of charts, such as line charts comparing observed data to forecasts over time, performance metrics for models over time, etc. The responsibility

of the front end is in other words to:

- Show forecasts made by the program
- Show data points for traffic, weather, and air quality to easily detect missing data
- Monitor the models

Implementation

The front end is a *Single-page application* written in Javascript, based on the React library developed by Facebook. With the exception of React there are too many libraries included in the development of the front end to go into details of all, but two worth mentioning are *redux* and *chartjs*. *Redux* is responsible for containing the application state of the frontend, mainly data collected from the Main Service's endpoints. *ChartJS* is simply put the library providing charts to the frontend. All the source files for the front-end are bundled upon deployment and hosted as static files accessed at braluft.no.

Visualisations

The front end is a utility created to make sure data are collected properly and monitor the performance of the models. The latter task is however difficult, as data can be too complex to interpret, often require processing. To overcome this issue, visualizations are integrated into the front end making the metadata related to the models easier to interpret. Furthermore, this can improve the domain knowledge, by looking at how the variables correlate and finding patterns in the data that can be used in feature engineering [13]. Interesting results that are derived from this process may even be considered a part of the artifact itself if the visual representations can provide new information about the air quality.

Chapter 7

Exploring the data

This chapter is dedicated to exploring the observational data gathered as a part of the Braluft-program. More precisely the relationships between the traffic level, meteorological variables, and the pollutants are examined using visualizations of the data along with the Pearson correlation coefficients. The data set created for this section is based on observations made between December 2018 - May 2019 and consists of 604 *intervals*. Each of the *intervals* can be considered an observation for the variables. There are in other words only observations gathered during winter and spring, and future data collected during the summer or fall may very well look different. This limitation should be accounted for when reviewing the data.

Traffic

Using traffic level for air quality forecasting appears to be useful for especially NO_2, and to a certain degree also for PM_{10}. The most important conclusion based on the patterns seen in Figure 7.2 is probably that low to nonexistent traffic levels result usually means good air quality considering these observations are clustered to the bottom left. However, high traffic levels do not necessarily mean bad air quality given the scattered observations when traffic levels are rising. The increase seems to be somewhat clearer for NO_2 than PM_{10} since the lowest registered pollutant levels increase as the amount

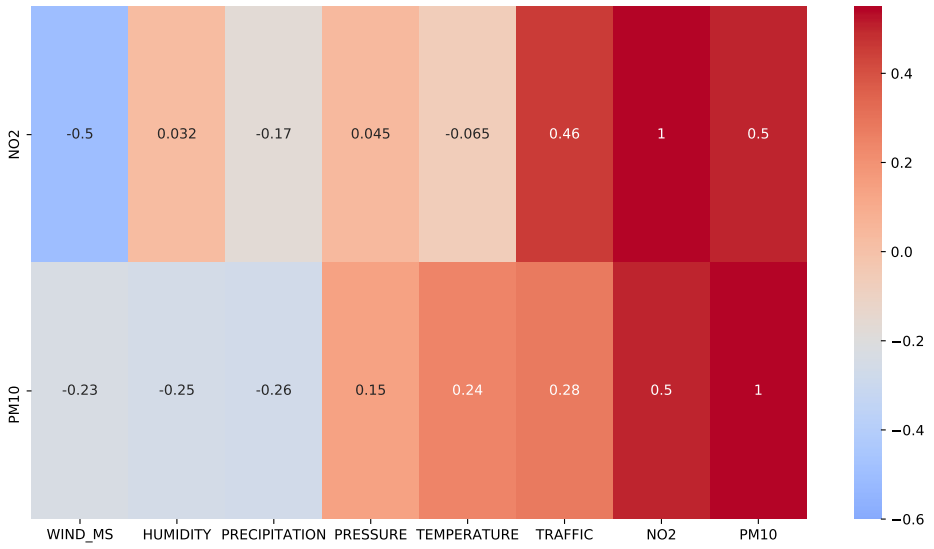
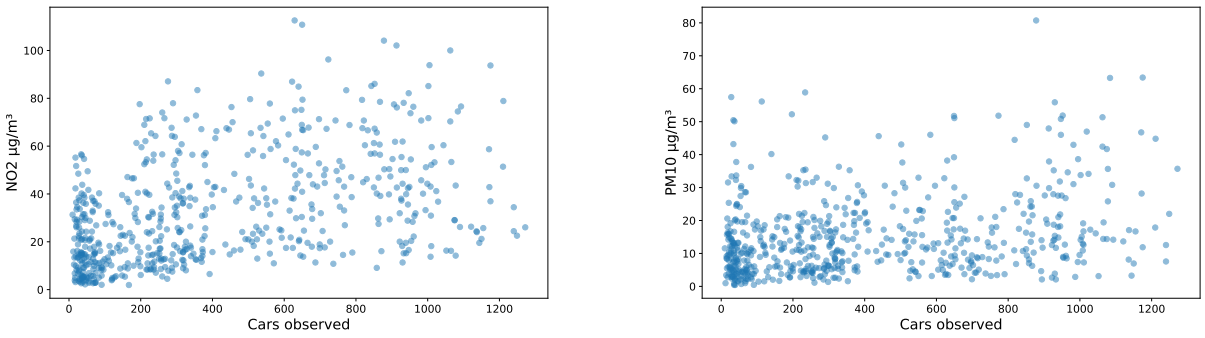


Figure 7.1: Correlation between observed data

of traffic grows. The relationship between the traffic levels and the two observed air pollutants seems to correspond with the coefficients in Figure 7.1.



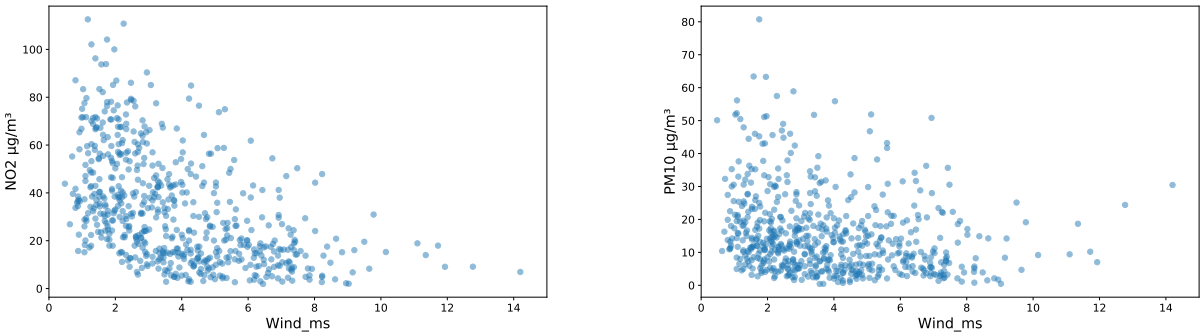
(a) NO_2

(b) PM_{10}

Figure 7.2: Traffic

Wind speed

Based on the data gathered by Braluft wind speed is the most influential meteorological variable given the visible decrease in pollutant levels as wind speed increases. Similarly as traffic level it does seem like wind speed has a greater effect on NO_2 than PM_{10}. This is illustrated by both Figure 7.3 and the correlation matrix in Figure 7.1.



(a) NO_2 (b) PM_{10}

Figure 7.3: Wind speed

Wind direction

The data gathered by the program show patterns in the wind directions that correspond with the geographical surroundings of Danmarksplass. As of 18 May 2019 604 *intervals* are registered in the program. 464 of these observations have a wind headed towards south-east (210), south (104), or north-west (150).

The remaining directions have many fewer observations registered which is likely because of the valley-like surroundings formed by the local mountains around the observation site, with *Ulriken* to the east and *Løvstakken* to the south-west.

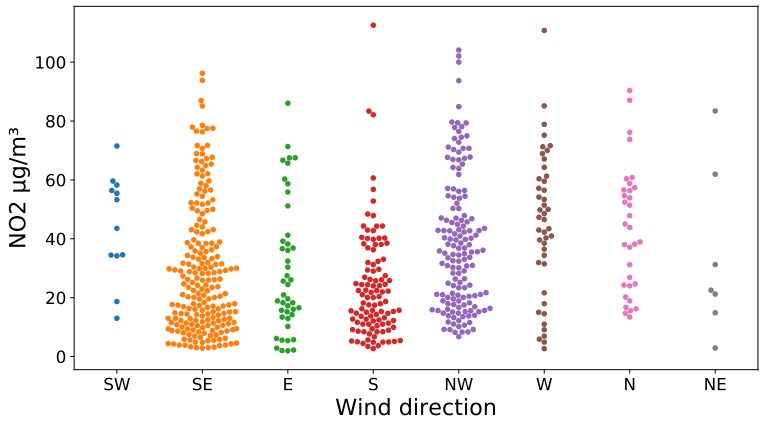


Figure 7.4: NO_2 - Wind directions

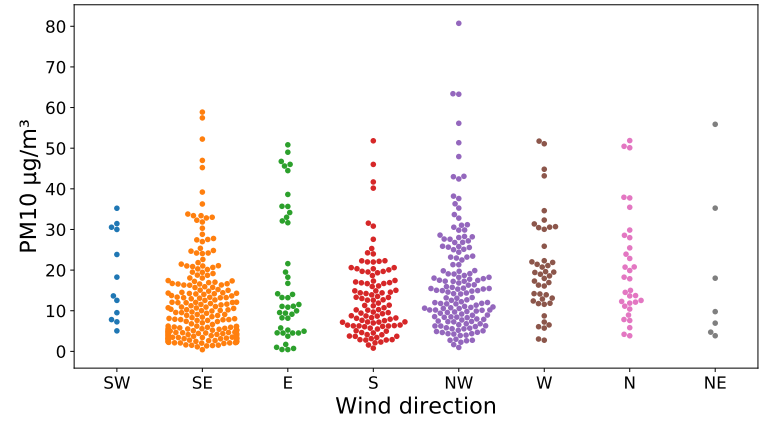


Figure 7.5: PM_{10} - Wind directions

More important, wind direction seems to have an impact on air quality. The general tendency in the observed data is that wind headed towards north-west tend to result in an increase in NO_2 and PM_{10} levels. Contrary, wind headed towards south-east / south leads to lower NO_2 and PM_{10} levels. The complexity behind this observation warrants a separate study, but it is possible that the wind headed north is more exposed to anthropogenic environments than its opposite and is therefore bringing pollutants from

other districts.

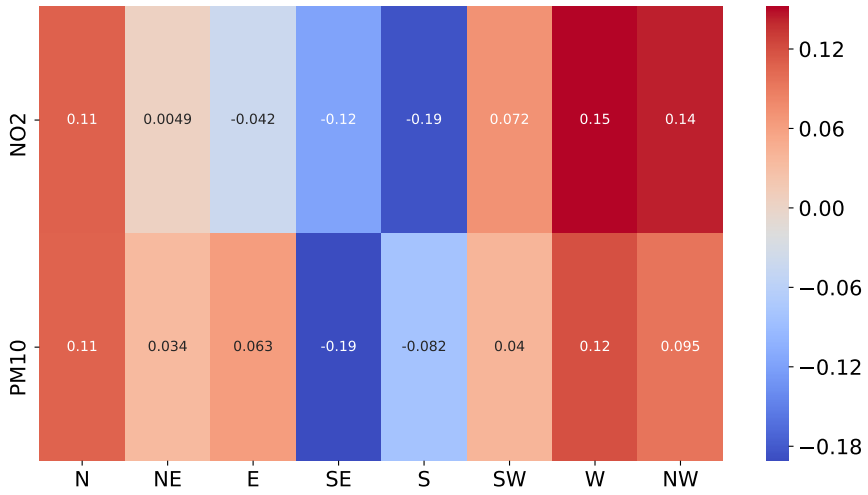
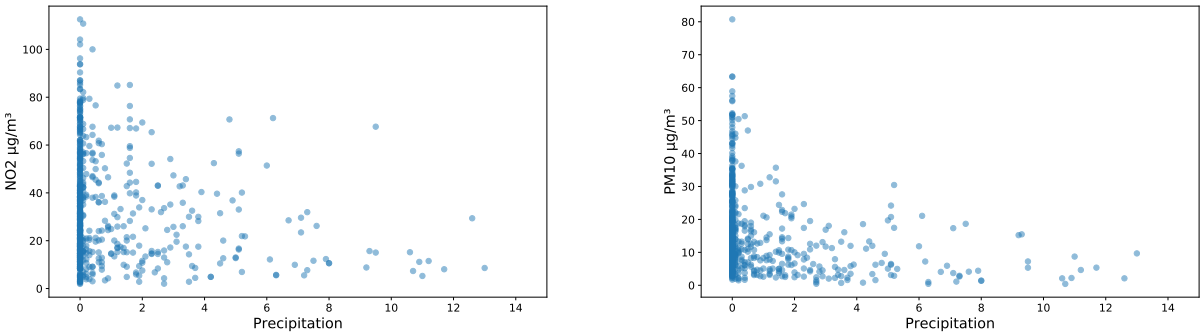


Figure 7.6: Wind directions with NO_2 and PM_{10} correlation

Precipitation

Precipitation seems to have some influence on the observed air quality, especially when considering PM_{10} levels. This could very likely be related to road dust being washed away resulting in less resuspension because of traffic.

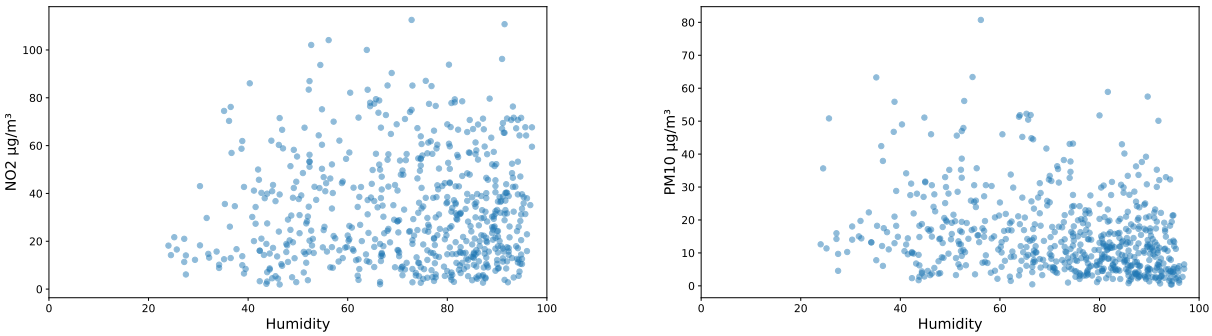


(a) NO_2 (b) PM_{10}

Figure 7.7: Precipitation

Humidity

Given the data gathered by the program there are not any significant signs of a connection between humidity and the NO_2 levels with a Pearsons correlation coefficient of 0.032 and no visible patterns showing in the plotted data. However, humidity does seem to have some association with PM_{10} considering the low pollutant levels during periods with high humidity, but no linear relationship seems to exist.



(a) NO_2 (b) PM_{10}

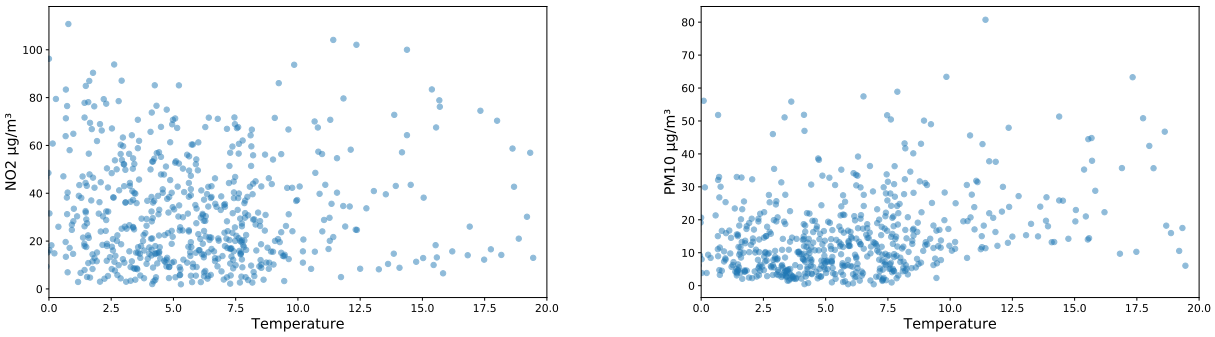
Figure 7.8: Humidity

Temperature

An increase in temperature seemingly has an effect on PM_{10} albeit not much. This relationship is not obvious in Figure 7.9, so a few machine learning models were trained to compare performance with and without temperature as a feature. Including temperature led to a small increase in performance as seen in Table 7.1.

Model	r2	median_abs_err	rmse	mean_abs_err
PAR - with temperature	0.43	5.22	10.28	7.25
PAR - without temperature	0.30	5.97	11.41	8.19

Table 7.1: PAR with and without temperature
24-hour forecasts - Performance April 2019



(a) NO_2 (b) PM_{10}

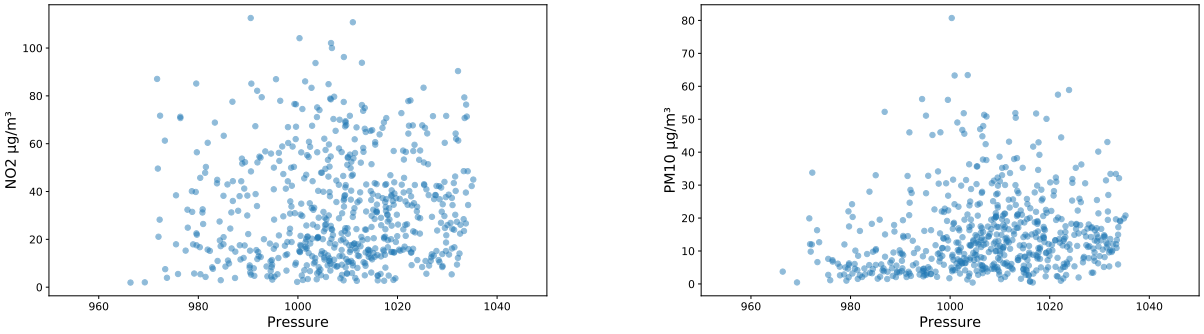
Figure 7.9: Temperature

Pressure

The least relevant feature available in the program was concluded to be pressure. This is based on a low Pearson correlation coefficient, no obvious patterns seen in figure 7.10, and by creating some sample models as seen in table 7.2. Unlike temperature exclusion of pressure actually increased the performance of the models.

Model	r2	median_abs_err	rmse	mean_abs_err
PAR - with pressure	0.41	5.21	10.42	7.40
PAR - without pressure	0.43	5.22	10.28	7.25

Table 7.2: PAR with and without pressure - Performance April 2019



(a) NO_2

(b) PM_{10}

Figure 7.10: Pressure

Chapter 8

Modelling

The Braluft program is working with three different kinds of machine learning tasks, *traffic,* NO_2, and PM_{10}. All these can be considered regression problems meaning the objective is to produce a numeric value. Previous comparisons between artificial neural networks and linear multiple regressions models in an air quality forecasting context have shown that artificial neural networks perform slightly better. However, simpler regressions models are easier to construct and can often be interpreted in terms of how much each feature contribute to the predictions. Such contributions are usually not made by artificial neural networks using a "black box" approach [18]. In order to represent simple and complex models, both model types are being used for this thesis.

The program is not concerned with making the actual learning algorithms since several high quality machine learning frameworks already offer implementations of various incremental solutions. The objective of this thesis is rather to make use of already existing technology and data to evaluate the concept of incrementally trained models for air quality forecasting.

The program makes use of two different machine learning frameworks. The first is Scikit-learn, providing state-of-the-art implementations of common machine learning algorithms wrapped in a user-friendly interface. The motivation behind the development of the package is the growing need of analysis tools for non-specialist making it suited for beginners [50]. The module does also provide several utility functions for model selec-

tion, evaluation, and preprocessing. In terms of online learning algorithms for regression problems the package provides implementations of *Passive-Aggressive Regressor (PAR)* and *Stochastic Gradient Descent (SGD)*, both tested in this program.

The other pre-built machine learning framework being used is Keras, a high-level *Neural Network (NN)* API capable of running on top of several different backends including Tensorflow. Using this approach has enabled fast prototyping of neural networks as a part of the Braluft-program.

The machine learning related processes in the program are written in Python, a language of a high-level nature that has established itself for algorithmic development and data analysis in scientific and industrial communities [50].

Stepwise modelling The modeling process in the program is split into three steps.

The first step was to explore and implement traffic forecasting models. The outcome of this step was traffic forecasting models and initial insight into how good the selected machine learning algorithms performed considering learning rate, number of observations, etc.

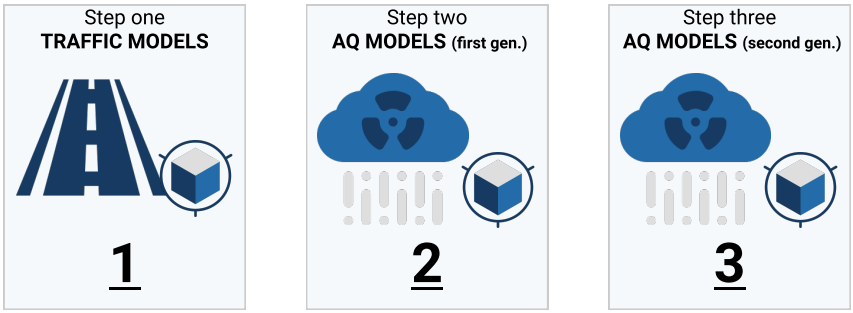


Figure 8.1: Modelling progress steps

The modelling was performed in three steps: (1) Creating traffic forecasting models, (2) Creating the initial air quality forecasting models, and (3) Creating more air quality models based on experiences gained in previous steps for increased performance.

The second step in the modelling process was the implementation of a first generation of air quality forecasting models. These models are characterized by their usage of all

the available variables as input data and being numerous in terms of tested learning parameters. The objective of this somewhat naive generation of models was primarily to gain insights into the data and the prediction task.

The final step was to convert the previously gained insights from the first generation of models, the explored data, and previously reviewed literature into models with presumably better performance. This is mainly done by making adjustments to the set of features eliminating variables with poor predictive abilities.

8.1 The model training process

The three problems the program is concerned with do share a common infrastructure for preprocessing, learning, and evaluation. This infrastructure is not limited to the Model Manger, but is a collaboration between all the services in the program where everyone contributes. Admittedly the Model Manager is the largest contributor, especially if the models are included.

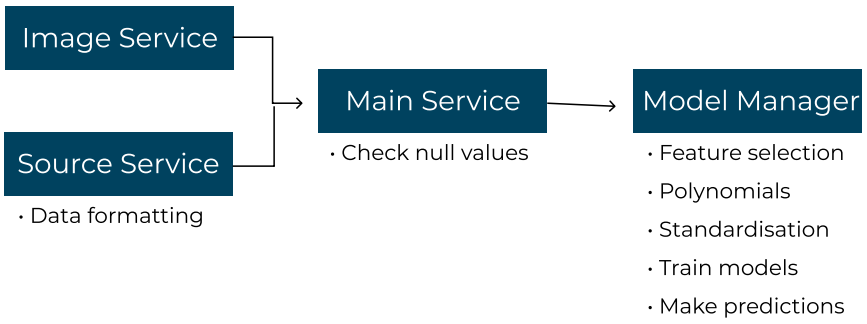


Figure 8.2: Responsibilities in the modelling process

The machine learning processes are not only performed by the Model Manager and the models. It is more accurately a collaboration between all the components making up the program where every part has a contribution. Services related to collecting external data are also performing formatting, the Main Service makes sure no **intervals** missing data are sent to the Model Manager for training / forecasting, and the Model Manager and its models take care of the rest.

8.1.1 Data preprocessing

The greater part of the preprocessing of input data performed by the program is related to data originating from external sources made available by the Source Service. Observational traffic data and traffic forecasts, on the other hand, require little processing since they are created by the program itself using the ideal data format and temporal resolution. The processing performed by the program is mainly done at three different stages: Data formatting performed by the Source Service, handling of missing data performed by the Main Service, and final preprocessing performed by the Model Manager where feature vectors are created among other things.

Data formatting The first part of the preprocessing is performed by the Source Service. Luckily, the external data used are properly formatted through well defined APIs and thus requiring small amounts of processing by the program. Most of the processing performed at this stage is therefore mainly related to formatting the data into the 6-hour *intervals* used by the program and using appropriate aggregation functions for each of the variables as described in the Architecture-chapter.

Dealing with missing data The next preprocessing step is performed by the Main Service making sure the **intervals** are not missing any data before they are sent to the machine learning models for training or forecasting. **Intervals** missing such data are ignored until all the relevant data are present.

Final processing The final part of the preprocessing is performed by the Model Manager. The Model Manager is equipped with three different preprocessing functions for: (a) Traffic models, (b) First generation of air quality models, and (c) Second generation of air quality models. These functions are responsible for performing the actions described in the following sections related to the individual modelling steps.

8.1.2 Learning

All the three selected algorithms (PAR, SGD, and NN) are being tested for each of the three problems in the program. The implementations of the models were usually performed in groups of 5-10, and the lessons learned from each group influenced the next resulting a pursuit of the ideal learning parameters for the algorithms.

The training is performed by uploading the models to the Model Manager and performing a restart of the program. This forces the program to discover the new models and sends the **intervals** for training and forecasting. The specifications of the trained models, such as learning parameters, and the performance metrics were during the development stored manually enabling reproduction of the best performing models.

Learning rate

The selected algorithms have their own ways of controlling how fast the algorithm should learn when exposed to new training examples, which is one of the key challenges of the entire program. In order to illustrate the effect of the learning rate in the context of Braluft, one might consider the following alternatives:

High learning rate

An online learning algorithm with a high learning rate will be capable of adapting to changes rapidly, but there is a large chance that it will forget the experience gained from older training examples [5, p. 16].

Low learning rate

A low learning rate for an online machine learning algorithm will naturally result in slower learning, but it will be less sensitive to noise and non-representative data [5, p. 16].

An important property of the program is the ambition of picking up recent changes within traffic and air quality trends. Old records going several months back should therefore be outweighed by new training examples. The data set is as of April 2019

rather small, meaning the models with a very low learning rate are not able to find the baseline and are likely to struggle when making forecasts. Such models might definitely be valuable as the data set increases in size, but serve little utility for now in terms of evaluating the concept.

That being said, too high learning rate could cause harm as well. Days with poor air quality are being outnumbered by days with good air quality by quite a wide margin, and longer periods with good air quality might occur. During those periods a model with a too high learning rate may forget what a day with bad air quality looks like, and therefore not be able to forecast it.

To evaluate the online learning concept models are equipped with a broad range of learning rates (low-high). Models with very low learning rate are dropped due to the small size of the data set.

8.1.3 Evaluation

A brief evaluation is performed of the traffic models in this chapter. The evaluation and discussions related to the air quality models are saved for chapters 9-10 of this thesis since they represent the overall forecasting abilities of the program.

8.2 Modelling traffic

The first step performed related to making models for the Braluft-program was building traffic forecasting models. A natural choice, considering air quality models are dependent on traffic forecasts in order to make air quality forecasts.

Preprocessing

The Model Manager performs only one operation as a part of the preprocessing for traffic models, namely converting the **interval** data into a vector of binary features. All the features in the vector are features related to time:

- Hours of the day

- Day of the week
- Whether or not the day is a public holiday

The *interval* starting 25.03.2019 at 6 AM will, for instance, be converted to the vector seen in Table 8.1.

00-06	06-12	12-18	18-00	Monday	Tuesday	..	Sunday	Holiday
0	1	0	0	1	0	..	0	0

Table 8.1: 25.03.2019 - 6AM as a vector of traffic data

Learning

A total of 57 models were implemented for traffic forecasting. The training and evaluation of the traffic forecasting models references in this section were based on data accumulated between 18.12.2019 and 05.04.2019. A full overview of the tested models and their performance metrics are available in appendix A.

Evaluation

Several of the models achieved what could be considered acceptable result for its usage, namely making traffic forecasts that serve as input to air quality forecasting models. Using data for March 2019 several models achieve an r^2 score between 0.7 and 0.8, and a **median absolute error** between 80 and 95 vehicles when comparing the predicted values and the observed number. However, the **mean absolute error** for many of these models is around 150 vehicles, which is higher than expected considering that outliers are rare in the data set. These somewhat disappointing results for the mean absolute errors are most likely due to the fact that most of the algorithms are performing well on *intervals* between Monday and Friday, but are struggling with making predictions for the weekend as illustrated by Figure 8.3. The models with a high learning rate are the best at taking weekends into account but make larger errors elsewhere. It is likely that the ideal model has a low to moderate learning rate, but is dependent on more

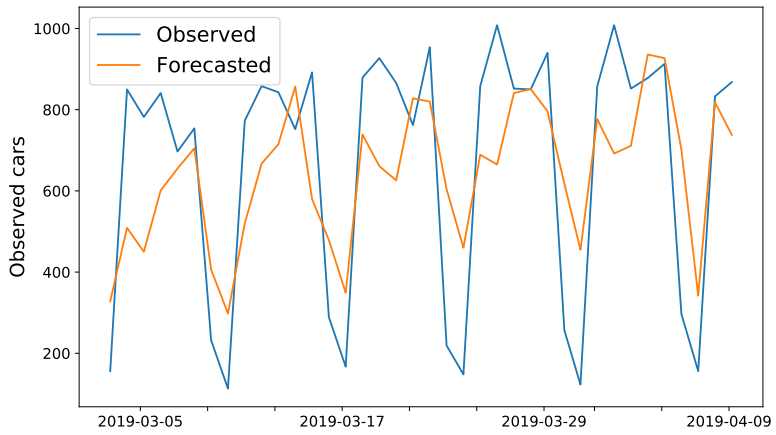


Figure 8.3: Observed traffic levels and traffic predictions by a small neural network
The traffic models perform generally better for the workweek compared to the weekends as illustrated by using data for a few weeks in March and April 2019

data since the traffic levels seem stable week after week, with the exception of days surrounding holidays. Given the incremental design of the program this issue should hopefully diminish over time.

For now the small variants of neural networks have the best performance when considering MAE and the issue of forecasting weekends, with a median absolute error around 70-80 vehicles and mean absolute error around 130 vehicles.

8.3 First generation of air quality models

The primary objective of the first generation of air quality forecasting models was to gain insights into the data and discover the potential of air quality forecasting using machine learning algorithms. The processes described in this section are identical for NO_2 and PM_{10} models with the only difference being the target value.

Preprocessing

The processing of **intervals** headed for training or forecasting with air quality machine learning models is somewhat more complex than traffic data. The process is mainly separated into two parts: (a) feature selection turning the **interval** data from JSON-format to a feature vector (b) scale the data using standardisation.

Feature selection In the first generation of air quality forecasting models all the available variables are included as features. This means all the six meteorological variables being recorded by the program along with traffic level, serve as input data for making NO_2 and PM_{10} forecasts. The variables are represented as continuous numerical features when the data are turned into a vector, with the exception of wind direction which is turned into binary features as seen in Table 8.2.

Pres.	Hum.	Prec.	Tem.	Wind speed	Traffic	Wind N	Wind NE	..	Wind NW
1023.13	61.17	0.0	3.83	9.2	142	0	1	..	0

Table 8.2: Sample air quality data as vector (First generation)

Polynomial features Some of the linear regression models make use of polynomial features of degree 2 or 3 in order to fit nonlinear functions. This conversion is made by the PolynomialFeatures module from Scikit-learn.

Standardisation The next step after the data have been converted to a vector is standardisation. This is performed by the StandardScaler imported from Scikit-learn tuned by using observational data between 1 January 2019 and 8 April 2019. The output of this process is a new vector illustrated in Table 8.3. The binary variables representing wind direction are unaffected.

Pres.	Hum.	Prec.	Tem.	Wind speed	Traffic	Wind N	Wind NE	..	Wind NW
1.25	-0.98	-0.52	-0.23	2.23	-0.80	0	1	..	0

Table 8.3: Air quality vector standardised (First gen)

Learning

25 models were implemented for both NO_2 and PM_{10} forecasting (50 models for the pollutants combined). The training and evaluation of these models were based on data accumulated between 18.12.2019 and 09.04.2019. A full overview of the tested models and their performance metrics are available in appendix B and C.

Evaluation

The evaluation of the air quality models are the subject of chapters 9-10. Performance metrics are also available in appendix B and C.

8.4 Second generation of air quality models

The idea of using all variables as input features for both NO_2 and PM_{10} was meant to be a part of a learning process enabling the creation of more ideal models. This led to a second generation of models where the processes are mainly the same as the previous generation with some alterations that are being discussed in this section.

Preprocessing

The new set of models is initially using the same preprocessing function as the previous generation. That means all the data are turned into a vector consisting of the available meteorological variables and traffic level, and standardised. However, the second generation of models is only using a subset of the features from the original vector. Considering the possible number of combinations given the amount of variables and available algorithms, it is unrealistic to test all possible combinations [19]. The subset of new features is therefore cherry-picked by reviewing the data, literature, and experience gained from the first generation of air quality models.

NO_2

Two of the available variables stand out as obvious selections for the second generation of NO_2 models: wind speed and traffic level. In addition to this, precipitation and a few of the wind directions are also promising candidates. Two variants of NO_2 models were created to explore this further.

Variant 1 The feature vector of the first variant is based on the features mentioned above resulting in a feature vector illustrated by Table 8.4.

Prec.	Wind speed	Traffic	Wind N	Wind SE	Wind S	Wind W	Wind NW
-0.52	2.23	-0.80	0	0	0	1	0

Table 8.4: Air quality vector standardised - Second generation (Var 1 NO_2)

The new feature vector makes use of three features from original vector without any modifications: Precipitation, wind speed, and traffic level. The three wind directions with least expected relevance are no longer represented.

Variant 2 The second variant is a lot simpler by only using wind speed and traffic as polynomial features of degree 2 as seen in Table 8.5

Wind speed (WS)	Traffic (T)	WS^2	$WS \times T$	T^2
2.23	-0.8	4.9729	-1.784	0.64

Table 8.5: Air quality vector standardised - Second generation (Var 2 NO_2)

PM_{10}

Based on the reviewed data in the previous chapter it is less obvious which variables to use as features for PM_{10} models compared to NO_2. Generally speaking all the available variables are slightly associated with increased PM_{10} levels with the exception of several wind directions. However, none of the variables stand out as essential. The selection of features for the second generation of PM_{10} models is for that reason still somewhat

broad, but excludes pressure and most of the wind directions. This results in a feature vector seen in Table 8.6.

Hum.	Prec.	Tem.	Wind speed	Traffic	Wind SE	Wind W	Wind NW
-0.98	-0.52	-0.23	2.23	-0.80	0	1	0

Table 8.6: Air quality vector standardised - Second generation (PM_{10})

Learning

All the three algorithms are still being used for the second generation, but in fewer variants. The new models are based on the experiences gained from the first generation such as the ideal learning rates but with altered feature sets.

Evaluation

The evaluation of the air quality models are the subject of chapters 9-10. Performance metrics are also available in appendix D and E.

Chapter 9

Results

The results in this chapter are based on the most promising models from the first and second generations of air quality models. Additionally, a couple of machine learning models trained using batch-learning on data gathered between 18 December 2018 and 31 April 2019 are also included. This trio of model types are selected to highlight any improvement in performance for the second generation and to compare how well the online learning approach is faring compared to batch learning.

All the results presented in this chapter are based on observational data and forecasts for April 2019. Only the best performing models are selected here, but metrics for the rest of the models are available in appendix D (NO_2) and E (PM_{10}).

Pollutant	Length	Model	r2	median_abs_err	rmse	mean_abs_err
NO_2	24-hour	PAR (C=1, 1st gen)	0.13	13.80	21.26	16.34
NO_2	24-hour	PAR (C=1, 2nd gen)	0.14	11.57	21.18	15.75
NO_2	24-hour	PAR (Wind + traffic)	0.23	11.06	20.01	15.01
NO_2	24-hour	SVR (Batch)	0.21	13.66	20.27	15.97
NO_2	3-day	PAR (C=1, 1st gen)	-0.06	14.09	23.47	18.03
NO_2	3-day	PAR (C=1, 2nd gen)	-0.00	11.98	22.80	17.01
NO_2	3-day	PAR (Wind + traffic)	0.16	14.85	24.54	19.25
NO_2	3-day	SVR (Batch)	0.08	15.22	21.82	17.55
NO_2	7-day	PAR (C=1, 1st gen)	-0.02	14.78	23.05	18.29
NO_2	7-day	PAR (C=1, 2nd gen)	-0.09	15.53	23.72	18.90
NO_2	7-day	PAR (Wind + traffic)	-0.18	20.41	24.70	20.56
NO_2	7-day	SVR (Batch)	0.03	15.26	22.40	17.92
PM_{10}	24-hour	PAR (C=1, 1st gen)	0.42	5.27	10.33	7.33
PM_{10}	24-hour	PAR(C=0.5) (2nd gen)	0.33	5.27	11.09	7.71
PM_{10}	24-hour	NN_medium (Batch)	0.26	6.95	11.70	8.77
PM_{10}	3-day	PAR (C=1, 1st gen)	-0.23	6.99	15.06	10.26
PM_{10}	3-day	PAR(C=0.5)(2nd gen)	-0.17	5.99	14.69	9.77
PM_{10}	3-day	NN_medium (Batch)	0.01	7.45	13.51	9.91
PM_{10}	7-day	PAR (C=1, 1st gen)	-0.48	7.77	16.54	11.84
PM_{10}	7-day	PAR(C=0.5)(2nd gen)	-0.40	7.49	16.10	11.33
PM_{10}	7-day	NN_medium (Batch)	-0.31	7.53	15.55	11.04

Table 9.1: Performance metrics for the best performing NO_2 and PM_{10} models

All the results are based on data gathered in April 2019. Both pollutants have models representing the first generation, second generation, and models trained using batch learning algorithms. Performance metrics are included for 24-hours, 3-day, and 7-days forecasts for each of the models. Unlike PM_{10}, NO_2 has two models from the second generation with the inclusion of a model trained using only wind speed and traffic data. By only considering these metrics the models are having similar performances overall. Some of the models perform better for specific length, i.e., The PAR trained using only wind and traffic data performs better than a PAR trained using all variables when comparing 24-hour forecasts, but the latter delivers better results for 7-day forecasts. Least variance between the different forecast lengths is present for forecasts made by the batch learning model, which is likely because it is only affected by changes in weather forecasts.

9.1 Analysis

By only considering performance metrics there is not much separating the different models when comparing the first generation, second generation, and models trained using batch learning. Furthermore, the three online learning algorithms (PAR, SGD, and NN) achieve comparable result for NO_2 and PM_{10} forecasting and the differences between the models are similar for 24-hour, 3-day, and 7-day forecasting. The models for both pollutants seem to be achieving similar performance levels considering the fact that NO_2 levels are usually a lot higher than PM_{10}.

Skewed errors

A general tendency for the models is that the mean absolute error is higher than the median. Figure 9.1 shows the absolute errors for the second generational PAR forecasting NO_2 in April 2019 illustrating where much of the gap between the mean and median is originating. The model keeps an overall decent error rate for most of the *intervals*, but a sudden increase is seen for a group of about 15-20 *intervals* to the right.

This error distribution reflects the strengths and weaknesses of the models as well as the problem in hand. It turns out that bad air quality often is a result of abrupt changes and such events do not occur very often. In other words *intervals* with bad air quality are rather outnumbered by the ones with good air quality, and *intervals* with bad air quality are preceded by several *intervals* with good air quality. This is observed for both NO_2 and PM_{10} levels as seen in Figure 9.2, but NO_2 does seem to be somewhat more volatile.

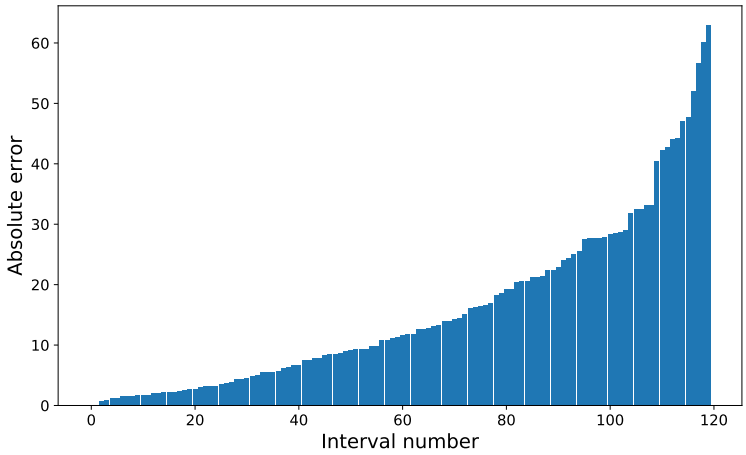
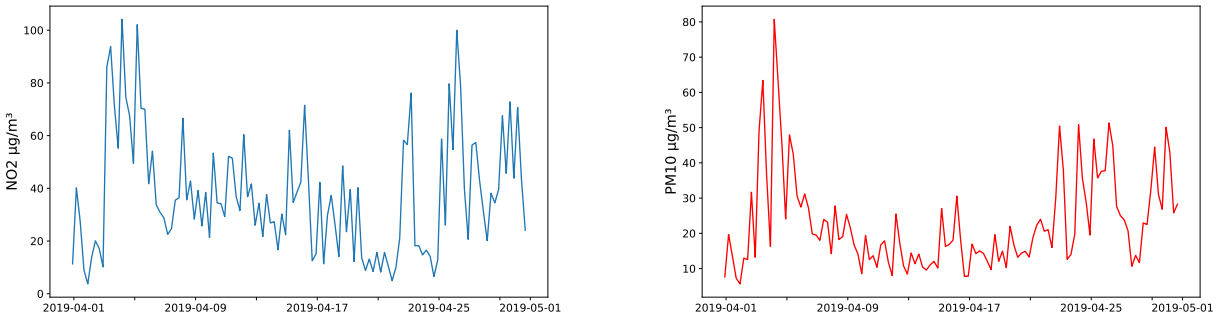


Figure 9.1: Absolute errors made by PAR (second gen.) forecasting NO_2 - April 2019

The figure shows the absolute errors made by a second generational Passive-Aggressive Regressor in 24-hour forecasts made April 2019. The absolute errors are, as mentioned before, the absolute distance between the forecasted values and the observed values. The order of the errors in the graph is set by the size of the errors and is in no way related to the dates of the errors. As illustrated the errors do have a stable increase for about 100 **intervals** and the final remaining 20 errors are seeing a sudden increase. This results in a higher mean absolute error than median absolute error for the model, which is seen in all the air quality models in the program. This is mainly attributed to the fact that few models are capable of forecasting sudden increases in pollutant levels, resulting in large errors when they do occur.



(a) NO_2 (b) PM_{10}

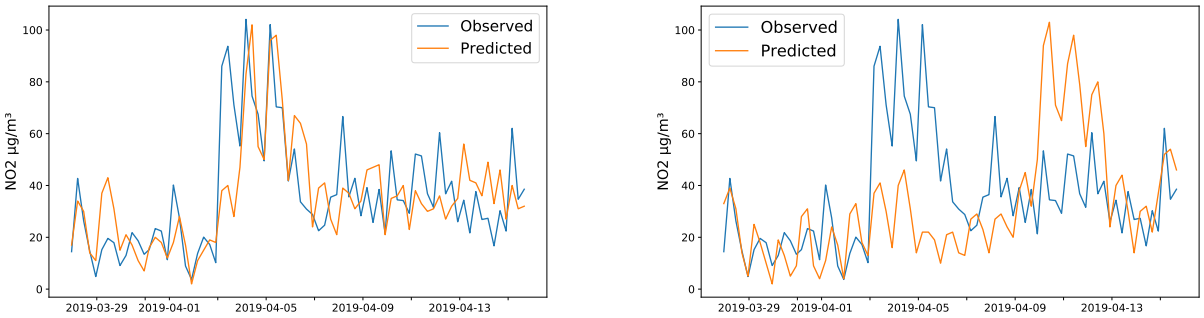
Figure 9.2: Observed pollutant levels - April 2019

Observed pollutant levels for April 2019 recorded at Danmarkplass, Bergen. The original data are in hourly intervals, but this graph is based on the data in six-hour *intervals* using mean values of the original data. NO_2 are somewhat more volatile than PM_{10}. Sudden raises in pollutant levels are however seen for both NO_2 and PM_{10}.

With the sudden worsening of air quality in mind the models can broadly be categorized into two behaviours characterized by their learning rates and the influence they have on predictions over time. To illustrate how the models handle periods with high pollutant levels, the sudden raise in NO_2 in the beginning of April is used as an example by looking at data gathered between 28 March 2019 and 15 April 2019 in the following sections.

High learning rate

The models with a high learning rate tend to respond rapidly when changes occur such as the sudden rise of pollutant levels. However, if these changes are preceded by several *intervals* with low pollutant rates the models have forgotten what bad air quality looks like and are unable to catch it before it is too late.



(a) 24-hour forecast

(b) 7-day forecast

Figure 9.3: Neutral network with high learning rate (NO_2)

Models with a high learning rate tend to fit well to the observed values overall but struggle when facing sudden raises in pollutant levels. The models remain one step behind the actual levels when such events occur, as seen in the 24-hour forecasts around 2 April. Perhaps more important, the 7-day forecasts are based in the high polluted periods, resulting in too high forecasted values.

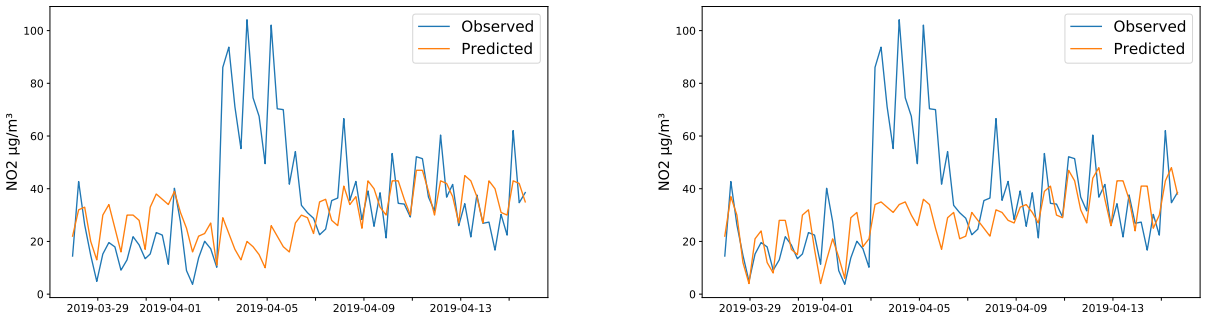
However, the model detects the sudden increase in NO_2 levels after it occurs and updates its weights with the high learning rate. The improvement is seen immediately for the 24-hour forecast which lines up nicely with the observational data. Even though this is good news, many models do seem to be one step behind the observational data as seen in Figure 9.3.

A potentially larger issue is the effect this has on the following days. When looking at the 7-day forecasts, it seems like the models almost ignore the data and rather focus on the high pollutant levels on the days the forecasts are made resulting in too high forecasted values.

Low/medium learning rate

The other behaviour is seen in models using a low to medium learning rate, resulting in less impressionable behaviour when exposed to sudden raises in pollutant levels. This has led to more consistent predictions when comparing 24-hours, 3-days, and 7-days

forecasts. These models generally perform well but the lowered learning rate has a natural downside. Since the *intervals* with high NO_2 and PM_{10} levels are outnumbered by the ones with low levels, the models have fewer training examples with bad air quality to learn from. This, combined with the low learning rate, results in models struggling to learn to forecast periods with high pollution levels, and the models end up ignoring the high polluted events altogether.



(a) 24-hour forecast

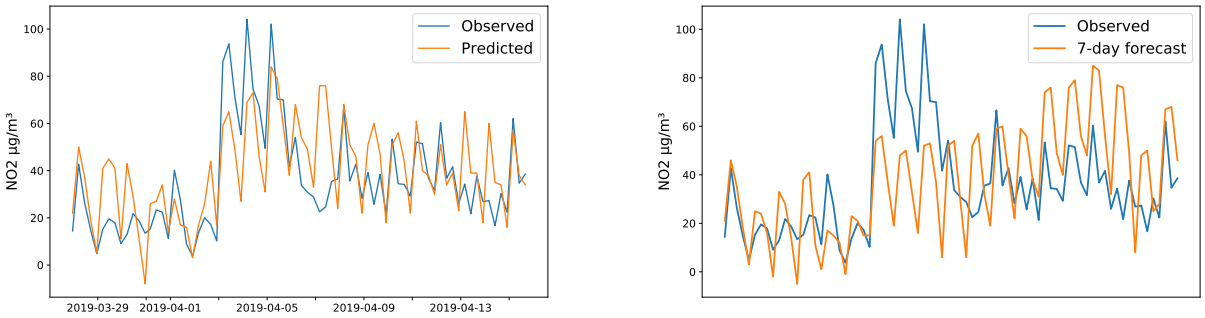
(b) 7-day forecast

Figure 9.4: SGD with low learning rate (NO_2)

Models with low learning rates are performing well for both 24-hour and 7-day forecasts for most days. However, sudden raises in pollutant levels are almost ignored since such periods are heavily outnumbered by days with low pollutant levels. The low learning rates are preventing the models to learn anything from the few high polluted days before the levels drop to normal.

Wind speed & traffic levels for NO_2

The two behaviours explained so far cover almost all the models for both NO_2 and PM_{10} forecasting regardless of learning algorithm and feature sets. There is however one exception, the Passive-Aggressive Regressor trained using only wind speed and traffic levels with a relatively low learning rate.



(a) 24-hour forecast

(b) 7-day forecast

Figure 9.5: PAR(C=0.5) using wind m/s & traffic levels (NO_2)

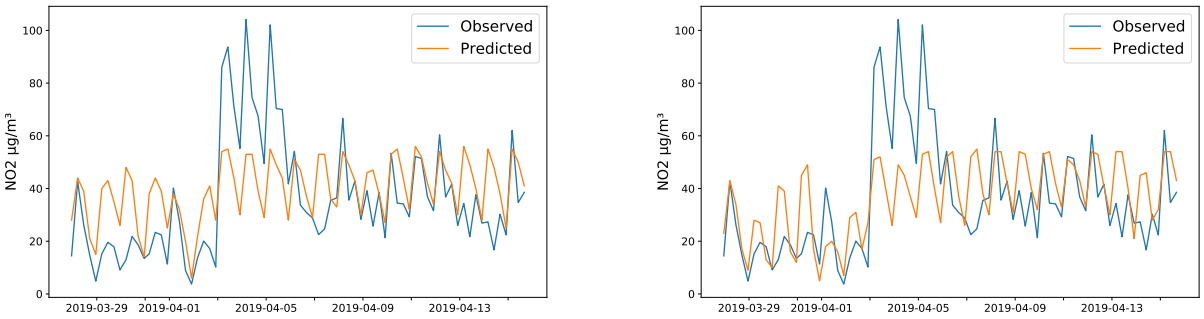
NO_2 models trained using only wind speed and traffic levels provide a unique behaviour compared to the other models being able to foresee the sudden raise in NO_2 levels for 24-hour forecasts. Some of the drawbacks from the other models are however also present here, considering the 24-hour forecasts are still too low compared to the observed values, and the 7-day forecasts are predicting too high values based on the sudden increase that occurred when the forecasts were made.

Compared to the other models it is capable of providing a somewhat unique behaviour by foreseeing the raise in NO_2 levels and still provides reasonable forecasts for the surrounding days. It should therefore be considered one of the more successful models in the program. Two drawbacks of the model in its current state are: (a) The forecasted value during the peak of the pollutant levels should ideally be even higher, and (b) The model is having some of the same issues as the models with higher learning rates when making forecasts for the following days. After being exposed to high pollutant levels overshooting occurs by some margin. Unfortunately, no models were able to capture the same behaviour for PM_{10} levels.

Batch learning

The final models included in the program are the ones trained using batch learning for comparison. Figure 9.6 shows the forecasts made by a Support Vector Regressor, but

all the models trained using batch learning show the same tendencies. The models are naturally more consistent than their online learning counterparts in terms of changes per day since they are only affected by changes in the weather forecasts. However, none of these models are able to foresee the raise of the pollutant levels and keep a stable output of forecasted values relatively consistent with days with low pollutant levels.



(a) 24-hour forecast

(b) 7-day forecast

Figure 9.6: SVR trained using batch-learning

Models trained using batch learning delivers consistent forecasts when comparing the lengths of the different forecasts (24-hours, 3-days, and 7-days). These models are, unlike the rest, only affected by changes in the weather forecasts. The behaviour of these models are similar with the incremental models trained using a low learning rate: Performing well for most days, but are unable to foresee sudden raises in pollutant levels. This is much likely because periods with high pollutant levels are underrepresented in the data set.

Chapter 10

Discussion

Variables used for forecasting

By looking at which variables affect the pollutant levels there are few surprises compared to previous findings in similar work. Wind speed and the amount of traffic are the variables with the best prediction capabilities for NO_2. PM_{10} on the other hand seems to be a result of a much more complex relationships of variables, making it harder to make good forecasts.

The intent behind the program is to capture changes in pollutant levels over time. Domestic heating for instance is difficult to measure compared to many other variables. The same goes for the ship traffic caused by tourism which mainly is a concern during the summer months [2]. A negative aspect of the incremental approach is that the program is not able to bring any insights into how much these implicit variables affect the pollutant levels.

Traffic assessment

The traffic assessment method used in Braluft is focusing on feasibility through simplicity by only accounting for the traffic volume, but the trade-off for using this approach is the loss of details in the form of traffic speed and vehicle type distribution, two important factors for modeling future air quality [2]. Assessing traffic through real-time

video with YOLO or a different detection method could solve this issue and potentially improve the air quality forecasting models of Braluft.

However, the relatively simple solution used in this thesis has been able to provide traffic observations with good representational capabilities well suited for the role of input data. The observed traffic data show well defined patterns in terms of workweek, weekends, and holidays.

The performance of the traffic forecasting models could also be improved by exploring new variables. The introduction of time differentiated road toll has, for instance, resulted in a 15-16% traffic reduction during hours in the morning and afternoon [2] compared to the levels before it was introduced. This might be included to some degree by the incremental nature of the program, but it is deserving a further exploration to see how it affects air quality through traffic levels.

Using incremental learning

By using the incremental approach the program is defining itself as reactive, instead of proactive. The models in the program are more specifically characterized by trying to make sense of the recent observations and the errors made in order to improve its future forecasts, and unlike models trained using batch-learning older historic observations are potentially forgotten over time.

The approach results in a program that performs well for most days where the pollutant levels are relatively low and more importantly, stable. These results are loosely connected to previous studies concluding that the best predictive variables for NO_X and PM_{10} in Athens and Helsinki were the air pollutant concentrations from the previous day [18]. This is essentially how the 24-hour forecasts from models with a high learning rate are made. The weights of these models are altered so much for the recent observations that asking for forecasts are essentially the same as asking "How was the air quality yesterday?".

A key challenge for the program is the fact the periods with bad air quality tend to arise abruptly and can therefore hardly be considered a trend. This results in many

models forgetting what bad air quality looks like or they are unable to learn it. The best performing models to counter these effects are simple NO_2 models trained using only wind speed and traffic data. No such model was found for PM_{10}, but the best predictor is probably NO_2 based on a Pearson coefficient of 0.5 and the patterns seen in Figure 10.1. A good NO_2 forecast has in other words potential to describe upcoming PM_{10} levels.

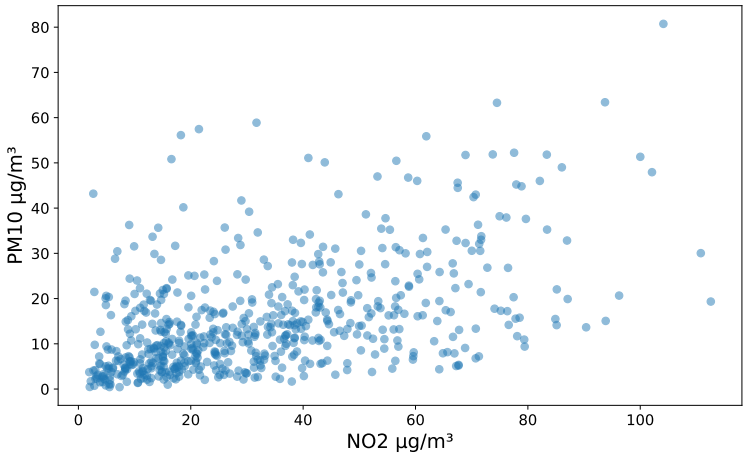


Figure 10.1: NO_2 - PM_{10} visualized

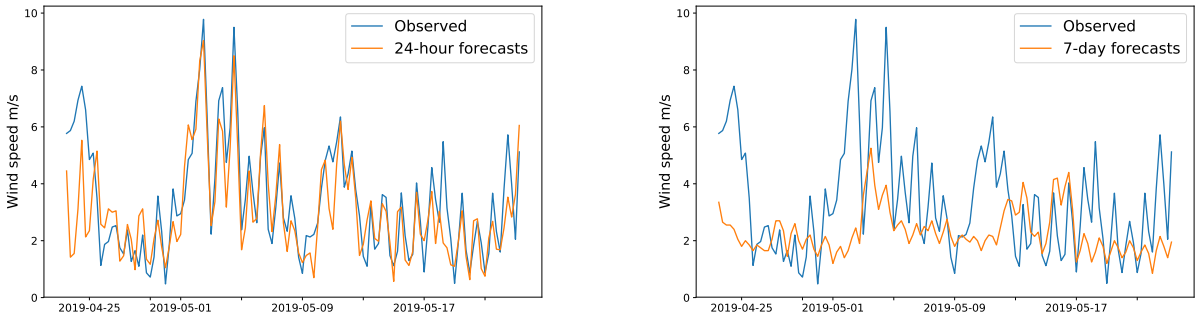
An important consideration related to these remarks and overall review of the program is the relatively small size of the data set used. This is naturally affecting both the review of the variables and the performance of the machine learning models. In addition to this all the data used in this paper are gathered during winter or spring, meaning performance may change considerable as time and seasons passes. It is also likely that the program would see an increased performance if tested at a location where raises in pollutant levels are less sudden.

Weather forecasts accuracy

An essential contributor to the air quality forecasts are naturally the weather forecasts. More important, accurate air quality forecasts require accurate weather forecasts.

It is difficult to tell exactly how tight the relationship between weather forecasts and air quality forecasts made by incremental models is. That being said, any reduction of accuracy in the weather forecasts will certainly lead to a reduction in accuracy for these models if the incremental nature is not accounted for. This effect is clearly observed for models trained using batch learning that are only affected by changes in the weather forecasts. A small and stable decrease in performance is seen for these models as the length of the forecasts increase. However, this decrease in performance is smaller than the decrease seen in most incremental models. This is most likely caused by two factors: (a) incremental models perform generally better than batch learning models for short term forecasts (24-hour) and worse for long forecasts (7-day), resulting in a broader range of errors, and (b) the incremental nature of the models acts as a potentially extra error source that most likely is a bigger contributor to the size of the errors than weather forecasts as the length of the forecasts increase.

The largest concern when considering the accuracy of weather forecasts is the wind speed. Wind speed has proven to be a very important variable for forecasting pollutant levels, especially for NO_2. As illustrated by Figure 10.2 short term forecasts are very accurate. However, the 7-day forecasts are struggling to foresee peaks in wind speed and remain rather conservative for most days. This behaviour is very similar to what is seen in many of the incremental air quality models in the Braluft program. Unfortunately the lack of accuracy in wind speed forecasts is not enough to explain why many air quality models are having difficulties foreseeing sudden raises in pollutant levels. The reason behind this is the fact that most air quality models are not able to forecast sudden increases for short term either, even though the weather forecasts are rather accurate. However, this raises the question of whether it is even possible to forecast the pollutant levels 7-days ahead using meteorological variables.



(a) 24-hour forecast

(b) 7-day forecast

Figure 10.2: Wind speed forecasts

Wind speed forecasts tend to be very accurate short-term, but the forecasts get more conservative as the length of the forecasts increases.

Challenges in development

The challenges related to the development of the program are mostly concerned with engineering of the machine learning pipeline or modelling of the machine learning models. Additionally, the incremental nature of the program increases the complexity of the program and especially the engineering part.

Engineering challenges

Considering engineering the biggest challenge is *time* for several reasons. First of all, the instructions in the program have a natural order. The web camera images must be processed before the traffic models can be trained, weather forecasts must be collected before the creation of air quality forecasts, and the list goes on. Second, some of the data required for the program are only available for a brief period of time. Third, much of the data provided by external sources are using different time intervals ranging from 1 to 6 hours.

The Microservice architecture proved to be a great way of dealing with the time re-

lated challenges. The Main Service was implemented to handle communication between the services and making sure actions were performed in the correct order. The other two time related challenges were also mitigated by using separate services capable of downloading and formatting the data without affecting the rest of the program. This separation of concerns also allowed easy redeployments upon changes in functionality.

Modelling challenges

In terms of modelling the largest challenge in the program is the absurd amount of possibilities when considering different algorithms, feature sets, and learning parameters. This might initially seem like a positive thing as many different sets of possible combinations exist offering opportunities, but the challenge is to find the *best performing* combination. The challenge of having too many options for a machine learning problem is not specific for this program. One might even argue that the process of selecting the best combinations for traffic, NO_2, and PM_{10} is assisted by the nature of this program.

First of all, the amount of already implemented learning algorithms is drastically reduced by the requirement of being able to update models incrementally. Furthermore, the intent behind the program is to test a concept, not to create production ready models. The linear models and the neural networks selected were deemed sufficient for this cause by representing both simple and more complex learning algorithms, even though different frameworks could perhaps provide marginally better performance.

Second, for the feature sets there are a limited number of relevant variables available for this problem. More importantly, the variables used are properly labeled and their relationships with pollutant levels have been explored previously, although local variations exist. However, even with such insights it is difficult to create an ideal feature set as proven with PM_{10} in this thesis. Experimentation is therefore required to find the best features for the problem.

Finally, the learning parameters were the most difficult part of the modelling process in the program. The learning parameters may affect both performance metrics and the behaviour of the models as illustrated by how models with high and low learning rates

act differently when faced with major changes in pollutant levels. These parameters were during the modelling adjusted by creating several different models and comparing the performance metrics and their behaviour. This is primary performed without any assistance from external resources. That being said, the learning algorithms did prove to have sensible default values that enabled decent performance in models from the very beginning.

The need of experimentation to find the ideal combination resulted in a total of 132 models when considering every one of the problems in the program (traffic, NO_2, and PM_{10}), all using different combinations of learning algorithms, feature sets, and learning parameters. The actual number is somewhat higher as several models were created to test different parts of the the machine learning pipeline. Most of the models were created and trained as a part of a small group of models, and almost all the groups revealed something new in the form of unseen performance metrics and behaviours. It was not until the number of models reached 100-110 the observation of new experiences started to stagnate. It was at this point the modelling phase ended. This goes to show how much work is required for a machine learning process, even when the options are narrow.

Chapter 11

Conclusion

This thesis has presented Braluft, a software stack created to gain insight into the air quality by examining data related to Bergen, Norway. The individual components of the stack are running in a microservice architecture and are brought together in order to make up a program capable of gathering data, incrementally train machine learning models, and create forecasts for the air pollutants NO_2 and PM_{10}.

Impact of variables

To review how meteorological variables and traffic levels affect NO_2 and PM_{10} levels a data set was constructed using observations made by the program between December 2018 and April 2019. Based on the data set the main influences of NO_2 levels were wind speed and traffic levels by a wide margin. This correlation is supported by the fact that the best performing model overall was responsible for forecasting NO_2 using only wind speed and traffic levels.

For PM_{10} the effect of each variable was more spread out, meaning none of the variables stands out as better predictors than the others. The PM_{10} levels do in other words seem to be a result of a larger system of processes. The variable with best predictive capabilities for PM_{10} is probably NO_2.

The effect of the variables is comparable with many findings from similar work,

such as reductions in pollutant levels as wind speed increases or vehicular traffic levels decreases.

In terms for wind direction similar effects are seen for both NO_2 and PM_{10}. Wind is mainly headed south/south-east or north/north-west due to the surroundings with elevated areas. Wind headed north tend to worsen the air quality, suggesting that pollutants are brought in from other areas. The opposite is seen when the wind is headed south. However, these effects are rather small for both the observed pollutants in the program.

Online learning for air quality forecasting

The use of incremental models is showing a lot of promise in terms of forecasting air quality using meteorological variables and traffic levels. The approach allows the program to capture the recent trends and changes in seasons and deliver forecasts with overall good precision for both NO_2 and PM_{10}.

However, even though the models are generally performing well, there is one major challenge associated with the task when considering Danmarkplass, Bergen. The program is built on the idea of adapting to changes over time, but periods with bad air quality have during the development of the program been a result of abrupt changes and are often not lasting more than a couple of days. Such occurrences can barely be considered a trend. This results in many models forgetting what bad air quality looks like and are therefore not able to forecast it. This is especially true for the models with a high learning rate that miss the initial raise in pollutant levels and remain one step behind the actual levels. The models with a low learning rate are on the other hand missing the raise altogether.

Online/batch learning

When comparing *online learning models* with *batch learning models* there are not much separating the models considering performance metrics. That being said, the strategies

for these models are somewhat different. As mentioned before most of the online learning models tend to perform well for most days where the pollutant levels are low, but make large errors upon sudden worsening of the air quality. The batch learning models, on the other hand, are making most forecasts targeting a middle ground between *ordinary* days with little pollution and highly polluted days. These models are therefore often closer to forecasting periods with bad air quality, but make more mistakes for *ordinary* days.

Finally, the data set being used for this paper is still fairly small for both online- and batch learning models with around 500 training samples. An increase in performance should be expected as time passes and the program collects more data, especially for the more complex models.

Bibliography

[1] S. Gregor and A. R. Hevner, "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly*, vol. 37, pp. 337–355, 2 2013.

[2] B. K. Ann Høiskar, I. Sundvor, M. Johnsrud, T. W. Haug, and o. Hilde Solli, "Tiltaksutredning for lokal luftkvalitet i Bergen," tech. rep., 2017.

[3] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, 2012.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.*

[6] O. Y. Al-Jarrah, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient Machine Learning for Big Data: A Review," *Big Data Research*, vol. 2, pp. 87–93, 9 2015.

[7] G. A. Susto, A. Schirru, S. Pampuri, and S. McLoone, "Supervised Aggregative Feature Extraction for Big Data Time Series Regression," *IEEE Transactions on Industrial Informatics*, vol. 12, pp. 1243–1252, 6 2016.

[8] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, "The role of big data in smart city," *International Journal of Information Management*, vol. 36, pp. 748–758, 10 2016.

[9] E. Curry, S. Hasan, C. Kouroupetroglou, W. Fabritius, U. ul Hassan, and W. Derguech, "Internet of Things Enhanced User Experience for Smart Water and Energy Management," *IEEE Internet Computing*, vol. 22, pp. 18–28, 1 2018.

[10] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, pp. 22–32, 2 2014.

[11] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, 10 2010.

[12] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine Learning With Big Data: Challenges and Approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[13] Y.-t. Zhuang, F. Wu, C. Chen, and Y.-h. Pan, "Challenges and opportunities: from big data to knowledge in AI 2.0," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, pp. 3–14, 1 2017.

[14] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Recent advances and emerging challenges of feature selection in the context of big data," *Knowledge-Based Systems*, vol. 86, pp. 33–45, 9 2015.

[15] C. Rudin and K. L. Wagstaff, "Machine learning for science and society," *Machine Learning*, vol. 95, pp. 1–9, 4 2014.

[16] A. Donnelly, B. Misstear, and B. Broderick, "Real time air quality forecasting using integrated parametric and non-parametric regression techniques," *Atmospheric Environment*, vol. 103, pp. 53–65, 2 2015.

[17] D. M. Stieb, S. Judek, and R. T. Burnett, "Meta-analysis of time-series studies of air pollution and mortality: effects of gases and particles and the influence of cause of death, age, and season.," *undefined*, 2002.

[18] A. Vlachogianni, P. Kassomenos, A. Karppinen, S. Karakitsios, and J. Kukkonen, "Evaluation of a multiple regression model for the forecasting of the concentrations

of NOx and PM10 in Athens and Helsinki," *Science of The Total Environment*, vol. 409, pp. 1559–1571, 3 2011.

[19] J. HOOYBERGHS, C. MENSINK, G. DUMONT, F. FIERENS, and O. BRASSEUR, "A neural network forecast for daily average PM concentrations in Belgium," *Atmospheric Environment*, vol. 39, pp. 3279–3289, 6 2005.

[20] U. Brunelli, V. Piazza, L. Pignato, F. Sorbello, and S. Vitabile, "Two-days ahead prediction of daily maximum concentrations of SO2, O3, PM10, NO2, CO in the urban area of Palermo, Italy," *Atmospheric Environment*, vol. 41, pp. 2967–2995, 5 2007.

[21] "Luftkvalitet i Norge." `https://luftkvalitet.miljostatus.no/`. Accessed: 2019-03-20.

[22] "Luftkvalitet." `https://www.bergen.kommune.no/hvaskjer/tema/luftkvalitet`. Accessed: 2019-03-20.

[23] "Airqualityforecast." `https://api.met.no/weatherapi/airqualityforecast/0.1/documentation`. Accessed: 2019-03-20.

[24] N. Pérez, J. Pey, M. Cusack, C. Reche, X. Querol, A. Alastuey, and M. Viana, "Variability of Particle Number, Black Carbon, and PM ¡sub¿10¡/sub¿ , PM ¡sub¿2.5¡/sub¿ , and PM ¡sub¿1¡/sub¿ Levels and Speciation: Influence of Road Traffic Emissions on Urban Air Quality," *Aerosol Science and Technology*, vol. 44, pp. 487–499, 6 2010.

[25] H. K. Elminir, "Dependence of urban air pollutants on meteorology," *Science of The Total Environment*, vol. 350, pp. 225–237, 11 2005.

[26] A. R. Hevner, S. T. March, J. Park, and S. Ram, "DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH," *Design Science in IS Research MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[27] S. T. March and G. F. Smith, "Design and natural science research on information technology," *Decision Support Systems*, vol. 15, pp. 251–266, 12 1995.

[28] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 5 2017.

[29] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, p. 67, 12 2016.

[30] K. Crammer, O. Dekel, and J. Keshet, "Online Passive-Aggressive Algorithms," tech. rep., 2006.

[31] "About NILU." `https://www.nilu.no/en/about-nilu/`. Accessed: 2019-03-18.

[32] "About the Norwegian Meteorological Institute ." `https://www.met.no/en/About-us/About-MET-Norway`. Accessed: 2019-03-21.

[33] P. Chakraborty, Y. O. Adu-Gyamfi, S. Poddar, V. Ahsani, A. Sharma, and S. Sarkar, "Traffic Congestion Detection from Camera Images using Deep Convolution Neural Networks," *Transportation Research Record: Journal of the Transportation Research Board*, p. 036119811877763, 6 2018.

[34] N. Buch, S. A. Velastin, and J. Orwell, "A Review of Computer Vision Techniques for the Analysis of Urban Traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 920–939, 9 2011.

[35] H. Zhang, Y. Wang, J. Hu, Q. Ying, and X.-M. Hu, "Relationships between meteorological parameters and criteria air pollutants in three megacities in China," *Environmental Research*, vol. 140, pp. 242–254, 7 2015.

[36] A. Thorpe and R. M. Harrison, "Sources and properties of non-exhaust particulate matter from road traffic: A review," *Science of The Total Environment*, vol. 400, pp. 270–282, 8 2008.

[37] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," 2015.

[38] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, pp. 16–28, 1 2014.

[39] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, pp. 245–271, 12 1997.

[40] "Model evaluation: quantifying the quality of predictions." `https://scikit-learn.org/stable/modules/model_evaluation.html`. Accessed: 2019-04-12.

[41] G. Corani, "Air quality prediction in Milan: feed-forward neural networks, pruned neural networks and lazy learning," *Ecological Modelling*, vol. 185, pp. 513–529, 7 2005.

[42] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software*, vol. 33, pp. 42–52, 5 2016.

[43] M. Fazio, A. Celesti, R. Ranjan, C. Liu, L. Chen, and M. Villari, "Open Issues in Scheduling Microservices in the Cloud," *IEEE Cloud Computing*, vol. 3, pp. 81–88, 9 2016.

[44] "UH-IaaS." `https://www.uninett.no/uh-iaas`. Accessed: 2019-03-21.

[45] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, IEEE, 6 2016.

[46] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 4 2018.

[47] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1370–1377, IEEE, 5 2017.

[48] D. Heo, E. Lee, and B. C. Ko, "Pedestrian Detection at Night Using Deep Neural Networks and Saliency Maps," *Electronic Imaging*, vol. 2018, pp. 060403–1, 1 2018.

[49] B. T. Nugraha, S.-F. Su, and Fahmizal, "Towards self-driving car using convolutional neural network and road lane detector," in *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, pp. 65–69, IEEE, 10 2017.

[50] F. Pedregosa, N. Alexandre Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, P. Prettenhofer, J. Vanderplas, M. Brucher, M. Perrot an Edouard Duchesnay, A. Matthieu Brucher, M. Perrot, and C. F. Edouard Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A

Traffic model metrics

Passive-Aggressive Regressors

loss	C	epsilon	r2	median_abs_err	rmse	mean_abs_err
epsilon_insensitive	1	20	-0.26	152	382.65	280.56
epsilon_insensitive	10	20	0.66	94	197.94	143.56
epsilon_insensitive	15	30	0.71	87	184.01	134.13
epsilon_insensitive	20	20	0.73	87	178.01	128.58
epsilon_insensitive	25	20	0.73	94	177.36	131.17
epsilon_insensitive	30	0.1	0.72	98	181.74	133.63
epsilon_insensitive	30	30	0.72	98	179.11	133.22
epsilon_insensitive	50	50	0.71	103	182.3	137.92
squared_epsilon_insensitive	1	20	0.6	118	216.74	160.01
squared_epsilon_insensitive	10	20	0.52	120	237.39	175.54
squared_epsilon_insensitive	15	30	0.53	118	232.77	171.89
squared_epsilon_insensitive	25	20	0.51	121	239.16	176.99
squared_epsilon_insensitive	30	0.1	0.45	132	252.09	187.96
squared_epsilon_insensitive	50	50	0.57	121	223.95	165.62

Stochastic Gradient Descent

Learning rate: constant

loss	eta0	r2	median_abs_err	rmse	mean_abs_err
squared_epsilon_insensitive	0.1	0.73	111	176.73	136.14
squared_epsilon_insensitive	0.2	0.6	115	216.3	157.89
squared_epsilon_insensitive	0.3	-0.42	240	406.5	313.81
squared_epsilon_insensitive	0.05	0.8	115	171.18	142.32
squared_epsilon_insensitive	0.15	0.69	107	191.32	142.42

squared_epsilon_insensitive	0.01	0.58	152	220.5	175.72
huber	4	0.54	144	231.38	179.78
huber	0.3	0.5	101	242.03	173.69
huber	5	0.42	166	259.47	200.14
huber	1	0.7	93	185.89	133.81
huber	0.5	0.64	95	205.19	146.29
huber	1.5	0.72	96	181.11	133.26
epsilon_insensitive	40	0.69	93	190.1	138.62
epsilon_insensitive	20	0.69	89	190.49	137.56
epsilon_insensitive	35	0.69	94	189.34	137.22
epsilon_insensitive	30	0.69	88	188.46	134.93
epsilon_insensitive	25	0.7	92	188.24	134.93

Learning rate: optimal

loss	alpha	r2	median_abs_err	rmse	mean_abs_err
squared_epsilon_insensitive	1.2	0.3	220	285.45	237.03
squared_epsilon_insensitive	2	0.19	241	306.33	254.81
squared_epsilon_insensitive	0.5	0.48	181	245.09	201.94
squared_epsilon_insensitive	1	0.34	213	277.12	230.4

Neural Networks

loss	optimizer	learning rate	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MAE	RMSprop	0.01	1	12-64-64	-248.82	352	5391.92	2458.48
MAE	Adam	1	1	12r-4r	0.35	133	274.05	199.65
MAE	RMSprop	0.05	1	12-64	0.23	170	299.73	225.95
MAE	Adam	0.01	1	12-64-64	0.32	160	281.18	211.55
MAE	Adam	0.01	1	12-8r	-0.43	203	407.48	306.75
MAE	Adam	0.001	1	12-64r-64r	-1	249	482.24	365.88
MAE	Adam	0.001	1	12-8r	-1.45	299	533.74	411.1
MSE	Adam	0.01	1	64-64r-64r	0.57	101	224.87	157.96
MSE	RMSprop	0.01	1	64-64r-64r	0.2	169	305.93	232.07
MSE	Adam	0.01	1	32-16r-8r	-0.29	205	388.9	296.01
MSE	Adam	0.01	1	128r-32-r-8r	-0.04	181	348.53	264.34
MSE	Adam	0.01	10	128-64r-32r-16r	0.51	97	238.76	166.51
MAE	Adam	0.01	10	128-64r-32r-16r	0.52	93	237.68	163.59
MSE	Adam	0.01	100	128-64r-32r-16r	0.57	93	225.66	158.74
MSE	Adam	0.001	20	6	0.64	97	206.8	147.78
MSE	Adam	0.01	20	6r-6r	0.65	91	201.61	142.13
MSE	Adam	0.01	50	6r-18r-12r-6r	0.6	98	215.92	154.78
MSE	Adam	0.01	20	128-64r-32-16r	0.54	96	260.97	178.6
MSE	Adam	0.01	20	6r-6r	0.64	92	226.74	159.55
MSE	Adam	0.01	20	6r-18r-12r-6r	0.8	115	171.18	142.32

Appendix B

First generation NO2

Passive-Aggressive Regressors

24-hour forecast / 05.03.2019 - 05.04.2019

C	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
0.01	1	-1.55	21.24	36.95	28.78
0.1	1	0.02	10.07	22.93	15.69
1	1	0.4	10.27	17.93	13.59
10	1	0.2	11.51	20.67	15.63
100	1	0.2	11.33	20.69	15.63
0.01	2	-1.53	20.41	36.78	28.62
0.1	2	0.11	10.38	21.87	15.49
1	2	0.39	10.08	18.1	13.59
10	2	0.34	12.43	18.84	14.35
0.01	3	-1.49	19.07	36.5	28.12
0.1	3	-0.03	11.78	23.51	17.04
1	3	0.34	11.22	18.82	14.26
10	3	0.33	13.27	18.92	14.69

3-day forecast / 05.03.2019 - 05.04.2019

C	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
0.01	1	-1.56	21.67	36.99	28.95
0.1	1	-0.01	9.93	23.23	16.12
1	1	0.13	13.08	21.59	16.72
10	1	-0.2	18.1	25.33	19.89
100	1	-0.19	17.86	25.26	19.8
0.01	2	-1.49	21.6	7 36.54	28.63
0.1	2	-0.02	12.18	23.35	16.97
1	2	-0.02	13.84	23.33	17.7
10	2	-0.1	15.78	24.25	19.17
0.01	3	-1.39	19.67	35.74	27.82
0.1	3	-0.27	13.69	26.09	19.98
1	3	-0.17	15.56	25.06	19.49

10	3	-0.12	15.15	24.53	19.19

7-day forecast / 05.03.2019 - 05.04.2019

C	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
0.01	1	-1.58	21.51	37.13	29.14
0.1	1	-0.02	10.73	23.41	16.34
1	1	0.14	13.23	21.46	16.76
10	1	-0.16	16.78	24.92	19.75
100	1	-0.15	15.47	24.81	19.62
0.01	2	-1.52	21.53	36.71	28.93
0.1	2	-0.05	10.91	23.69	17.6
1	2	0	14.91	23.14	17.91
10	2	-0.15	15.67	24.82	19.36
0.01	3	-1.4	21.41	35.81	28.06
0.1	3	-0.43	15.45	27.65	21.71
1	3	-0.24	16.24	25.8	20.34
10	3	-0.23	17.58	25.65	20.28

Stochastic Gradient Decent

24-hour forecast / 05.03.2019 - 05.04.2019

loss	learning rate	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
squared_loss	invscaling	1	-0.02	9.4	23.4	15.9
squared_epsilon_insensitive	invscaling	3	-7.26	21.43	66.49	34.02
squared_epsilon_insensitive	invscaling	2	0.2	10.9	20.64	15.12
squared_epsilon_insensitive	invscaling	1	0.07	13.07	22.31	16.68
squared_loss	invscaling	3	0.07	12.15	22.27	16.22
squared_loss	invscaling	2	0.05	11.59	22.55	15.76
squared_loss	constant(0.001)	1	-0.67	11.5	29.92	21
squared_loss	constant(0.01)	1	0.22	13.27	20.46	16.19
squared_loss	constant(0.1)	1	0.21	11.95	20.54	15.42

3-day forecast / 05.03.2019 - 05.04.2019

loss	learning rate	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
squared_loss	invscaling	1	0.06	10.22	22.43	15.76
squared_epsilon_insensitive	invscaling	3	-4.51	23.7	54.32	35.87
squared_epsilon_insensitive	invscaling	2	0.1	11.84	21.9	16.63
squared_epsilon_insensitive	invscaling	1	0.16	11.42	21.22	16.05
squared_loss	invscaling	3	-0.18	14.25	25.14	19.18
squared_loss	invscaling	2	0.02	11.76	22.95	16.82
squared_loss	constant(0.001)	1	-0.71	13.92	30.22	21.86
squared_loss	constant(0.01)	1	-0.02	13.86	23.35	17.79
squared_loss	constant(0.1)	1	0.17	13.3	21.11	16.39

7-day forecast / 05.03.2019 - 05.04.2019

loss	learning rate	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
squared_loss	invscaling	1	0.07	9.8	22.34	15.86
squared_epsilon_insensitive	invscaling	3	-10.9	18.72	79.81	40.6
squared_epsilon_insensitive	invscaling	2	0.05	13.06	22.49	17.19
squared_epsilon_insensitive	invscaling	1	0.19	11.92	20.84	15.68
squared_loss	invscaling	3	-0.33	15.38	26.65	21
squared_loss	invscaling	2	-0.02	12.99	23.35	17.65
squared_loss	constant(0.001)	1	-0.78	15.3	30.83	23.16
squared_loss	constant(0.01)	1	-0.11	15.1	24.41	18.99
squared_loss	constant(0.1)	1	0.17	13.34	21.05	16.2

Neural Networks

24-hour forecast / 05.03.2019 - 05.04.2019

loss	optimizer	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MSE	Adam(0.01)	20	128-64r-32r-16r	0.1	13.05	20	15.73
MSE	Adam(0.01)	20	6r-6r	0.286	9.83	17.87	13.03
MSE	Adam(0.01)	20	6r-18r-12r-6r	-0.38	10.55	24.77	15.96

3-day forecast / 05.03.2019 - 05.04.2019

loss	optimizer	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MSE	Adam(0.01)	20	128-64r-32r-16r	-0.32	12.93	24.23	17.75
MSE	Adam(0.01)	20	6r-6r	-0.2	10.26	23.1	16.81
MSE	Adam(0.01)	20	6r-18r-12r-6r	-0.55	11.27	26.29	18.83

3-day forecast / 05.03.2019 - 05.04.2019

loss	optimizer	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MSE	Adam(0.01)	20	128-64r-32r-16r	-0.44	10.55	25.34	18.21
MSE	Adam(0.01)	20	6r-6r	-0.34	13.78	24.4	18.34
MSE	Adam(0.01)	20	6r-18r-12r-6r	-0.09	11.95	21.98	16.62

Epsilon=5 - loss = epsilon_insensitive

Appendix C

First generation PM10

Passive-Aggressive Regressors

24-hour forecast / 05.03.2019 - 05.04.2019

C	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
0.01	1	-1.01	11.21	21.48	15.42
0.1	1	0.12	6.37	14.23	9.58
1	1	0.37	6.66	12	8.88
10	1	0.27	6.87	12.93	9.37
100	1	0.27	6.87	12.93	9.37
0.01	2	-1	12.09	21.42	15.73
0.1	2	-0.21	6.98	16.67	11.25
1	2	-0.38	9.55	17.77	12.96
10	2	-0.4	8.78	17.92	12.97
0.01	3	-0.9	10.09	20.88	15.25
0.1	3	-0.5	9.57	18.54	13.58
1	3	-0.51	11.02	18.61	13.96
10	3	-0.51	11.03	18.61	13.99

3-day forecast / 05.03.2019 - 05.04.2019

C	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
0.01	1	-1.05	11.59	21.71	15.75
0.1	1	-0.12	6.93	15.99	10.73
1	1	-0.15	6.98	16.25	10.69
10	1	-0.38	8.28	17.8	11.81
100	1	-0.38	8.28	17.8	11.81
0.01	2	-0.96	10.74	21.19	15.39
0.1	2	-0.22	7.31	16.7	11
1	2	-0.34	8.03	17.51	11.46
10	2	-0.31	7.26	17.33	11.28
0.01	3	-0.89	10.52	20.8	15
0.1	3	-0.47	7.24	18.39	12.35
1	3	-0.55	8.29	18.88	12.44

10	3	-0.54	8.12	18.77	12.39

7-day forecast / 05.03.2019 - 05.04.2019

C	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
0.01	1	-1.07	11.69	21.8	15.93
0.1	1	-0.11	6.9	15.93	10.55
1	1	-0.18	7.59	16.45	11.54
10	1	-0.38	9.84	17.76	13.27
100	1	-0.38	9.84	17.76	13.27
0.01	2	-0.89	11.4	20.85	14.91
0.1	2	0.09	6.45	14.43	9.79
1	2	0.22	6.81	13.37	9.69
10	2	0.3	7.18	12.68	9.4
0.01	3	-0.76	9.69	20.08	14.31
0.1	3	-0.24	7.83	16.83	11.05
1	3	-0.06	8.11	14.67	10.64
10	3	0.09	7.96	14.47	10.55

Stochastic Gradient Decent

24-hour forecast / 05.03.2019 - 05.04.2019

loss	learning rate	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
squared_loss	invscaling	1	-0.08	5.82	15.74	10.03
squared_epsilon_insensitive	invscaling	3	-0.91	10.38	20.91	13.84
squared_epsilon_insensitive	invscaling	2	0.08	6.11	14.5	9.69
squared_epsilon_insensitive	invscaling	1	0.1	6.62	14.39	9.5
squared_loss	invscaling	3	-0.18	6.41	16.44	10.64
squared_loss	invscaling	2	0	5.59	15.13	9.84
squared_loss	constant(0.001)	1	-1.6	19.98	24.42	20.81
squared_loss	constant(0.01)	1	-1.8	18.19	25.32	20.39
squared_loss	constant(0.1)	1	-0.17	7.92	16.37	11.44

3-day forecast / 05.03.2019 - 05.04.2019

loss	learning rate	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
squared_loss	invscaling	1	-0.24	5.72	16.84	10.92
squared_epsilon_insensitive	invscaling	3	-0.95	11.5	21.16	15.49
squared_epsilon_insensitive	invscaling	2	-0.25	6.19	16.93	11.14
squared_epsilon_insensitive	invscaling	1	-0.09	6.31	15.83	10.52
squared_loss	invscaling	3	-0.56	7.76	18.95	12.97
squared_loss	invscaling	2	-0.32	6.62	17.39	11.41
squared_loss	constant(0.001)	1	-1.23	18.18	22.61	18.77
squared_loss	constant(0.01)	1	-1.16	15.88	22.29	17.95
squared_loss	constant(0.1)	1	-0.12	7.05	16.04	11.05

7-day forecast / 05.03.2019 - 05.04.2019

loss	learning rate	poly. deg.	r2	median_abs_err	rmse	mean_abs_err
squared_loss	invscaling	1	-0.24	6.1	16.86	11.03
squared_epsilon_insensitive	invscaling	3	-1.68	10.57	24.78	16.57
squared_epsilon_insensitive	invscaling	2	-0.21	6.82	16.64	11.25
squared_epsilon_insensitive	invscaling	1	-0.07	7.05	15.7	10.21
squared_loss	invscaling	3	-0.54	9.52	18.79	13.68
squared_loss	invscaling	2	-0.32	7.28	17.4	11.89
squared_loss	constant(0.001)	1	-1.31	16.4	23.02	18.69
squared_loss	constant(0.01)	1	-0.96	15.41	21.23	16.89
squared_loss	constant(0.1)	1	-0.18	7.62	16.48	11.39

Neural Networks

24-hour forecast / 05.03.2019 - 05.04.2019

loss	optimizer	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MSE	Adam(0.01)	20	128-64r-32r-16r	0.13	5.5	11.81	8.38
MSE	Adam(0.01)	20	6r-6r	0.21	6.46	11.21	8.39
MSE	Adam(0.01)	20	6r-18r-12r-6r	0.33	6.11	10.35	7.59

3-day forecast / 05.03.2019 - 05.04.2019

loss	optimizer	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MSE	Adam(0.01)	20	128-64r-32r-16r	-0.33	5.23	14.58	9.16
MSE	Adam(0.01)	20	6r-6r	-0.54	7.46	15.7	10.61
MSE	Adam(0.01)	20	6r-18r-12r-6r	-0.13	6.03	13.43	8.91

7-day forecast / 05.03.2019 - 05.04.2019

loss	optimizer	epochs	structure	r2	median_abs_err	rmse	mean_abs_err
MSE	Adam(0.01)	20	128-64r-32r-16r	-0.44	10.55	25.34	18.21
MSE	Adam(0.01)	20	6r-6r	-0.34	13.78	24.4	18.34
MSE	Adam(0.01)	20	6r-18r-12r-6r	-0.09	11.95	21.98	16.62

Appendix D

Second generation NO2

First gen.	First generation NO_2 online learning model
Second gen.	Second generation NO_2 online learning model
Batch	NO_2 model trained using batch-learning
WT	Feature set: only wind speed and traffic (polynomial degree of 2)
PAR	Passive-Aggressive Regressor C=1, epsilon=5)
PAR(0.5)	Passive-Aggressive Regressor C=0.5, epsilon=5)
SGD	Stochastic Gradient Descent from Scikit-learn (default)
NN_small	Hidden layers: 6-6
NN_medium	Hidden layers: 6-18-12-6
KNeighborsRegressor	KNeighborsRegressor from Scikit-learn
SVR	Support Vector Regression from Scikit-learn
RandomForest	RandomForestRegressor from Scikit-learn

All metrics are based on observational data and forecasts for April 2019.

24-hour forecast

Type	**Model**	**r2**	**median_abs_err**	**rmse**	**mean_abs_err**
First gen.	PAR	0.13	13.80	21.26	16.34

First gen.	SGD	-0.26	11.33	25.55	17.79
First gen.	NN_small	0.14	11.78	21.14	15.72
Second gen.	PAR	0.14	11.57	21.18	15.75
Second gen.	PAR(C=0.5)	0.17	12.67	20.76	15.69
Second gen.	SGD	0.05	13.71	22.23	16.83
Second gen.	NN_medium	-0.27	19.18	25.69	20.40
Second gen.	NN_small	0.24	11.78	19.80	15.12
Second gen. (WT)	PAR	0.23	11.06	20.01	15.01
Second gen. (WT)	NN_small	-0.14	13.01	24.27	18.08
Batch	SVR	0.21	13.66	20.27	15.97
Batch	KNeighborsRegressor	0.12	14.57	21.33	16.84
Batch	RandomForest	-0.05	15.48	23.31	18.31
Batch	NN_small	0.11	15.23	21.51	17.25
Batch	NN_medium	-0.03	17.70	23.15	18.94
Batch (WT)	SVR	0.21	12.66	20.20	15.71
Batch (WT)	NN_small	0.19	12.27	20.54	16.12

3-day forecast

Type	**Model**	**r2**	**median_abs_err**	**rmse**	**mean_abs_err**
First gen.	PAR	-0.06	14.09	23.47	18.03
First gen.	SGD	-0.18	13.44	24.70	18.27
First gen.	NN_small	-0.26	13.79	25.61	19.39
Second gen.	PAR	0.00	11.98	22.80	17.01
Second gen.	PAR(C=0.5)	0.05	13.30	22.15	17.18
Second gen.	SGD	-0.04	13.89	23.18	17.86
Second gen.	NN_medium	-0.78	17.46	30.41	23.03
Second gen.	NN_small	-0.39	14.05	26.86	19.71
Second gen. (WT)	PAR	-0.16	14.85	24.54	19.25
Second gen. (WT)	NN_small	-0.85	16.61	30.94	22.81
Batch	SVR	0.08	15.22	21.82	17.55
Batch	KNeighborsRegressor	-0.24	16.13	25.40	20.18

Batch	RandomForest	-0.44	16.20	27.36	21.61
Batch	NN_small	-0.34	17.61	26.35	21.28
Batch	NN_medium	-0.59	20.17	28.74	22.95
Batch (WT)	SVR	0.08	15.26	21.80	17.59
Batch (WT)	NN_small	-0.13	17.23	24.18	19.25

7-day forecast

Type	**Model**	**r2**	**median_abs_err**	**rmse**	**mean_abs_err**
First gen.	PAR	-0.02	14.78	23.05	18.29
First gen.	SGD	-0.17	13.72	24.62	18.42
First gen.	NN_small	-0.27	19.25	25.63	21.12
Second gen.	PAR	-0.09	15.53	23.72	18.90
Second gen.	PAR(C=0.5)	-0.05	16.09	23.39	18.84
Second gen.	SGD	-0.11	13.78	24.04	18.54
Second gen.	NN_medium	-0.81	13.48	30.60	22.24
Second gen.	NN_small	-0.46	18.78	27.48	22.53
Second gen. (WT)	PAR	-0.18	20.41	24.70	20.56
Second gen. (WT)	NN_small	-0.66	19.78	29.30	23.85
Batch	SVR	0.03	15.26	22.40	17.92
Batch	KNeighborsRegressor	-0.36	18.11	26.59	21.42
Batch	RandomForest	-0.40	17.07	26.99	21.07
Batch	NN_small	-0.40	17.87	26.92	22.04
Batch	NN_medium	-0.63	22.43	29.10	23.83
Batch (WT)	SVR	0.03	15.71	22.49	17.92
Batch (WT)	NN_small	-0.18	16.21	24.78	19.98

Appendix E

Second generation PM10

First gen.	First generation NO_2 online learning model
Second gen.	Second generation NO_2 online learning model
Batch	NO_2 model trained using batch-learning
WT	Feature set: only wind speed and traffic (polynomial degree of 2)
PAR	Passive-Aggressive Regressor C=1, epsilon=5)
PAR(0.5)	Passive-Aggressive Regressor C=0.5, epsilon=5)
SGD	Stochastic Gradient Descent from Scikit-learn (default)
NN_small	Hidden layers: 6-6
NN_medium	Hidden layers: 6-18-12-6
KNeighborsRegressor	KNeighborsRegressor from Scikit-learn
SVR	Support Vector Regression from Scikit-learn
RandomForest	RandomForestRegressor from Scikit-learn

All metrics are based on observational data and forecasts for April 2019.

24-hour forecast

Type	Model	r2	median_abs_err	rmse	mean_abs_err
First gen.	PAR	0.42	5.27	10.33	7.33

First gen.	SGD_SEI	-0.15	7.97	14.60	10.47
First gen.	NN_medium	0.20	6.37	12.15	8.98
Second gen.	PAR	0.43	5.22	10.28	7.25
Second gen.	PAR(C=0.5)	0.33	5.27	11.09	7.71
Second gen.	SGD_SEI	-0.08	6.78	14.15	9.83
Second gen.	NN_medium	-0.09	6.59	14.21	9.98
Second gen.	NN_small	0.21	6.30	12.08	9.03
Second gen.	SGD	-0.23	5.99	15.07	9.96
Batch	SVR	0.02	5.27	13.44	8.83
Batch	KNeighborsRegressor	0.03	7.73	13.37	9.84
Batch	RandomForest	0.08	8.47	13.08	9.91
Batch	NN_small	0.10	8.00	12.92	9.93
Batch	NN_medium	0.26	6.95	11.70	8.77

3-day forecast

Type	**Model**	**r2**	**median_abs_err**	**rmse**	**mean_abs_err**
First gen.	PAR	-0.23	6.99	15.06	10.26
First gen.	SGD_SEI	-0.46	7.51	16.44	11.57
First gen.	NN_medium	-0.44	6.03	16.34	10.59
Second gen.	PAR	-0.15	6.02	14.59	9.82
Second gen.	PAR(C=0.5)	-0.17	5.99	14.69	9.77
Second gen.	SGD_SEI	-0.43	6.51	16.27	11.02
Second gen.	NN_medium	-0.79	7.36	18.20	12.31
Second gen.	NN_small	-0.50	7.31	16.66	11.67
Second gen.	SGD	-0.60	6.93	17.19	11.44
Batch	SVR	-0.11	6.21	14.31	9.57
Batch	KNeighborsRegressor	-0.41	8.21	16.16	11.41
Batch	RandomForest	-0.37	7.01	15.92	11.24
Batch	NN_small	0.03	7.29	13.40	9.97
Batch	NN_medium	0.01	7.45	13.51	9.91

7-day forecast

Type	Model	r2	median_abs_err	rmse	mean_abs_err
First gen.	PAR	-0.48	7.77	16.54	11.84
First gen.	SGD_SEI	-0.50	8.51	16.66	11.75
First gen.	NN_medium	-0.59	7.71	17.13	12.05
Second gen.	PAR	-0.42	7.71	16.20	11.67
Second gen.	PAR(C=0.5)	-0.40	7.49	16.10	11.33
Second gen.	SGD_SEI	-0.53	7.78	16.80	11.75
Second gen.	NN_medium	-0.87	11.82	18.60	14.64
Second gen.	NN_small	-0.61	8.21	17.28	12.67
Second gen.	SGD	-0.73	8.63	17.88	12.41
Batch	SVR	-0.18	6.03	14.80	9.86
Batch	KNeighborsRegressor	-0.73	8.21	17.91	12.05
Batch	RandomForest	-0.84	8.01	18.45	12.95
Batch	NN_small	-0.16	6.99	14.62	10.36
Batch	NN_medium	-0.31	7.53	15.55	11.04