Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Erling Van De Weijer
Odd André Owren

Forecasting Ambulance Demand in Oslo and Akershus

Artificial Intelligence and Statistical Methods

Master's thesis in Computer science
Supervisor: Ole Jakob Mengshoel
June 2022

NTNU
Norwegian University of
Science and Technology

Erling Van De Weijer
Odd André Owren

Forecasting Ambulance Demand in Oslo and Akershus

Artificial Intelligence and Statistical Methods

Master's thesis in Computer science
Supervisor: Ole Jakob Mengshoel
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Forecasting ambulance demand can be a critical tool for emergency medical services to allocate their resources as efficiently as possible. In this thesis, we use data provided by Oslo University Hospital (OUH) to forecast spatio-temporal ambulance demand in Oslo and Akershus. We perform different experiments inspired by related work both within spatio-temporal forecasting in general and ambulance demand forecasting. To forecast demand volume, we experiment with machine learning and statistical decomposition for time series to extract information about seasonal variations and the trend. We apply multilayered perceptron (MLP) and regression to forecast the residual and trend, respectively. We also experiment with a proposed error metric to reduce the underestimation of demand. We forecast at a high resolution and on different aggregations for spatial forecasts. For high resolution forecasts we experiment with Wasserstein generative adversarial neural network (WGAN) to generate realistic distribution scenarios, and MLP with different input sets. To produce aggregated data, we apply genetic algorithms with multi-objective optimization, using population data from Statistics Norway (SSB), and apply an MLP to forecast the hourly aggregated distribution. We conclude that our proposed models for volume forecasting outperform models used in the industry and models proposed in related research. Furthermore, WGAN does not generate good scenarios, and forecasting of distribution at high resolution are unable to outperform the historical average of our data. We make improvements with the aggregation of data as this makes the forecasts from MLP able to outperform the historical average by a slim margin.

Sammendrag

Evnen til å forutsi ambulanseetterspørsel er et kritisk verktøy innen akuttmedisin for å kunne fordele ressursene så effektivt som mulig. I denne oppgaven benytter vi et datasett gitt av Oslo Universitetssykehus for å forutsi romlig-temporal ambulanseetterspørsel i Oslo og Akershus. Vi gjennomfører flere ulike eksperimenter inspirert av relatert forskning innen både romlig-temporal prediksjon, samt prediksjon av ambulanseetterspørsel. For å forutsi volum av etterspørsel benytter vi statistiske dekomponeringsmetoder for tidsserier for å ekstrahere informasjon om sesongvariasjoner og trend og benytter flerlags perceptron (MLP) og regresjon for å forutsi henholdsvis rest og trend fra dekomponering. Vi gjennomfører også eksperimenter med en foreslått avviksmåling ment for å redusere underestimering av etterspørsel. Vi predikerer også fordeling av etterspørsel i rom både med høy oppløsning og med aggregert data. For høy oppløsning eksperimenterer vi med Wasserstein generative adversarial neural network (WGAN) for å generere realistiske fordelinger av hendelser, samt MLP med ulik input-data. For å lage aggregert data benytter vi genetiske algoritmer (GA) med multi-objektiv optimisering der vi bruker data fra Statistisk Sentralbyrå (SSB) og benytter MLP for å forutsi fordelingen basert på det aggregerte datasettet for hver time. Vi konkluderer med at våre foreslåtte modeller for volum av etterspørsel utkonkurrerer modeller i bruk innen industrien og modeller foreslått i relatert arbeid. Videre har vi at WGAN ikke genererer tilfredsstillende scenarier, og prediksjoner med høy oppløsning ikke er i stand til å utkonkurrere det historiske gjennomsnittet. Vi forbedrer prediksjonene ved å aggregere dataen slik at MLP er i stand til å utkonkurrere det historiske gjennomsnittet med en tynn margin på det aggregerte datasettet.

Preface

This thesis was written and carried out by Erling Van De Weijer and Odd André Owren to finalize their Master of Science in Computer Science degree at the Department of Computer Science, Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology (NTNU). The project was supervised by professor Ole Jakob Mengshoel at the Department of Computer Science, Faculty of Information Technology and Electrical Engineering, NTNU.

We want to thank Professor Mengshoel for his contributions to our project through weekly meetings, general research advice, and feedback throughout the project. We would also like to extend our thanks to Jo Kramer-Johansen, chief physician at the Norwegian National Advisory Unit for Prehospital Emergency Medicine (NAKOS), for presenting us with an exciting research topic and encouragement for our work.

Contents

Figures

Tables

Acronyms

AI artificial intelligence.

ANN artificial neural network.

ARIMA autoregressive integrated moving average.

BFS breadth-first search.

CCE categorical crossentropy.

DFO derivative-free optimization.

EA evolutionary algorithm.

EMCC Emergency Medical Communication Center.

EMD emergency medical dispatch.

EMS emergency medical service.

GA genetic algorithm.

GD gradient descent.

GM multivariate gray.

LSTM long short-term memory.

MAE Mean absolute error.

MLP multilayered perceptron.

MLR multiple linear regression.

MOEA multi-objective evolutionary algorithm.

MSE Mean squared error.

NAKOS Norwegian National Advisory Unit for Prehospital Emergency Medicine.

NSGA-II non-dominated sorting genetic algorithm II.

OUH Oslo University Hospital.

PNN Poisson neural network.

ReLU rectified linear unit.

SARIMA seasonal autoregressive integrated moving average.

SGD stochastic gradient descent.

SMA simple moving average.

SSA singular spectrum analysis.

SSB Statistics Norway.

STL seasonal and trend decomposition using LOESS.

WGAN Wasserstein generative adversarial neural network.

Chapter 1

Introduction

Emergency medical services (EMS) are a crucial part of modern health care systems. The EMS responds to emergency calls and provides pre-hospital care and patient transport. The response time of the EMS is an important part of how success is measured in this field of work and can be a decisive factor when it comes to the survivability of patients.

The quality of an EMS is affected by strategic, operational, economic, and political decisions. The decisions that affect EMS are shaped by the uncertainty of emergencies, such as call volume, severity, location, available units, and response time to the location of the emergency.

This thesis focuses on modeling ambulance demand in Oslo and Akershus, Norway. Doing so involves a lot of different challenges. There is a need for high-resolution forecasts for the results to be used in a realistic setting, but doing so results in sparse and noisy data. To overcome sparsity issues, we will try to reduce the resolution within reason to a point where the forecasts are less sparse but still not too low resolution.

1.1 EMS Response

The demand for EMS is ever-present in our society, and when dealing with medical emergencies, time is of the essence. The response time of an emergency vehicle has been widely understood to be a critical part of a patient's survival rate.

The implementation of EMS systems around the world differs a lot. However, for the Nordic countries, the systems mostly follow the steps presented in Figure 1.1, as presented in the Norwegian Health Department's report on the Nordic emergency model (Olsen et al., 2019). The steps of handling an emergency begin the moment an incident happens. Usually, there

might be some delay until emergency services are notified when the incident occurs. After this delay, an emergency service operator at the emergency medical dispatch (EMD) receives the call and is responsible for assessing the situation, urgency, and available resources while also providing instructions to the caller before deciding which resources to dispatch where. These resources include which ambulance to dispatch from which facility and which facility to deliver the patient. The selected crew will then need to acquire the necessary equipment and get in the ambulance to travel to the incident scene. The crew might spend some time locating the patient and reaching them at the scene. When located, the patient will receive medical care at the scene before being transported to the destination facility while the medical staff provides care underway. After arriving at the facility, the patient is transferred to the facility staff for further care. At the same time, the ambulance might need cleaning and replenishing of equipment before it is ready for the next deployment.

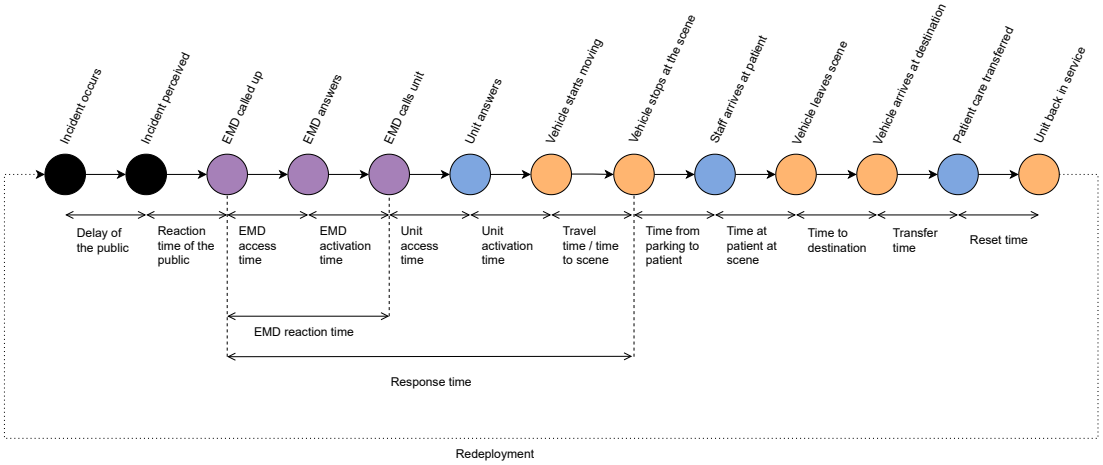

Figure 1.1: EMS-response timeline, adapted from Olsen et al. (2019). Response time, which is the time from EMD is called up to vehicle stops at the scene, is the focus of our research.

Currently, the time points in Figure 1.1 marked in purple are usually tracked automatically, while the points marked in orange are tracked manually by the ambulance personnel. Additionally, Olsen et al. (2019) suggests that the blue points should be tracked in the future to improve the ability to assess patient outcome measurements instead of response time as a sole measure.

1.2 Research Goal

Given a data set with response time being the most quantifiable measure to improve, most of our research will be focused on reducing response time. We will not study the locations of ambulances but are looking to provide data and forecasts for unplanned incidents that will give context while planning resource allocations for future EMS operations. The overall research goal of this thesis is, therefore:

Research goal *Provide methods for forecasting unplanned demand to improve the planning process for EMS operations.*

1.2.1 Research Question 1; High-Resolution Demand Forecasting

Our primary research question relates to forecasting the demand $\lambda(t) = \boldsymbol{u}^t \in \mathbb{N}^N$ where u_k^t is the forecast at time t in region k for $k \in 1, 2, ..., N$ and N is the number of spatial regions. We want to provide precise forecasts at a high spatio-temporal resolution as this gives the most insightful data for operations planning. As mentioned earlier, a limitation of high resolution is that the data becomes too sparse, especially in the spatial dimension. The following research question summarizes this:

Research question 1 *How can we forecast demand at a high spatio-temporal resolution?*

The spatial and temporal resolution for this thesis is at minimum 1×1km and 1 hour, respectively, due to the contents of our data set. We aim to test various sets of statistical and artificial intelligence (AI) models and assess how these perform in comparison to previously suggested models in related literature.

1.2.2 Research Question 2; Time Series Decomposition and Machine Learning

Hourly EMS demand is often assumed to have recurring patterns and cycles. These cycles are assumed to follow a daily, weekly, or yearly pattern, if not a combination of these. We are looking to explore this assumption further as we apply statistical decomposition to capture and extract these presumed elements, thus reducing the complexity of the problem space for our machine learning models to learn.

Research question 2 *Can time series decomposition be used in combination with machine learning models to produce improved total hourly ambulance demand?*

1.2.3 Research Question 3; Aggregating Spatial Data

Due to challenges related to sparsity in high-resolution spatio-temporal data, we investigate possible means to work around sparsity through aggregating data in the spatial dimension. Aggregation can help reduce the number of locations without events and thereby reduce the stochastic nature of our data.

Research question 3 *Can aggregation of spatial data improve forecasts of EMS demand?*

At a higher spatial resolution, we are prone to get a zero-inflated distribution which tends to be harder to use in forecasting. Another critical factor to consider here is not to aggregate too much, as large areas will lead to the predicted area of an incident being too general to provide valuable information for the Emergency Medical Communication Center (EMCC) department.

1.3 Report Structure

Chapter 2 presents relevant background and motivation for this thesis, which includes an introduction to a previous Master's thesis written on the subject, our pilot project for this thesis, and the provided data set. Chapter 3 introduces the relevant theory of the methods used in our experiments. Chapter 4 presents related research on demand forecasting, EMS demand, and spatio-temporal forecasting in general. In Chapter 5, we introduce our novel methods for time-series forecasting and spatial aggregation, as well as implementation details. Results from experiments with our methods are presented in Chapter 6 before we conclude and discuss our results and identify possible future work in Chapter 7

Chapter 2

Background and Motivation

In emergency medical situations, time is of the essence. Injuries, sicknesses, and medical conditions that require immediate treatment can occur in any place at any time. When requested, the EMCC provides necessary assistance in these situations. The assistance provided can vary greatly, depending on the situation and the available resources. Providing ambulance transportation is one of the many vital contributions of the EMCC. When an ambulance is dispatched for any mission, it will, in most cases, remain unavailable to respond to other missions until the assigned mission is completed. Since there are limited personnel and vehicles, efficient planning is a crucial tool in resource allocation. The current solution is based on the experience of the resource coordinators manually deciding which resource to assign to each incoming mission. Creating models to accurately forecast ambulance demand at both a temporal and spatial dimension could assist immediate and long-term resource allocation.

The EMCC in Oslo is the largest in Norway and answers calls from the Oslo and Akershus region with a population of 1.6 million. They receive over 500 000 telephone calls and provide over 150 000 ambulance dispatches per year.

2.1 Previous Research

Some work has been done on this problem with the same data set, such as a master's thesis from 2021 and our pilot project in the fall of 2021. We introduce both of these to provide some background for our thesis.

2.1.1 Previous Master's Thesis

In 2021 a Master's thesis was written on the same subject and data set. The thesis is summarized in the article by Haugsbø Hermansen and Mengshoel (2021) and introduces several challenges with ambulance demand forecasting. Sparsity is considered one of the main challenges when looking at distributional forecasts. Another problem related to volume forecasting is the stochasticity of the data. Haugsbø Hermansen and Mengshoel propose two different models for forecasting, a multilayered perceptron (MLP) model and a long short-term memory (LSTM) model. In this work, we implement these proposed models, re-create the results, and compare our models and findings to relevant findings of Haugsbø Hermansen and Mengshoel (2021).

2.1.2 Pilot Project

During the fall of 2021, we performed a pilot project for this thesis, where we had access to the same data set we are using in this thesis. The pilot project was performed as a literature review and data analysis of our data set to obtain a deeper understanding of the data we have available and different approaches from earlier research to apply to our problem. The main focus of the pilot project was the time series forecasting of our data, less so on spatial forecasting. We present most of our pilot project findings in this thesis, some of which are in the next section.

2.2 Data Set

The EMCC department at OUH, and the Norwegian National Advisory Unit for Prehospital Emergency Medicine (NAKOS) provided the data set used in this thesis. The data set contains an anonymized set of incidents from the 1st of January, 2015, to the 11th of February, 2019. The anonymization of the data set is achieved by mapping each incident to a 1x1km grid cell, acquired from SSB, and assigning the ID of the cell to the incident. The anonymization allows us to maintain a high resolution of the data while still preserving the anonymity of those involved in the incidents.

The initial processing of the data set consists of three steps: removal of duplicate rows, filtering events outside the years 2015-2019, and removal of events outside the borders of Oslo and Akershus. The filtered data set is presented in Table 2.1. We see a stable increase in incidents each year, except for 2019, where the data set ends in February. We can also see that most incidents are acute or urgent, making up 80.9% of total incidents in the data set.

Table 2.1: Summary of the filtered data set.

(a) Incidents per year.

Year	Number of incidents
2015	118 384
2016	135 749
2017	139 076
2018	146 416
2019	18 520
Total	558 145

(b) Incidents per level of urgency.

Urgency	Number of incidents
Acute	237 732
Urgent	213 520
Unplanned regular	56 489
Planned regular	50 404
Total	558 145

2.2.1 Incident Data with Urgency Levels

First, we begin by looking at the average distribution of different incidents throughout the hours of the week. The four different levels of urgency used in this thesis are as follows:

Acute *The most urgent of incidents, suspected to be a matter of life or death.*

Urgent *High urgency, should be handled as quickly as possible but can wait if more urgent incidents are being handled.*

Unplanned regular *Low urgency, usually queued for periods of low demand and can be handled multiple hours after a call is received.*

Planned regular *Planned events, such as transport for medical examination, outpatient transport from hospitals.*

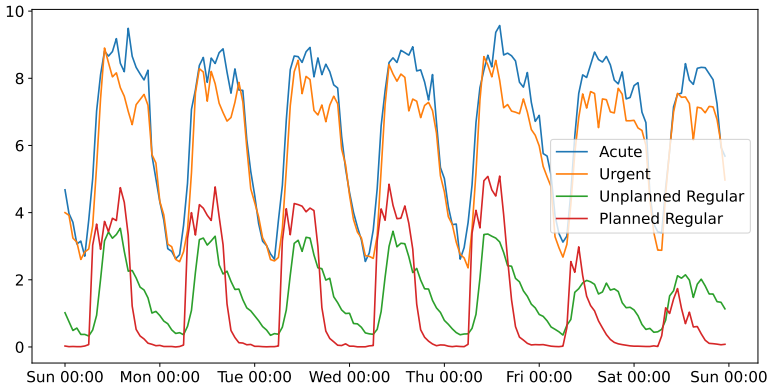

Figure 2.1: Mean number of incidents per urgency level throughout a week.

Looking at Figure 2.1, we can see that the regular unplanned and regular planned events are usually within the working hours on weekdays. We also see a higher demand generally during the day than during the night, which appears to follow a fixed interval of approximately 24 hours.

Presented in Figure 2.2 is the distribution of events per grid cell for the complete data set. We can see that there are some high-demand areas concentrated in and around the center of Oslo. These high-demand areas are not that surprising, as the population density is higher in this area.

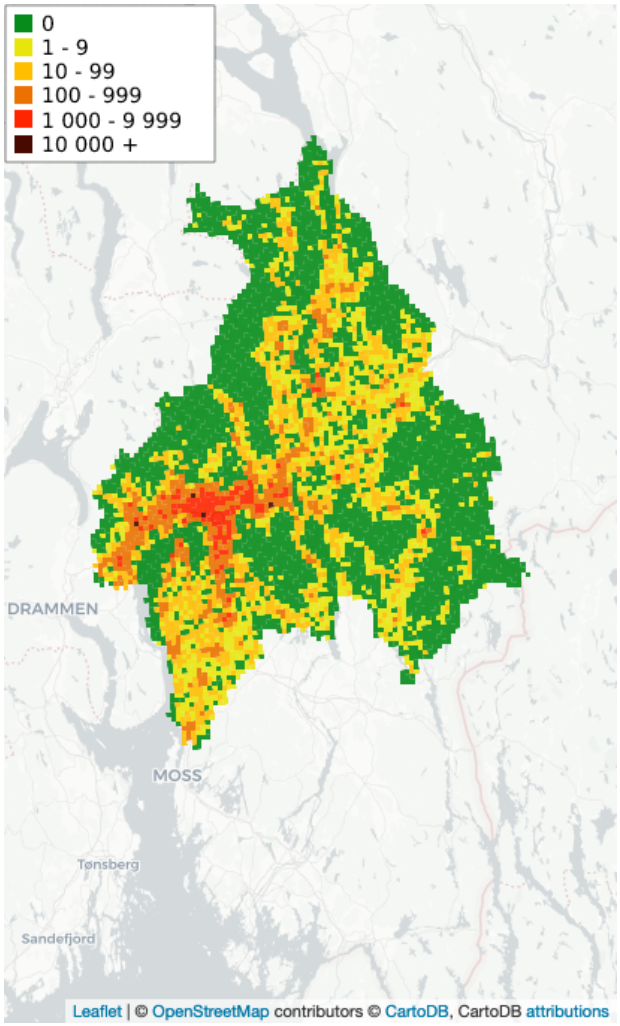

Figure 2.2: All incidents in the filtered data set, mapped out over Oslo and Akershus.

Furthermore, looking at the distribution per urgency level in Figure 2.3, we see the same as presented in Table 2.1; the acute and urgent incidents make up a large part of the data.

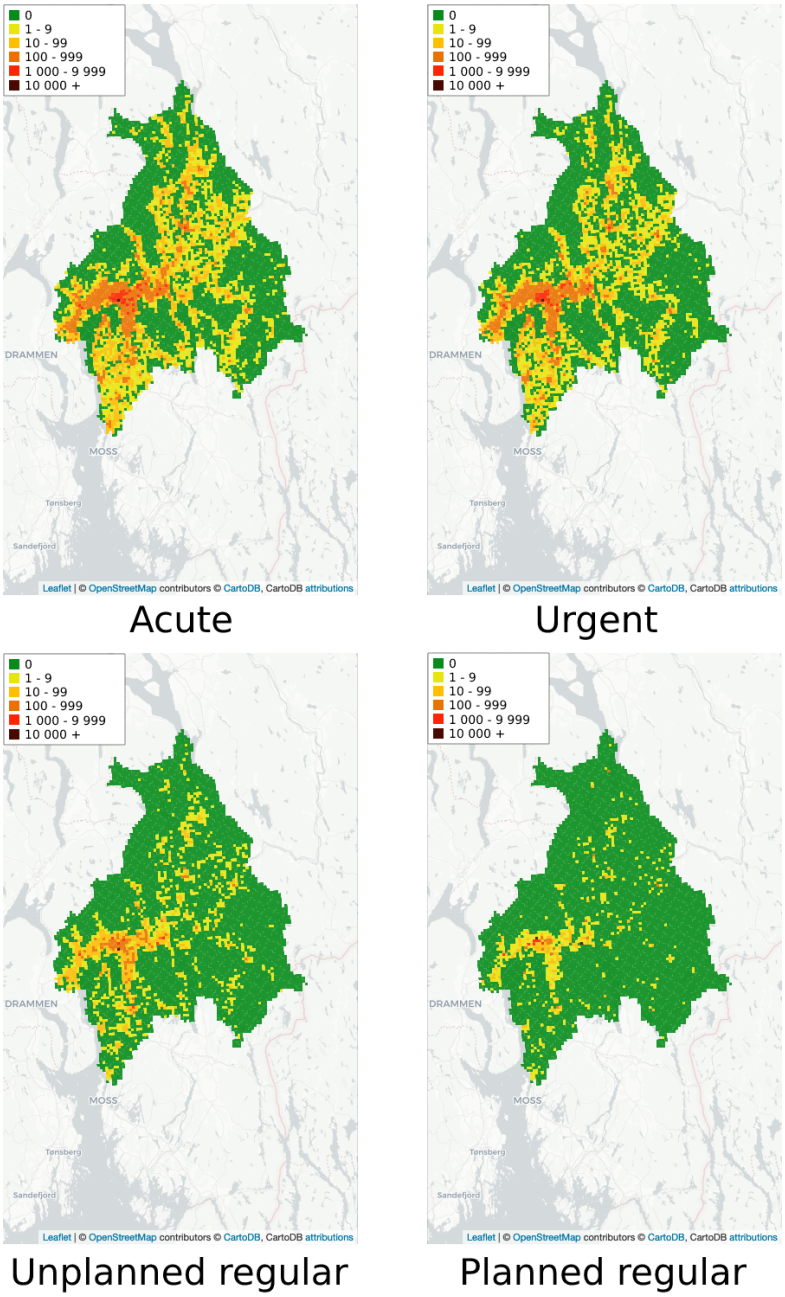

Figure 2.3: All incidents in the filtered data set, mapped out over Oslo and Akershus, split into each urgency level.

From the above distributions, we can see the problem with sparsity discussed in Haugsbø Hermansen and Mengshoel (2021), where about half the cells have zero incidents in the complete data set. There are at most 36 distinct cells with incidents in a given hour, out

of a possible 5569 cells. The number of cells with zero incidents is an issue we want to explore further in our thesis and look at ways we can overcome the issue with sparsity in the distributions of incidents.

2.2.2 Population Data

Looking at population data for a geographical area might help us better understand where high-demand areas are. Intuitively, incidents requiring medical assistance do not happen without people present, and there are bound to be incidents where there are people. Although the population data does not tell us where people are at any given time, it is a good indicator of where some incidents happen. Discussion with OUH has informed us that a large proportion of reported incidents happen in the person's own home.

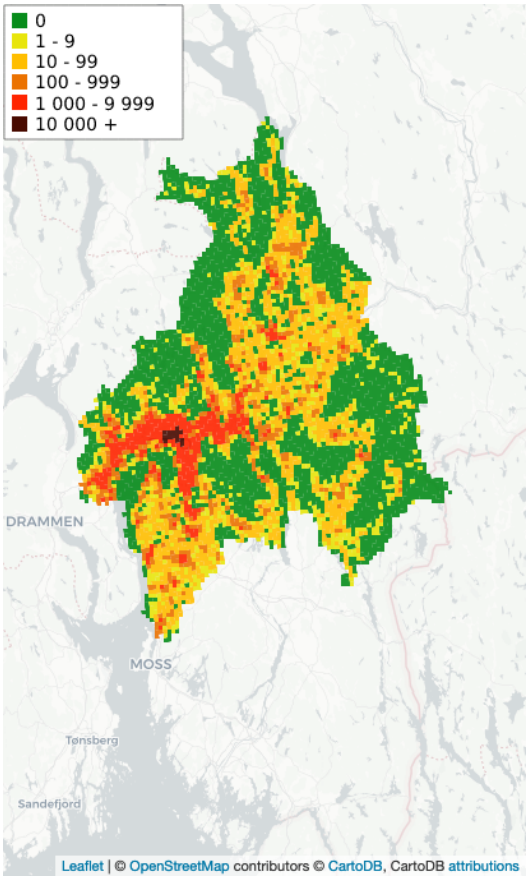

Figure 2.4: Population data for Oslo and Akershus. Data from Statistics Norway (SSB) [1].

[1] https://kart.ssb.no/share/7e9bd10aca46

As we can see from Figure 2.4, a lot of high population areas coincide with high-demand areas as presented in Section 2.2.1. Still, there are high-demand areas that do not have a corresponding high population. Looking further into this issue and comparing the different urgency levels, we see that several regular events are concentrated around hospitals in the region. We believe this is patient transport between hospitals or similar activity. Furthermore, we can look at the number of cells with different levels of population and demand in Figure 2.5.

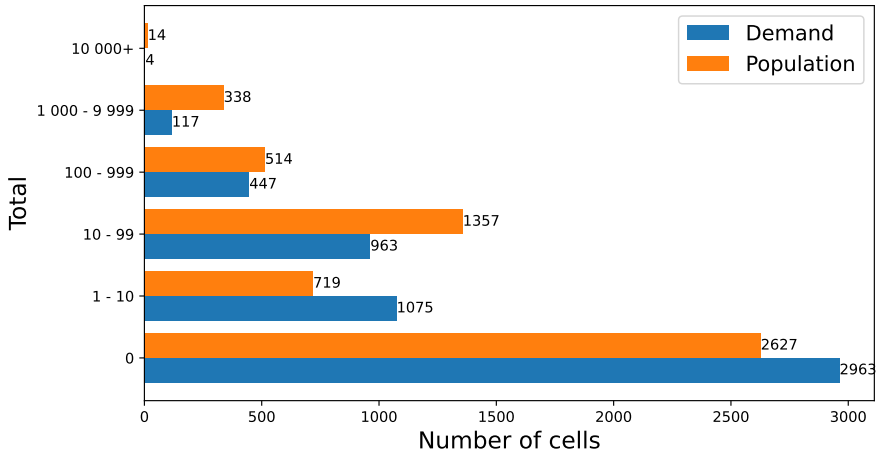

Figure 2.5: Distribution of grid cells with regards to demand and population in the data set.

We can see some disparity between demand and population in the cells. The relatively high disparity between the number of cells with 1 - 10 incidents and population is interesting. We believe these cells might be tourist routes or similar, where few people live, but more people visit throughout the year.

2.3 Initial Analysis of Data Set

As the first step in our analysis, we want to look at the autocorrelation of our time series. Autocorrelation is the correlation between lagged variables in a univariate time series, with one value c_k for each k level of lag giving the correlation between x_t and x_{t-k}. To find the autocorrelation, we begin with the autocovariance $\hat{y}_k$ between x_t and x_{t-k} given by Equation (2.1) where k is the lag.

$$\hat{y}_k := \frac{1}{n}\sum_{i=1}^{n-|k|}(x_{i+|k|}-\overline{x})(x_i-\overline{x}), \quad -n < k < n. \tag{2.1}$$

The autocorrelation function is then given by the autocovariance between x_t and x_{t-k} divided by the autocovariance of x_t with itself, as given in Equation (2.2). Here, $\overline{x}$ is the mean of $x_1, ..., x_n$.

$$c_k = \frac{\hat{y}_k}{\hat{y}_0} = \frac{\sum_{i=1}^{n-|k|}(x_{i+|k|} - \overline{x})(x_i - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}, \quad -n < k < n. \tag{2.2}$$

By plotting the autocorrelation of a time series, we can illustrate potential trends and seasonal components present in the series. If the autocorrelation increases or decreases with increased lag, we have indications of a trend in the series. Peaks in the autocorrelation at regular intervals will indicate seasonality with a frequency equal to the interval between the peaks.

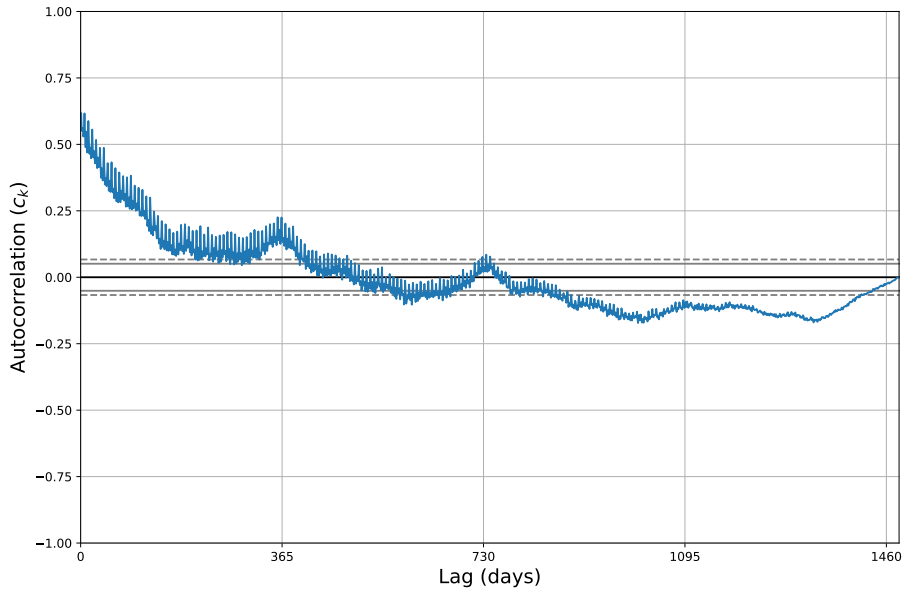

Figure 2.6: Autocorrelation of daily incidents for the full time period of the time series. Lags of 365 days are marked on the x-axis. See Figure 2.7 for higher detail from hourly incidents in March 2018.

From Figure 2.6, we can see some semblance of peaks at 365 days lag, indicating a seasonality of 365 days periods. The smaller but recurrent peaks at approximately seven days of lag are also worth noting, which are also present in Figure 2.7. Further, we see a downward slope in the autocorrelation, indicating an increasing trend in the time series. This is also

reflected in the data presented in Table 2.1 in Section 2.2.

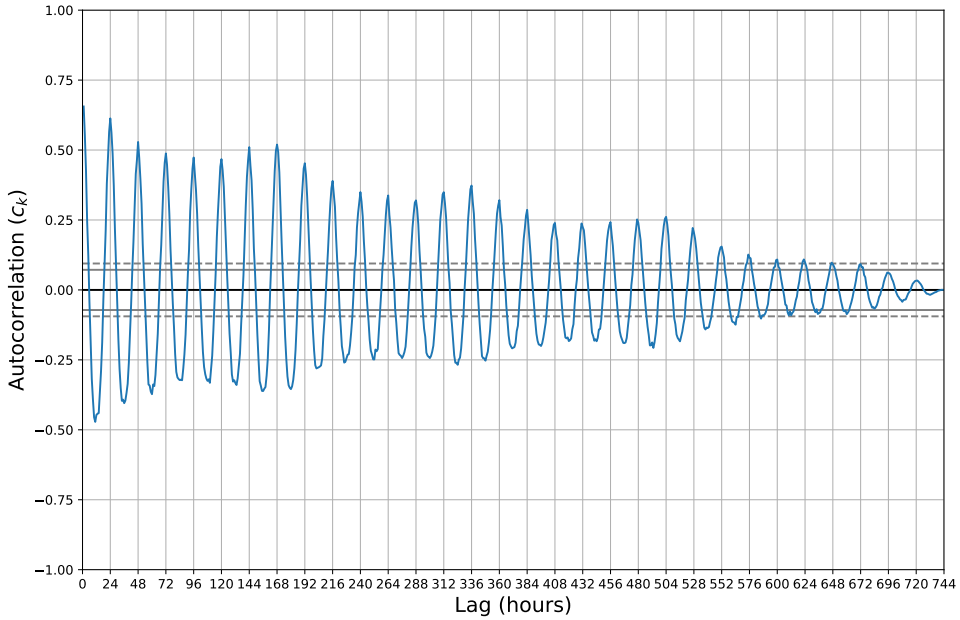

Figure 2.7: Autocorrelation of hourly incident, an excerpt from March 2018. This pattern is repeating for other months in the data set as well.

Looking at Figure 2.7, we see a strong tendency of seasonality at 24-hour periods and an even stronger seasonality at 168 hours, i.e., a week. The trend is close to constant as the period for the autocorrelation is too narrow. These findings indicate that further exploration of the seasonality and trend of our data is possible and might help us better understand the data and improve forecasting accuracy.

Chapter 3

Theory

Our thesis presents several models and concepts focused on forecasting ambulance demand. This chapter presents theoretical concepts and background that form the basis of our proposed models. Here we introduce theory related to time series, applicable statistical models, artificial intelligence methods, neural networks, and error metrics used in our research.

3.1 Time Series

This section gives an introduction to time series and methods used to analyze them, some of which are retrieved from the book *Introduction to Time Series and Forecasting* (Brockwell & Davis, 2002).

A time series is a series of observations $\mathbf{y} = [y_1, y_2, ..., y_t]$, observed over time, typically at regular periods of time called a discrete-time series. Here, y_t denotes the ambulance demand at time t for our time series and is a real positive integer. The resolution of the time series refers to the interval between observations. A time series is either univariate or multivariate. A univariate time series consists of a single value y_t observed at a time t. Any forecasting made on these observations is heavily based on the pattern of the time intervals. A multivariate time series contains multiple variables $x_t = [z_0, z_1, ..., z_n]$ observed at a time t. In other words, spatio-temporal data is a form of multivariate time series where the geographical location of the observation is one of the variables. By aggregating spatio-temporal data to a larger geographical area, the resulting time series might be univariate if no other variables are present. On the other hand, one could look at each geographical location independently, thus creating a collection of univariate time series.

A time series is said to be stationary if each observed value y_t is independent of other ob-

servations and the time of observation. In the case of a nonstationary time series, one can extract information about the observation, given that we know which time the observation was made. The dependence of an observation on the time of observation might be expressed as, e.g., a seasonal component, a recurring element at fixed time intervals. This notion is also introduced in Section 2.3 through the autocorrelation of a time series.

3.1.1 Time Series Decomposition

A nonstationary time series expressing seasonality and trend can be decomposed to $y_t = s_t + m_t + r_t$ where y_t is the observation at time t, s_t is the *seasonal* component, m_t is the *trend*, and r_t is the *residual* component. A trend is a pattern in data that shows a shift in the average values of a time series over a longer period. Seasonality refers to seasonal characteristics and predictable patterns in data and is not restricted to the four annual seasons. The residual component is the remaining variation not captured by the trend and seasonality components. Adjusting for the seasonal component leaves one with a residual and a trend component, both possible subjects to regression or machine learning. In EMS, a typical seasonality may be seen throughout the day, with nightly hours having a lower demand than during the day. As discussed in Chapter 2 and shown in Table 2.1 there is an increasing trend. Equation (3.1) shows how the forecast from residual and trend can be combined with the seasonal component, giving back a forecast $\hat{y}_{t+h}$, where p is the period of the seasonal component, h is the forecast horizon, and $n = \lceil \frac{h}{p} \rceil$.

$$\hat{y}_{t+h} = s_{t+h-np} + \hat{m}_{t+h} + \hat{r}_{t+h}. \tag{3.1}$$

To exemplify Equation (3.1), if we have 200 observed events, and a seasonality with a period of 168, and wish to predict 2 steps after the last observed event, we get $t = 200$, $h = 2$, $p = 168$, and $n = \lceil \frac{2}{168} \rceil = 1$, as values for our parameters. We thus get $\hat{y}_{202} = s_{34} + \hat{m}_{202} + \hat{r}_{202}$, such that the predicted value is the sum of the predicted trend at step 202, the predicted residual at step 202, and the seasonal component at step 34. Since the seasonal component repeats itself after every period of 168 steps has passed, the seasonal component at step 34 is predicted to have the same value as it would at step 202. On the other hand, both the trend and residual have to be forecast in another way.

3.1.2 Forecasting

Several methods exist for forecasting time series data, some simpler than others. Some examples of simple forecasting methods are simple moving average (SMA) and naïve forecast-

ing. A moving average amounts to analyzing data points in a time series by using the averages of different subsets of the data. There are several variations of moving average models. Some variations also make use of autoregression such as autoregressive integrated moving average (ARIMA) models, as well as seasonality such as seasonal autoregressive integrated moving average (SARIMA) models.

Simple Moving Average

In an SMA model, the last w values are used to create averages. For an SMA model with a window $w < t$, the value of the SMA can be calculated as shown in Equation (3.2).

$$\hat{y}_{t+1|t} = \frac{1}{w} \sum_{i=(t-w)}^{t} y_i. \tag{3.2}$$

Naïve Forecast

Naïve forecasts predicts a value observed h steps in the past. For $h = w = 1$, naïve forecast and SMA gives the same forecast. Naïve forecast for a time t and a horizon h is calculated as shown in Equation (3.3).

$$\hat{y}_{t+h|t} = y_t. \tag{3.3}$$

3.2 Error Metrics

Most AI research is dependent on a common ground to assess the results and capabilities of models. To quantify the error of a prediction compared to an actual sample, we use error metrics. The application of an error metric is different between different tasks and predicted values. Some error metrics are preferable when comparing two probability distributions, while others are relevant for single-value predictions.

3.2.1 Mean Absolute Error

Mean absolute error (MAE) measures the average absolute difference between a predicted and target value, equally punishing for overestimating and underestimating. This quality makes it suitable as an error metric for value forecasting and regression. MAE can be calculated using Equation (3.4)

$$MAE(\hat{y}, y) = \frac{\sum_{i=0}^{n} |\hat{y}_i - y_i|}{n}. \tag{3.4}$$

3.2.2 Mean Squared Error

Mean squared error (MSE) measures the squared difference between two values. Similar to MAE, it equally punishes for underestimating and overestimating the target value. Since MSE squares the difference, larger differences between the two values are punished more severely than with MAE. This property makes it a common error metric in machine learning models if it is a priority to avoid large differences between predicted and target values. The mathematical definition of MSE is presented in Equation (3.5)

$$MSE(\hat{y}, y) = \frac{\sum_{i=0}^{n} (\hat{y}_i - y_i)^2}{n}. \tag{3.5}$$

3.2.3 Categorical Cross-Entropy

Categorical crossentropy (CCE) is a measure of the entropy between two probability distributions, often used to minimize the difference between two probability distributions in optimization.

$$CCE(\hat{y}, y) = -\sum_{i=0}^{n} y_i \log \hat{y}_i. \tag{3.6}$$

3.3 Artificial Neural Networks

Biological structures and concepts have inspired many methodologies within AI, and artificial neural networks (ANN) are no different. Biological brains have inspired ANNs, and the artificial neurons make use of a lot of the concepts from how a biological neuron propagates information. The propagation of information from one neuron to the next depends on the set of inputs to the neuron. For each incoming connection to the neuron, a weight is multiplied by the activation value from the previous neuron. These weighted incoming values are then summed. Each neuron has an activation threshold, and when the sum from the previous neurons is above the threshold, the neuron fires its output. This output is then propagated to the next neuron with associated weights until the output neurons are reached. This process is presented mathematically in Equation (3.7), where w_{ij} represents the weight from neuron j to neuron i, x_j is the output from neuron j, and b_i represents the bias of neuron i.

$$z_i = \sum_j w_{ij} x_j + b_i. \tag{3.7}$$

An activation function decides the threshold of activation for a given neuron, and the activation function acts in two ways. The first part of the activation function is the threshold for activation. Some activation functions propagate z_i directly (linear activation). In contrast, other activation functions employ a ramp function such that all values z_i below a given threshold propagates a 0, and values above the threshold propagate z_i directly (ReLU activation). The second part of the activation function is what information to propagate. Some activation functions propagate the exponential values of z_i (exponential activation), while some are logistic functions (sigmoid activation). Which activation to choose depends on many factors, and the resulting network will display different properties based on the activation chosen for each node. Formally, a neuron's output x_i is given in Equation (3.8), where σ is an activation function.

$$x_i = \sigma(z_i). \tag{3.8}$$

3.3.1 Multilayered Perceptron

There are different ways of implementing ANNs, one of which is a fully connected feed-forward neural network structured in layers, namely an MLP. Single-layered neural networks consist of only one input layer and one output layer. However, these networks can only approximate linearly separable functions, e.g., classification tasks where one can separate two classes with a simple line. When the function is not linearly separable, we can use MLP to represent convex regions in high-dimensional space. It has been proven that multilayer feed-forward networks are universal approximators, even with only a single hidden layer, given a sufficient number of hidden units (Hornik et al., 1989). Figure 3.1 illustrates an example of a three-layer MLP, with three input nodes, one output node, and two hidden layers.

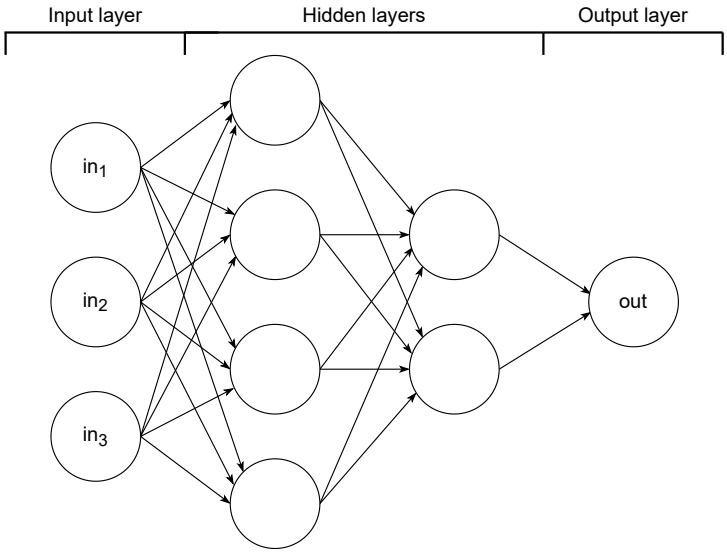

Figure 3.1: A simple MLP with one input layer, two fully connected hidden layers and an output layer consisting of one output node.

3.3.2 Supervised Learning

Even though one can design and implement neural networks with predefined weights and biases for each neuron, most modern neural networks apply some way of updating the weights and biases of the network. One way to update the weights of a neural network is through supervised learning. In supervised learning, we have a data set with a solution, or a target, for each input. The goal of the model is to create a mapping function $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \Theta)$, where $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T$ are the input values of n neurons and $\hat{\boldsymbol{y}} = [y_1, y_2, ..., y_m]^T$ are the output values of the m output neurons, having $\hat{\boldsymbol{y}}$ be an approximation as close as possible to the target $\boldsymbol{y}$. Θ represents the weights and biases of the network. Given that we know the true target of our data, the error between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ can be computed and used to update the weights and biases of the network. This process involves a loss function, $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$, which returns a measure of how different the output and target are. Some examples of possible loss functions are presented in Section 3.2.

K-Fold Cross Validation

K-fold cross-validation can be used to validate a proposed model using supervised learning. This method splits the data set into k subsets of equal size. We use $k-1$ subsets for training and the remaining subset for validation during the validation phase. This process is repeated

k times until all subsets have been used as validation sets. Using k-fold cross-validation is a way to better estimate the model's performance, as different training sets might uncover overfitting of the model on a given subset of data.

Overfitting

When approximating a mapping function from a set of inputs to outputs, we are looking for the most general model. This is analogous to polynomial regression for noisy data, where a lower degree is usually more general while higher degrees can perfectly fit every point of the observed values. We are more likely to have new observations closer to the generalized model than the complicated model if we introduce new observations. If we have two relatively equal models, we are inclined to keep the more generalized and straightforward model, following the principle of Occam's razor. Figure 3.2 presents a visual example of overfitting, where the actual function is the green line, and the approximated function is the red line, fitted to the initial observations marked in yellow. The approximation is not precise when new observations, marked in purple, following the actual function are presented.

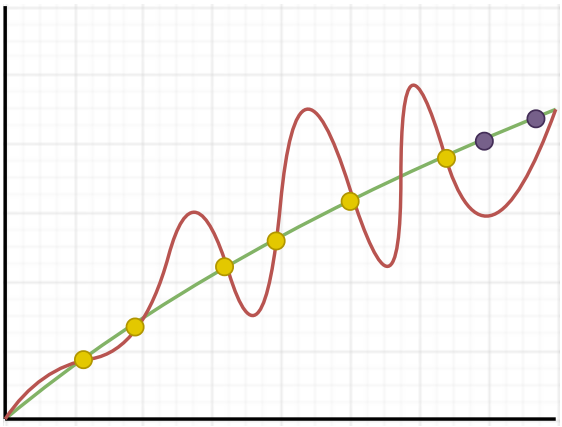

Figure 3.2: Illustration of the problem with overfitting. The true function is represented by the green line, the approximated function is the red line. Yellow points are initial observations and purple points are introduced after fitting the function to the yellow points.

One way to avoid overfitting is through the use of regularization techniques. One of the most commonly used regularization techniques in supervised learning is *early stopping*. A simple implementation of early stopping may use the validation set from a train-test-validation split of the data set to monitor the performance for each epoch of training. Given a value p for patience, if the model cannot improve within p epochs, early stopping is triggered, and the model stops the training process to avoid overfitting.

3.3.3 Optimizers

Optimizers are functions used to adjust parameters, to minimize an objective function. In an ANN, optimizers are commonly used to adjust weights within the network, where the objective function is the loss function. Most implementations of optimizers apply the gradient of the loss function and take a step in the opposite direction of the gradient. This process is used in the backpropagation algorithm to propagate the error to weights and biases through the network layers.

Gradient Descent

Gradient descent (GD) is a commonly used optimizer for ANNs. The weights w_j at iteration $i+1$ is updated based on the previous value with the learning rate α multiplied by the partial derivative of the loss function L with respect to the previous weight subtracted. One of the drawbacks of GD is that it is prone to finding local minima rather than a global. This update of weights is formulated in Equation (3.9), where θ_i represents the collection of weights at step i.

$$\theta_{i+1} = \theta_i - \alpha \frac{\partial}{\partial \theta_i} L(\theta_i). \tag{3.9}$$

Adam

Adam is an alternative optimizer to GD presented by Kingma and Ba (2014). The method was designed to combine advantages of *AdaGrad* (D. Xu et al., 2021) and *RMSProp* (Tieleman, Hinton et al., 2012). Similarly to GD, *Adam* uses the previous values of parameters and the learning rate multiplied by a function to update parameters. Rather than directly using the gradient of the loss function, *Adam* uses a momentum based function where m_i and v_i are moment vectors calculated using the gradient of the loss function, and ϵ is a small positive number to avoid division by zero. This update is formulated in Equation (3.10), where θ_i represents the collection of weights at step i and α is the learning rate.

$$\theta_{i+1} = \theta_i - \alpha \frac{m_i}{\sqrt{v_i} + \epsilon}. \tag{3.10}$$

3.3.4 Derivative-Free Optimization

In some cases, we do not have a differentiable loss function, but we still want to be able to perform weight updates to an ANN. Derivative-free optimization (DFO) is a technique

where the goal is to approximate a mapping function from a set of inputs to a set of outputs as before, without using information about the derivative of the loss function. Some examples of derivative-free algorithms are random search (Anderson, 1953), Bayesian optimization (Močkus, 1975), particle swarm optimization (Kennedy & Eberhart, 1995), and evolutionary algorithms (EA) which we will present in Section 3.5. The process of DFO is usually formulated as a search problem, where we search for optimal weights and biases to approach a given function. We still need some information about the network's output compared to the true target to evaluate and update weights accordingly. In the case of EAs, this is a fitness function that measures the performance of different settings of weights and biases in the network. Weight update is performed with an appropriate representation of weights that undergo the usual process of an EA to create new settings of weights and biases.

3.4 Optimal Transport Problem

In Section 3.2, we introduced an error metric used to compare two probability distributions, CCE. Even though this loss function is highly applicable to distributions, some limitations would be relevant to try to overcome. When applying CCE we are looking at the relative entropy between two distributions $P(x)$ and $P'(x)$. CCE does not take the relative distance between two points in the vector space of $P(x)$ and $P'(x)$ into account, also known as the Wasserstein distance, i.e., a measure of the minimum cost of transforming a probability distribution P' into a target P.

The Wasserstein distance can be formulated as an optimal transport problem, which is an optimization problem. This implies that to use Wasserstein distance as an error metric, we would have to know that our calculated distance is the optimal solution to the problem, which is not a guarantee. Instead, some implementations employ an ANN to estimate the Wasserstein distance, such as in Wasserstein generative adversarial neural network (WGAN) (Arjovsky et al., 2017). In this example, an ANN called a generator provides a distribution, and another ANN called the discriminator estimates the Wasserstein distance between the generated sample and the real sample. The estimated distance is then used to update weights during backpropagation in the networks.

3.5 Evolutionary Algorithms

Evolutionary algorithm (EA) is considered a modern heuristics-based search method and is a technique for optimization and search problems. Darwinian evolution has inspired the core

concept of EAs, often referred to as "survival of the fittest.". Evolution in EAs is done in much the same way as natural evolution; through selection, recombination, mutation, and evaluation. An EA allows for performing simulations of multiple generations where the selection is based on a fitness function, which is the heuristic intended to push the evolution in the right direction. Fitness functions are objective functions designed to evaluate individuals of a population, subject to either maximization or minimization. The score of the fitness function is used to select the fittest individuals to include in the next generation of individuals, usually with some randomness included.

Chapter 4

Related Work

This section presents previous work related to our problem area. In Section 4.1, we look at work on demand forecasting in several domains, providing a general overview of the usefulness of demand forecasting. Section 4.3 introduces a more specialized case of demand forecasting, namely spatio-temporal forecasting, looking at different resolutions of spatio-temporal data. As there is already done research on ambulance demand forecasting in Oslo, this will also be reviewed in this chapter.

4.1 Demand Forecasting

Many industries and services require precise demand forecasts to make short- and long-term operational decisions, and AI has proven to be a powerful tool in demand forecasting. Typical methods used to forecast demand are regression models, neural network models, or a combination of these. Regression models have been successfully applied to forecast attendance to soccer matches (Yamashita et al., 2022), demand in e-grocery (Ulrich et al., 2021), short term electrical load demand (Y. Chen et al., 2017; Fattaheian-Dehkordi et al., 2014), urban water demand (Brentan et al., 2017) and tourism demand (K.-Y. Chen & Wang, 2007). Similarly, neural network models have been successfully applied to forecast station-free bike sharing demand (C. Xu et al., 2018), long-term reservoir inflow (Herbert et al., 2021), outpatient department demand (Jiang et al., 2017) and cash demand in ATMs (Venkatesh et al., 2014).

Y. Zhang et al. (2021) introduces a decomposed deep learning approach to forecast tourism demand. The authors argue that two underlying problems are limiting the potential of deep learning models for predicting tourism demand; limited access to data volumes and

additional explanatory variable requirement. The proposed solution uses a decomposition method, STL (Cleveland et al., 1990), decomposing the tourism volume into three components: trend, seasonality, and residual. The deep learning model is then trained on trend and residual, giving back a prediction for both, combined with the seasonality component to provide a total volume prediction.

4.2 Time Series Decomposition and Forecasting

As discussed in Section 3.1.1, a nonstationary time series is possible to decompose into a set of components, usually in the form of a seasonal component, trend component, and residual. Y. Zhang et al. (2021) makes use of STL to accomplish this decomposition. In comparison, Basak et al. (2017) aims to improve the STL-decomposition to preserve extrema in time series smoothing in soil moisture analysis. The proposed model is a novel method, HyperSTL, implementing a stochastic optimization to minimize an objective function over the residual and trend. This objective function is composed of three weighted terms. The first term contains the residual variance. The second term includes the residual range to account for extrema carried over to the residual. The final term is a measure to keep the trend component smooth. The optimization result is a smooth trend component replicating the time series while also providing forecasts on the time series. Basak et al. (2017) notes that the objective function is not the only possible function, and other applications of HyperSTL might need a different objective function. Regarding ambulance demand, this approach might require a different objective function. The approach to preserve extrema, especially peaks, is an important trait in our problem area, considering a forecast of demand indicates the required ambulance crew in a given period and having too few crew members available could have dire consequences.

Another decomposition method is singular spectrum analysis (SSA). SSA is a non-parametric spectral estimation method that can be used to decompose a time series into a sum of components. As opposed to STL, SSA does not require the period length to be predetermined. Basic SSA consists of two stages: decomposition and reconstruction, each with its own two steps. First, map the time series to a trajectory matrix. Then decompose the matrix using single-valued decomposition. The third step is grouping these matrices. Finally, applying diagonal averaging produces a reconstructed series. SSA has successfully been applied for decomposition of rainfall times series (Wu et al., 2010). In this paper, a modular artificial neural network (MANN) was used to predict the residue from the decomposition. The model was measured against several other forecasting methods and outperformed every other method.

4.3 Spatio-Temporal Demand Forecasting

With a worldwide increase in population and urbanization, spatio-temporal data forecasting has proven to be a useful tool with numerous applications within cities. Spatio-temporal forecasting is the science of predicting when and where something will happen. Applications include traffic forecasting (Yu et al., 2017), solar power forecasting (Tascikaraoglu et al., 2016), and ambulance demand forecasting. When discussing spatio-temporal forecasting, the resolution is an important factor. The following sections discuss different approaches to spatio-temporal forecasting at different resolutions.

As spatio-temporal forecasting is a general term used for several different applications, there are not any formal definitions of what counts as high or low resolution. The resolution necessary to provide valuable information is highly domain-dependent. For example, temporal forecasting in financial markets requires a precision of minutes or even seconds (Zanc et al., 2019). Any temporal forecasting predicted with a lower resolution than minutes could thus be considered low resolution. As for ambulance demand forecasting, temporal resolutions of minutes are not feasible, and to our knowledge, there is no research done with a higher temporal resolution than hourly forecasts. Still, it may be helpful to assign high or low resolution to forecasts, as the information gathered from high and low resolution differs. We define high temporal resolution as a maximum of 4 hours and low resolution as 24 hours and above. Similarly, high spatial resolution is at a maximum of 4×4 km, and low spatial resolution is defined as entire regions at a time, usually cities and municipalities or even entire counties. We also consider research focused on only time-series forecasting for an entire city as low spatial resolution, even though the spatial factor is not considered.

Some related work regarding ambulance demand forecasting introduces the MEDIC method as a commonly used method in the industry and is a good baseline for evaluating forecasting models. The MEDIC method is a modification of the moving average, where similar hours from past weeks and years are used to forecast the demand. More specifically, MEDIC takes the average number of incidents for the same hour of the week for the past four weeks and from the same weeks of the past five years. This results in 20 observations assumed to be relatively similar to the one we want to forecast. For example, if we want to forecast incidents between 12:00 and 13:00 on the Monday of week 42 in 2022, we look at the number of incidents between 12:00 and 13:00 on Monday of weeks 38-41 of 2018-2022.

4.3.1 Low Spatial and Temporal Resolution

Low-resolution spatio-temporal forecasting has been used to examine how weather affects the daily demand for ambulance services in Hong Kong (Wong & Lai, 2012). Using regression analysis, Wong and Lai found a few significant causal relationships between weather factors and daily ambulance demand. The study found that weather impacted demographic groups differently. Older people and patients with severe conditions were more prone to the weather than younger people.

Huang et al. (2019) presents an approach to forecasting daily ambulance demand in the city of Ningbo, China. This approach is a combined model applying neural networks, genetic algorithms, and regression models. The proposed solution assumes that the data follows a Poisson distribution, i.e., the probability of observing X incidents in a given period. With this assumption, a Poisson neural network (PNN) is implemented with an exponential activation function, which gives a probability density function that is subject to maximizing. The initialization of the weights in the network is done with a genetic algorithm, where the weights and thresholds of the network are represented as the chromosome, and the fitness function is given by minimizing the inverse logarithmic probability density function, i.e., $F = \frac{1}{ln(P)}$. The output of the network is combined with three regression models: multiple linear regression (MLR), ARIMA, and multivariate gray (GM) to estimate the residual deviation, and a weighted average is applied to the output from the three models. The input of the neural network includes data for the date, weather conditions, and incidents in the previous six days. We find the approach interesting to explore further. However, we are inclined to evaluate whether the assumption of a Poisson distribution holds or not without any further preprocessing of our data.

4.3.2 Low Spatial and High Temporal Resolution

An example of low spatial and high temporal resolution forecasting is the work of Matteson et al. (2011). In this paper, the authors explore forecasting hourly demand in Toronto based on data from 2007 to 2008. With Setzler et al. (2009) as a baseline, the authors generate improved forecasts of the hourly demand, using integer-valued time series with a dynamic latent factor structure. Smoothing splines are incorporated to estimate the factor levels and loading to improve the long-term forecast.

4.3.3 High Spatial and Low Temporal Resolution

A. Y. Chen et al. (2016) use support vector regression, artificial neural networks, and regression to forecast ambulance demand for New Taipei City. Forecasting is done for both 3-hour buckets and daily demands for 3km × 3km and 2km × 2km, respectively. The spatial and temporal resolution was chosen based on an attempt to find a balance between accuracy versus data sparsity. The models were trained and tested on data from 2010 to 2012 with about 140 000 incidents per year. As for the model input, seven cases are presented with a different combination of input parameters. Some of these input combinations include a feature for weekends and the day of the week. The argument for adding this extra parameter, even though it is implicated by the day of the week, is that adding this weekend feature adds an obvious measure for the model as the weekend demand differs noticeably from weekdays. The same arguments are used to explain the addition of a season input feature, being a number from one to four, representing spring, summer, fall, and winter. The authors believe that EMS managers can use the results from the paper to allocate ambulances. As for future work, they believe that more input features could be included to increase the model's accuracy. These features include traffic conditions, national holidays, temperature, and population density.

4.3.4 High Spatial and Temporal Resolution

Setzler et al. (2009) designs an ANN to forecast ambulance demand in Mecklenburg County, North Carolina, USA. The ANN predicted ambulance demand for several combinations of 1-3 hour time buckets and 2-4 square mile grids. The neural network follows a feedforward multilayer perceptron model. A single hidden layer is added to the model to capture the nonlinear relationships between the input and the output. The authors argue that there is no significant benefit to the model of adding more than one layer in causal forecasting. This hidden layer consists of four hidden nodes. Compared to the MEDIC method, the proposed model scored slightly better on a 4×4-mile grid. It scored evenly on a 3×3-mile grid and slightly worse on a 2×2-mile grid. In addition to this, the authors found that forecasting only zeros outperformed both models.

Zhou and Matteson (2015) proposes a spatio-temporal kernel density estimation (stKDE) to address the challenges of high spatial and temporal demand forecasting. Kernel density estimation has been successfully applied to several different areas of spatio-temporal forecasting, such as crime incidence (Brunsdon et al., 2007; Nakaya & Yano, 2010), disease spread (Wilesmith et al., 2003; Z. Zhang et al., 2011) and data stream (Aggarwal, 2003; Procopiuc & Procopiuc, 2005). The authors model Toronto's ambulance demand over a con-

tinuous spatial domain and a temporal domain of one-hour intervals. A bivariate spatial kernel is placed at the location of each past observation, weighted with a weight function. The weight function aims to capture the usefulness of each past observation for predicting demand in a future period. Each location in the spatial domain is placed into a grid cell of 5×5 km. Within each cell, it is assumed that the usefulness of past observations to predict future events is only dependent on how far back in time the observation is. The resulting model scored highly on forecast accuracy while requiring low computational power.

4.3.5 Ambulance Demand Forecasting in Oslo and Akershus

Using the same OUH data set as this thesis, Haugsbø Hermansen and Mengshoel (2021) explore different machine learning methods for predicting the ambulance demand in the Oslo area. The authors use split and combined machine learning models to forecast demand at hourly intervals and 1 × 1 km grid cells. For the volume and complete models, different combinations of the number of hidden layers and nodes in each layer were tested for both MLP and LSTM. Volume models also include both MLP and LSTM, as well as the average distribution of the demand for the entire test data. Additionally, the authors wanted to see if online learning could improve their models.

The authors tested their models with four different sets of inputs. These sets consisted of a combination of the hour, day of the week, day, month, precipitation, and temperature. All input features based on time were one-hot encoded. At the same time, precipitation and temperature were encoded as floats. They included the day's weather divided into 3-hour intervals, resulting in 16 input features for each time slot.

The authors implemented MEDIC, Setzler et al. (2009), and all zeroes as baselines to compare their models to. The results from the study suggested that split models were superior to the complete models. The authors suggested that this was due to the split models learning two more straightforward problems than the combined model, which had to learn one single problem composed of the two, and the relationship between them. The best performing volume model, surpassing the performance of both MEDIC and Setzler, was an MLP using hour, day of the week, and month as input features. As for the distribution models, they did not improve upon the baseline of averaging all events across the data set. All machine learning models were improved when online learning was applied.

4.4 Spatial Aggregation

A lot of related work presented in the previous sections is somewhat hung up on the concept of grids of both the original size and different sizes. An issue with this is noted by Setzler et al. (2009); zero-inflated demand distributions are created when scaled to finer degrees of spatial resolution. Setzler et al. (2009) suggest that future research should focus on varying population densities in order to determine optimal or near-optimal geographic grid size and time intervals.

This notion is supported by Martin et al. (2021), who argues that "while fixed geographic grid layouts provide a straightforward approach to spatial segmentation, the equal division of space does not account for the unequal distribution of populations and associated call demand.". Martin et al. (2021) applies K-means clustering to create 7, 8, and 9 clusters based on call volume density and population density while maintaining a low Euclidean distance from the centroid to each sample in the cluster. The authors then apply an MLP-model to each cluster and perform time series forecasting, outperforming MEDIC, SARIMA and Holt-Winther's exponential smoothing method.

A different approach to spatial aggregation concerning the domain of ambulance optimization is optimizing the location of ambulances on standby. Aytug and Saydam (2002) proposes an approach using a genetic algorithm (GA) to solve a maximum expected covering location problem. The authors design a GA to apply to a set of generated test problems, and the results are compared to a random search measuring the frequency of solutions reaching a given percentage of an optimal solution. The findings of Aytug and Saydam (2002) underline how a GA can create a coverage segmentation with a set of given constraints and objectives to optimize and how well it outperforms a basic random search.

4.5 Discussion

As discussed in Chapter 1, we aim to provide forecasts at a high resolution as this yields the most useful information. As Haugsbø Hermansen and Mengshoel (2021) already have researched forecasting ambulance demand in Oslo and Akershus, using the same OUH data set we are, there is much inspiration to be gained from their paper. The authors argue that split models were more suited than complete models for this problem area. The assumption that it is easier to train two distinct models, giving them each a more straightforward task, rather than one model a more difficult task, seems reasonable and will be used going forward. The input features that yielded the best volume prediction results in this study consisted of the hour, day of the week, and month as one-hot encoded values. We believe there are

possibly other input features that can be included to improve forecasts and therefore want to explore this area further.

Given the indication of seasonality within the data set, creating a hybrid model using time series decomposition and neural networks could be an interesting approach. STL was successfully combined with machine learning in Cleveland et al. (1990), and we believe this, in addition to SSA, could be interesting to explore further. We believe that PNN proposed by Huang et al. (2019) cannot be directly used for hourly predictions, as the hourly ambulance demand volume does not seem to be stationary. Additionally, we can see that the ambulance demand increases each year, and both these features go against the assumption of our time series following a Poisson distribution. Still, we are inspired by the idea presented and want to incorporate the main ideas of the method, but without the parts associated with the Poisson assumption. This idea is further explored in Chapter 5.

Looking at the applied methods of K-means and GA to solve problems related to ambulance coverage, it is interesting to explore this area for our forecasting models further. Although we do not aim to solve a maximum expected covering location problem, we are interested in ways to better distribute population and call volume to create an equal distribution, thereby reducing the number of locations with 0 incidents.

Chapter 5

Prediction Methods

The goal of our research is divided into three research questions. For each question, we provide models that will allow us to answer these questions. Our primary research question concerns forecasting at the highest resolution of our data set, which is 1×1km grids and hourly periods. We are also interested in working with spatial aggregations and studying different aggregations that are still reasonable to use for OUH.

In this section, we propose and implement various models for forecasting, both spatial and temporal, and models for aggregating spatial data. These models are compared to each other and models proposed in related work looking at the same problem area.

Section 5.1 explains our preprocessing of the data set and initial analysis based on a set of different statistical models, such as STL. These statistical models allow us to produce stationarized time series, in which we use the residual component to perform predictions with neural networks to try to improve forecasts.

A common factor for our models is the necessity of data input. Different sets of input features are required or preferred in different situations. Table 5.1 lists all the features we use as input for our machine learning models. We will exclusively represent all our inputs as a one-hot encoded vector except for the year. We will represent the year either as a one-hot encoded vector or a numerical value between 0 and 1. Since 2015 is the first year present in our preprocessed data set, we will subtract 2015 from the year, such that the year will be a value between 0 and 4. We then divide the number by 4, such that the final representation is a decimal between 0 and 1. We will refer to the year's one-hot encoded and numerical representations as (Y+O) and (Y+N), respectively. Table 5.2 shows the resulting representations of the year as an input feature.

Table 5.1: Different representations of data used as input to MLP.

Data	Encoding
Hour of the day (H)	One-hot encoded
Day of the week (D)	One-hot encoded
Week of the year (W)	One-hot encoded
Month (M)	One-hot encoded
Year (Y+O)	One-hot encoded
Year (Y+N)	Numerical value in the interval [0, 1]

Table 5.2: Representations for year as a one-hot encoded vector and numerical value.

Year	One-hot encoding (O)	Numerical value (N)
2015	[1,0,0,0,0]	0
2016	[0,1,0,0,0]	0.25
2017	[0,0,1,0,0]	0.5
2018	[0,0,0,1,0]	0.75
2019	[0,0,0,0,1]	1

5.1 Preprocessing and Data Analysis

Experiment goal *Further explore the data and apply statistical methods and analysis to propose ways to extract recurring elements, constant variables, and noise to simplify the problem space.*

Data *Filtered data set summed for each hour of the period January 1st 2015-February 11th 2019.*

As introduced in Section 2.1.2, a pilot project was conducted before the beginning of this thesis. The aim of this pilot project was the same as the experiment goal presented above. A simplified method of our pilot project is presented in this section, as we base a lot of our experimental methods on the pilot project's findings. The findings of our pilot project are presented in Chapter 6.

The initial filtering of our data set included removing incidents outside Oslo and Akershus, removing data from before 2015, and removing duplicate incidents. In addition to our initial filtering, we also remove all planned regular events from our data set. Since these events are planned, we do not see the need to forecast them. This step was also done by Haugsbø Hermansen and Mengshoel (2021). After the preprocessing of the data is done, we can analyze

the data. The initial analysis is mainly presented in Section 2.2 and Section 2.3 and includes some visualizing of the data to provide an impression of the data itself.

From the pilot project, we found indications of seasonality and trend in the data set, which led us to apply STL to our data to decompose the time series and extract recurring incidents, constants, and residual noise.

5.1.1 STL-Decomposition

One of the main experiments from our pilot project was the application of STL-decomposition to our data and using the findings from this to further understand the data set. When using STL for decomposition, a predefined period must be set. Our initial method consisted of multiple decompositions at different periods, such as daily, weekly, and yearly. This proposal is based on the indications of seasonality at periods of 24 hours, 168 hours, and 365 days. After some experimenting, we discovered that a seasonal period of 168 hours was able to capture the period of 24 hours. Additionally, the annual seasonality was found to be negligible, contrary to what we initially assumed based on the autocorrelation in Figure 2.6.

Further, we explored how the trend of our data behaves, which was also an indication of the autocorrelation. Initially, we used the trend from the weekly decomposition. However, we later found that doing a 2-step decomposition where we attain the seasonality of period 168 hours in the first step and trend of period 365 days in the second step gave the most feasible trend and residual for further forecasting.

5.2 Volume Forecasting Methods

Experiment goal *Create new models for forecasting ambulance demand volume at hourly intervals, surpassing the accuracy of previous forecasting models.*

Data *Data used for these experiments are different combinations of the input features mentioned in Table 5.1 as well as the hourly demand volume collected from the OUH data set.*

Inspired by the work of Huang et al. (2019) and Y. Zhang et al. (2021), we explore a machine learning model inspired by PNN and time series decomposition that can be used to provide accurate volume forecasts for ambulance demand in Oslo and Akershus. Y. Zhang et al. (2021) use STL for decomposition. Since SSA also seems like a promising tool for decomposition, we try this as well. We know from our pilot project that our data has an increasing annual trend. Thus we wish to see if incorporating the year in the input data can improve volume forecasts.

The design of our MLP is inspired by the MLP architecture presented in Haugsbø Hermansen and Mengshoel (2021), as these models have proven to provide good results on our data set. One exception to this is the use of linear activation functions instead of rectified linear unit (ReLU) in models that forecast using decomposed data since ReLU cannot predict values below zero. Usually, this is not an issue in demand forecasting. However, after seasonality and trend are extracted, the residual will have a mean close to 0, with a deviation allowing for negative values. Since we want our models to predict these values as well, we need to use an activation function that can have negative values as output, which is why we use a linear activation function. This change is only done to the final layer of our model to keep the architecture as close as possible to the original.

These experiments will be performed with the filtered data set with events until the end of the year 2018. This data is split into training, validation, and test sets. We use the first 80% of the data for training, the following 10% for validation, and the final 10% for testing. Figure 5.1 shows how the data is distributed, as well as the corresponding demand.

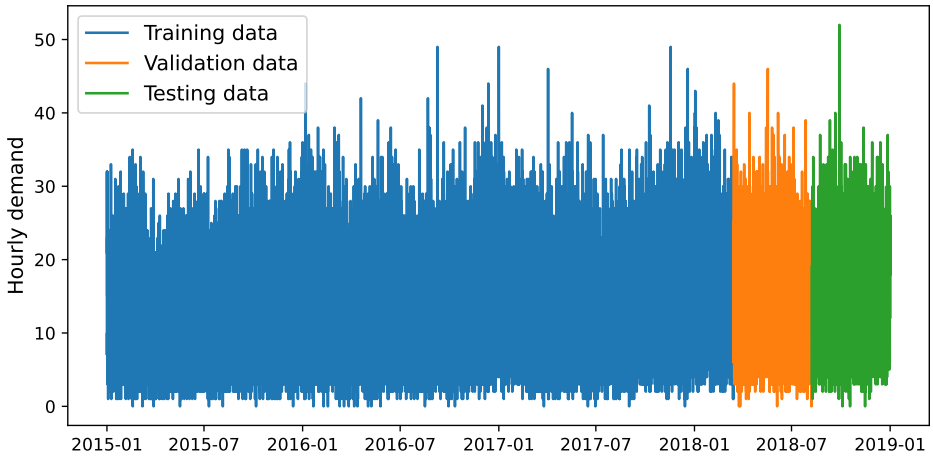

Figure 5.1: Distribution of hourly incidents into train, validation, and test set.

5.2.1 Varying Data Input with MLP

Experiment goal *Explore different input sets, and how they affect MLP predictions.*

From our data analysis, we can see that the ambulance demand is increasing annually. We believe that including the year as one of our input features in our neural networks can be used to improve our models. As the trend is almost linear, we will experiment with including the year both as a one-hot encoded vector and as a numerical value between 0 and 1.

Our experiment will consist of comparing three different input features in an MLP model with the same architecture but different input layers. The different input sets will be [H, D, M], [H, D, M, (Y+O)] and [H, D, M, (Y+N)]. The first input set corresponds to the *Basic* input set in Haugsbø Hermansen and Mengshoel (2021), which was the best performing input set in their research. The two other input sets will be new for our data set and will be our attempt at improving model prediction. We use a 5-fold cross-validation with the training data as input and score the models based on the average loss for the fully trained model for each fold. We implement early stopping with our validation set and patience of 5 to avoid overfitting during training.

5.2.2 STL-Decomposition and Neural Networks

Experiment goal *Explore how STL-decomposition can affect performance of time series forecasting of an MLP.*

Using STL, we aim to extract a seasonality and trend from our time series so that we can train our MLP on the residual. From Section 5.1, we found that using a weekly period seemed most promising, but we will also look at daily seasonality. We will use an additive version of STL, such that the sum of each component returned from the decomposition results in the original time series. A multiplicative version of STL demands non-zero values for each hour, which is not the case for our data. Figure 5.2, illustrates the conceptual framework. This experiment will be conducted in two parts. First, we will find the parameters for STL-decomposition best suited for our problem area. Then we will use those parameters for decomposition and combine them with the MLP. The flow presented for decomposition is only used for some of our time-series forecasting models and none of the distribution forecasting models.

We follow the expression for time series decomposition presented in Section 3.1.1, and use STL to decompose our time series to seasonal component s, trend component m and a residual component r. We fit linear regression to the trend component m and use machine

learning trained on the residual component r, to create forecasted values $\hat{m}$ and $\hat{r}$, respectively. The final forecasted value $\hat{y}$ is the sum of the forecasted values $\hat{m}$, and $\hat{r}$, and the extracted seasonal component s. Equation (3.1) formulates mathematically how a forecast at a given time is made.

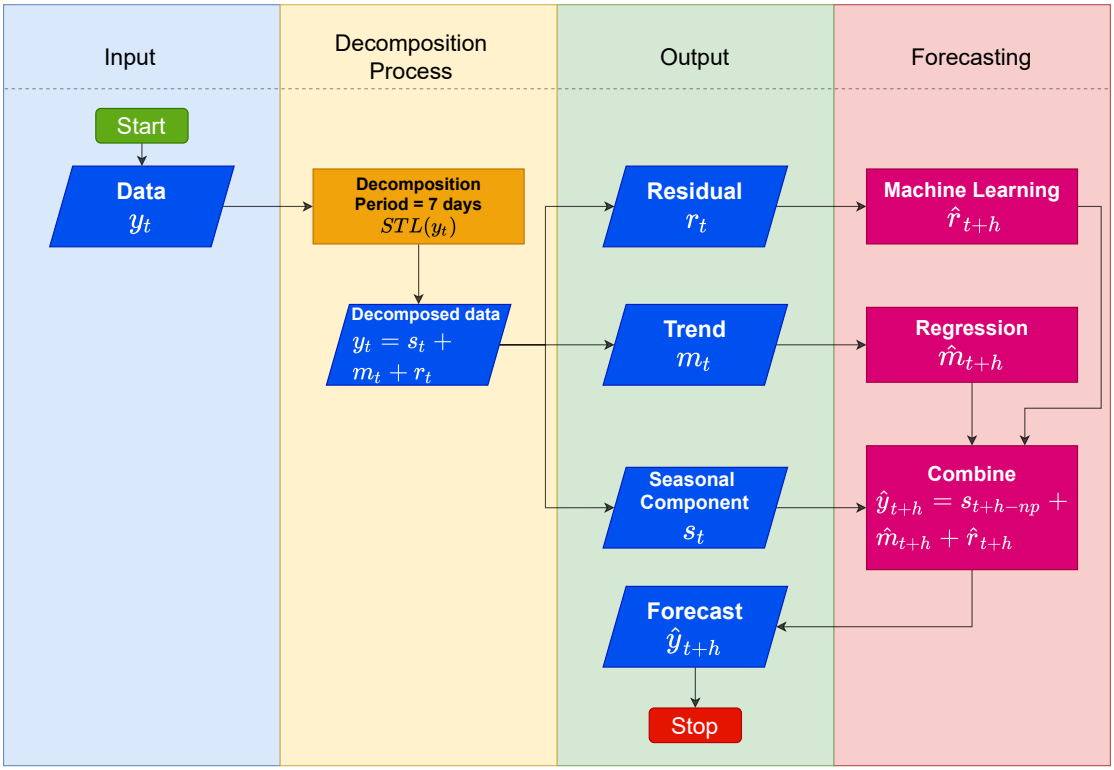

Figure 5.2: Framework for the combined model of STL and neural network. A mathematical formulation of the framework is presented in Section 3.1.1.

Since we split our data into training, validation, and test sets, it is essential only to use the training set during decomposition. We want to extract the seasonal component from the training set and then extend the component such that it extends into the validation and test set. We assume that the seasonal component repeats itself every week when using a weekly period. Therefore, we can use the test data to construct a weekly seasonal component and then extend this component by repeating it until it covers the entire data set.

When extracting the seasonal component from the data set, we simply subtract the value of the seasonal component for each timestamp. We can do this because the decomposition is based on an additive model, such that the sum of all components results in the original data set. Since we train our models on the data without the seasonality, seasonality must be added to the predictions before evaluating the models.

We can also use STL to extract a trend component from our time series. After finding a trend component from our training data, we can approximate and forecast the trend component using polynomial regression. We will test different degrees of polynomial regression going from the first degree up to the fifth. To avoid the issues with overfitting described in Figure 3.2, we prefer a lower degree of polynomial regression. Alternatively, we can decide not to remove the trend component and leave it to the neural network to learn this feature in our data set.

Our goal with decomposition is to extract as much information from the time series as possible before handing it over to the MLP model. When comparing the different STL parameters, we will use the same error metrics as we use for comparing models as the decomposition with the lowest error will also be the one that extracts the most information from our time series.

5.2.3 SSA-Decomposition and Neural Networks

Experiment goal *Explore if SSA-decomposition can be used to improve MLP model predictions*

Similarly to Section 5.2.2 our experiment consists of two parts. The first goal is to find SSA parameters best suited for the decomposition of our time series. The second is to see if SSA-decomposition can be used in combination with MLP to improve forecasts. Unlike STL, SSA does not require a predetermined period before decomposition, something that could be beneficial as we do not have to make assumptions about the length of the optimal period.

For SSA-decomposition, we use *Rssa* (Golyandina & Korobeynikov, 2014), which is an implementation of SSA written in the programming language R (R Core Team, 2021). SSA is a well defined statistical method for decomposition presented in Wu et al. (2010). To ensure that a correct implementation, according to the specification of SSA, we use this package to perform the decomposition. Since the rest of our code is written in Python (Van Rossum & Drake, 2009), we choose to use RPy2 (‘RPY2’, n.d.), which is a Python package that provides an interface to run embedded R in Python.

A key parameter in SSA is the grouping used for reconstruction. The grouping is used to decide which components from the deconstructed time series to be used for reconstruction. The *Rssa*-package provides two grouping methods using either a periodgram or a W-correlation matrix. When grouping using periodgram, one looks at the contribution provided by each component and includes components where the contribution to the reconstruction is greater than a specified threshold. The other option is using hierarchical clustering, according to algorithm 2.15 in *Singular Spectrum Analysis with R* (Golyandina et al., 2010), with the W-correlation matrix as a proximity matrix. In either case, we want the resulting decomposition

to consist of two reconstructed series, one containing the decomposed trend and seasonality, and one for the residual.

Similarly to the decomposition using STL, we only use test data for our decomposition. In order to extend the decomposition, to cover validation and test data, we can use the forecasting methods provided by Golyandina and Korobeynikov (2014). There are three available forecasting methods: bootstrap forecasting, recursive forecasting, and vector forecasting, all implemented according to the algorithms presented in chapter 2.3 in *Basic Singular Spectrum Analysis and forecasting with R* (Golyandina & Korobeynikov, 2014).

As we have not found literature supporting choosing one method for grouping or forecasting over the others, we will test all possible combinations and score them using error metrics introduced in Section 3.2.

To summarize, we will decompose our time series y into g and r such that $y = g + r$, where g is the grouping of components found in SSA-decomposition, representing seasonalities and trend. We use forecast methods provided Golyandina and Korobeynikov (2014) to create the forecast values $\hat{g}$. We train an MLP model on the residual component r to create predictions $\hat{r}$. We get our final predicted values $\hat{y}$ by summing these components, such that $\hat{y} = \hat{g} + \hat{r}$.

5.2.4 Weight Initialization using Genetic Algorithms

Experiment goal *Produce accurate volume forecasts by initializing MLP weights using genetic algorithms.*

As mentioned in Section 4.5, we want to implement an adapted version of the PNN proposed by Huang et al., 2019. We follow the authors' architecture and implement a neural network with a single hidden layer and gradient descent (GD) as the optimizing function. One of the core ideas of Huang et al., 2019 is that genetic algorithms can be used to set initial weights to reduce the chance of getting stuck in a local optima, which is often a problem when using GD as the optimizer. The fitness function used in PNN assumes that the predicted value follows a Poisson distribution. We will replace this fitness function with MSE, as our time series does not follow a Poisson distribution. To distinguish this model architecture from our other volume forecasting MLPs, we will call this MLP model MLP_{GD}, and the model described in Section 5.2.1 for MLP_{Adam}.

Huang et al., 2019 use an exponential activation function for every layer in their PNN architecture. Since our time series could have negative values for demand after decomposition, we will use a linear activation function for the output layer when decomposition is used, since an exponential activation function cannot produce negative values.

We will conduct an experiment to see if using our genetic algorithm for weight initialization improves the accuracy of our model, both for the version with exponential activation and linear activation on the output layer. This experiment will be conducted without decomposition and with the input parameters with the best scores from Section 5.2.1.

Initialization

We define our chromosome as a list containing all the weights of our machine learning model. We initialize models with random weights, which we use to create our initial population.

Fitness function

In the PNN-model presented by Huang et al. (2019), the log maximum likelihood function of a Poisson distribution is used to calculate the fitness of our chromosomes. We will replace this with MSE as we do not have a Poisson distribution when working with hourly resolution. To reduce computation costs during fitness calculation, we select a random sample of timestamps from our training data to be used to evaluate fitness. We create a model using the chromosome to set the model weights for each individual. We then use this model to predict ambulance demand for each time slot.

Algorithm 1 presents pseudocode for the fitness calculation. The algorithm takes in the chromosome as input c, a set of inputs x to use for model prediction sampled from our training data, and a set of outputs y corresponding to inputs x. The output of the algorithm is the fitness value of the individual, based on the predictions.

Algorithm 1: Evaluate chromosome fitness

input : Chromosome c, input values x, target values y

output: Fitness f

```
1 m ← CreateModel(c)        // Create model with weights using chromosome
2 ŷ ← m.predict(x)
3 f ← MSE(ŷ, y)
4 return f
```

Crossover

We use a simple two-point crossover when creating offspring. Before crossover is applied, the parents are weighted by fitness such that the best scoring parents have a higher probability of crossover with each other.

Mutation

Mutations are performed using Gaussian mutation. A Gaussian mutation is performed by adding a random value defined by a Gaussian distribution to each element of the individual's chromosome. We use a mean of zero and a standard deviation of 0.1 as parameters of our Gaussian distribution, as this corresponds well to our initial weights.

Selection

We select which chromosomes to keep after each generation, following the elitism methodology. Offspring and parents are evaluated equally, and we choose to keep only those with the best fitness.

After enough generations have passed, we set weights using the chromosome with the best fitness in the final evaluation. We then train our model using the entire training data set, optimizing weights with GD. Similarly to the other neural networks, we implement early stopping using the validation set.

5.2.5 Alternative Error Metric

Experiment goal *Produce forecasts with a stronger emphasis on the high-demand hours.*

Haugsbø Hermansen and Mengshoel (2021) suggests an alternative error metric for future work. The idea is to push a model to overestimate demand rather than underestimate it. This metric is formulated in Equation (5.1) and is similar to MSE with a double error when underestimating compared to the target. The goal is to provide better forecasts at the high-demand hours at the cost of overestimating the low-demand hours.

$$L(\hat{y}, y) = \begin{cases} 2(\hat{y} - y)^2 & , \hat{y} < y \\ (\hat{y} - y)^2 & , \hat{y} \geq y. \end{cases} \tag{5.1}$$

We are interested in applying this metric to explore how the forecasts relate to the target volume. This relation is also interesting from the perspective of OUH, as they might be more interested in forecasts that provide an upper bound to better plan the availability of resources. To perform this experiment, we use some of the approaches from previously introduced methods. Given that the gradient of the error metric is undefined for $\hat{y} = y$, we apply derivative-free optimization (DFO). We perform DFO with a GA, similar to Section 5.2.4, where the fitness function is the proposed loss function applied to a set of predictions made

by an individual. I.e., we are replacing MSE on line 3 of Algorithm 1 with $L(\hat{y}, y)$ from Equation (5.1). The neural network models are based on the MLP introduced earlier.

If the initial experiment with DFO gives results as expected, we also want to perform experiments with gradient descent after DFO, with MSE as the loss function. In most settings of initial weights, the model begins with moderate predictions and applies the gradient to approach the target from an underestimation. We are interested in exploring if the gradient descent behaves differently if the initial weights are set to overestimate the demand. To compare the results, we repeat the same procedure of DFO and GD with an MLP using MSE-loss in DFO instead.

5.3 Distribution Forecasting

Experiment goal *Produce high-resolution spatial forecasts for Oslo and Akershus.*

We want to accurately forecast incidents at the highest resolution for our data set. To accomplish this, we want to try two different models in two experiments. The first is a Wasserstein GAN, which will provide a generated scenario of the incident distribution for an arbitrary hour. The second is a more generic MLP, similar to the best-performing model of Haugsbø Hermansen and Mengshoel (2021) with some attempts at improvements.

5.3.1 Wasserstein Generative Adversarial Network

Experiment goal *Explore an alternative way of estimating the distance between two distributions through the use of an approximation to the Wasserstein distance.*

Data *Data set aggregated by cell ID and normalized for each hour of the data set. This results in a data set with a distribution of incidents per location per hour, which is then mapped to a 124 x 124 grid for training our WGAN.*

One goal of our methods with regards to distribution forecasting is to make use of the Wasserstein distance between a prediction and a target through the use of a WGAN. We assume incorporating some distance factor between two probability distributions will give us better forecasts. As discussed in Section 3.4, we aim to make use of a discriminator to approximate the Wasserstein distance of a generated distribution.

As the generator's input, we provide a 100-dimensional latent vector, sampled uniformly random from the interval $[-1, 1)$. The generator's output is a 124 x 124 2-dimensional vector

representing the distribution of incidents in Oslo and Akershus. The discriminator model is presented with the generator's output and real samples from our data set. The output of the discriminator is 1 for a real image and -1 for a fake image. Real and fake prediction from the discriminator is then multiplied to give an estimated distance between a real and a fake distribution, thereby approximating the Wasserstein distance. The Wasserstein distance approximation from the discriminator is used to update the generator, the discriminator, and the adversarial networks. The task of the adversarial is to present the discriminator with fake distributions but with real labels in an attempt to "trick" the discriminator.

5.3.2 Distributional Multilayered Perceptron

Experiment goal *Experiment with different input combinations in an attempt to understand how spatial distribution is related to different temporal information.*

Data *Data set aggregated by cell ID and normalized for each hour of the data set. This results in a data set with a distribution of incidents per location per hour. The input set is made up of different combinations of the hour of the day, day of the week, week, and year.*

Our implementation of an MLP is a relatively simple neural network consisting of three hidden layers and an output of size 5569, one value per cell in our data, as proposed in Haugsbø Hermansen and Mengshoel (2021). As for the model's input, we want to experiment with how different information about the selected period affects our predictions. We want to explore if our distributional data is affected by seasonal variations, given the seasonality of our temporal data. We want to test most of the different combinations of the data presented in Table 5.1. Due to the exponential growth of combinations to test for each additional input variable, we have chosen to limit our combinations to exclude month and day of the month. We assume the day of the week and week gives a better chance of extrapolating a possible weekly seasonality in the data. Our limit of combinations results in 23 possible combinations of input data for our model. The experiments are performed with k-fold cross-validation, with $k = 5$, and the error is measured with CCE. Early stopping is also applied to each fold, with a patience of 5.

5.4 Spatial Aggregation

Experiment goal *Explore how different spatial aggregations might affect our forecasts, and if it will improve the precision of forecasts compared to other, simpler models.*

Data *Data with information about grid cells in Oslo and Akershus, from SSB*[1]*. Full data set grouped by cell ID and each hour of the data set.*

The spatial data in our data set has a high sparsity when looking at hourly periods, and as argued in Setzler et al. (2009), this may lead to a zero-inflated distribution. We believe that our predictions will be better if we aggregate parts of the spatial regions into "buckets" of arbitrary size and shape, guided by a set of limits or objectives that we want to stay within or minimize. To solve this problem, we want to apply a multi-objective evolutionary algorithm (MOEA) to approach a set of Pareto-optimal spatial segmentations for our data set.

5.4.1 Model

Before applying genetic algorithms to our problem, we need to define a model to represent our problem. Since we have a set of cells in Oslo and Akershus with a unique ID and different information such as population, area, location, and the number of incidents, we want to use this data.

Given the complete set of 5569 cells, C, we define subsets $g_j \subset C$, with $c_i \in C$ and $c_i \in g_j$, resulting in m subsets. We then define an individual $G = \{g_1, g_2, ..., g_m\}$ such that $\forall_{g_a, g_b \in G, g_a \neq g_b} g_a \cap g_b = \emptyset$ and $\bigcup_{g \in G} g = C$.

In particular, we are working with a *Stirling number of the second kind*, i.e., the number of ways to partition a set of n elements into k non-empty subsets. In our case, we have an arbitrary k limited by a max number of subsets, m_{max}. A Stirling number of the second kind can be calculated as presented in Equation (5.2).

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n. \tag{5.2}$$

As an example, if we have $n = 100$ cells, and we want $k = 4$ non-empty subsets, we have that $\left\{ {100 \atop 4} \right\} = 6.7 \cdot 10^{58}$. If we further scale this to our problem space with $n = 5569$ cells and $k \in [1, m_{max}]$, the search space increases incredibly fast, which illustrates why a heuristic search algorithm will be useful.

We also want to have a concept of segments s_k, which is a subset of C, where $s_k \subseteq g$. We want all cells to be interconnected in the cardinal directions in a segment. I.e., in a segment with more than one cell, each cell should have a neighbor either north, east, west, or south within the same segment. Thus, a subset of cells might contain multiple segments, or it could

[1] https://kart.ssb.no/share/7e9bd10aca46

be a segment itself. Finding out if the cells within a subset are fully connected is performed using breadth-first search (BFS), visiting the cells within the same subset in each cardinal direction. If any cells within the subset have not been visited after termination, we recursively visit these cells and count the number of distinct segments within the subset. This conversion of subsets to segments is performed after the entire genetic algorithm is terminated, as this is a costly operation for each individual in each generation. When we have a complete set of segments containing all 5569 unique cells in total, we have a segmentation of Oslo and Akershus.

5.4.2 Objectives

Working with MOEAs requires a set of different objectives to optimize. These objectives are problem-specific and are subject to either minimizing or maximizing. In our case, we have suggested four objectives that we want to minimize to create a segmentation of Oslo and Akershus that might help reflect the coverage of ambulances and improve our predictions for a given segment.

Total Population Above Threshold

We have discovered from our data that there is a correlation between population and the number of incidents. Therefore, we are looking at ways to limit the maximum population in each subset and keep about the same population within each. Since the total population in the area we are segmenting is constant, we set an arbitrary limit to the population for a given subset and look to minimize the exceeding population for each segment in a segmentation. Our goal is for this limit to represent the average population per ambulance, such that each segment in the segmentation might illustrate the coverage of one ambulance. The population measure is based on population data in 2019 available from SSB. For each subset g, the objective score is given by $o_1(g)$, as defined in Equation (5.3), where p_{max} is the population limit and $c.population$ is the population in cell c.

$$o_1(g) = \max\{\sum_{c \in g} c.population - p_{\max}, 0\}. \tag{5.3}$$

Furthermore, we denote the overall objective for an individual G as $O_1(G)$, and it is defined mathematically in Equation (5.4).

$$O_1(G) = \sum_{g \in G} o_1(g). \tag{5.4}$$

Braun et al. (1990) found that the average number of population per required ambulance in the US in 1980 was 51 223, which forms the basis of our choice of p_{max} given an assumption of an increase in efficiency. Thus, we set $p_{max} = 60\ 000$ for our experiments.

Total Area Above Threshold

In the same manner as with population, there is a limited area in which one ambulance can cover within a given target response time. Again, the total area in Oslo and Akershus is constant, so we introduce a limit as to how large one subset should be. This limit is intended to capture the average coverage area for one ambulance, and we look to minimize the exceeding area for each subset. Given that we are working with 1×1km cells, the area calculation is simply the number of cells within one subset, given by the cardinality of a subset, $|g|$. For each subset g, the objective score is given by $o_2(g)$, as defined in Equation (5.5), where a_{max} is the population limit.

$$o_2(g) = \max\{|g| - a_{\max}, 0\}. \tag{5.5}$$

We denote this objective for an individual G as $O_2(G)$, which is defined in Equation (5.6).

$$O_2(G) = \sum_{g \in G} o_2(g). \tag{5.6}$$

Given an approximately circular shape of a segment, we want to make the distance from edge to edge traversable in approximately 10 minutes in an emergency. This is also combined with the assumption that high-population and high-traffic areas will not reach the area limit given the population restriction introduced above, making it possible for ambulances to drive faster to the incident. We, therefore, use a maximum area of 280 km^2 per segment for our experiments, which results in a radius of 9.44km given a circular shape.

Separate Segments

A subset where the cells are not interconnected makes little sense in the domain of ambulance response areas. For a segmentation to make sense, the cells within each subset need to be interconnected in one of the cardinal directions. We want to minimize the number of separate segments within the same subset to accomplish this. The number of segments within a subset g is given by $o_3(g)$, where we use BFS to find the number of segments. We denote the overall objective for an individual G as $O_3(G)$, and define it in Equation (5.7).

$$O_3(G) = \sum_{g \in G} o_3(g). \tag{5.7}$$

Total Circumference

For each subset, we want to have as regular shapes as possible, such that the travel time from the center to the outer areas is close to equal for each cell on the border of the subset. We want to approach a circular shape while still covering the whole area we are segmenting. We denote this objective for individual G as $O_4(G)$, and define it in Equation (5.8) where $o_4(g)$ is the circumference of subset g.

$$O_4(G) = \sum_{g \in G} o_4(g). \tag{5.8}$$

Overall Objective

Given the objectives above, the segmentation we believe will be most applicable to real life is where the densely populated areas have small segments to mostly maintain the population limit (O_1), while the rural areas will be larger but still mainly within the area limit (O_2). Furthermore, we believe larger and more regular-shaped areas are more feasible as long as they stay within limits. This is intended to be regulated by the number of segments (O_3) and total circumference (O_4). To accomplish this, we apply non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002).

5.4.3 Segmenting Non-Dominated Sorting Genetic Algorithm II

One of the main parts of NSGA-II is the use of the Fast Non-Dominated Sorting algorithm described in Deb et al. (2002), which assigns a rank and a crowding distance to each individual. The lowest rank is a subset of individuals whom all dominate those of higher ranks, i.e., have a better objective score for each objective. This is illustrated for two objectives in Figure 5.3. The crowding distance measures the average density of solutions, and a higher crowding distance indicates a more diverse solution within the rank. Our approach consists of the traditional steps of a genetic algorithm; initialize population, parent selection, crossover, mutation, and evaluation. Each of these steps is tailored to our problem and requires a short description of the methodology.

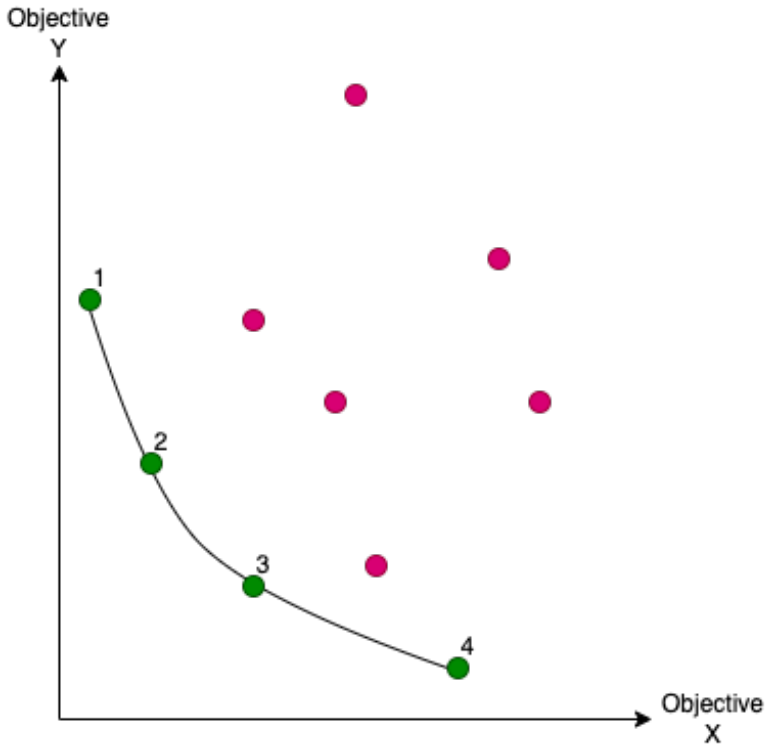

Figure 5.3: Illustration of the ranking in NSGA-II. Points marked in green belong to the lowest rank, as these dominates all purple points in objectives X and Y.

Initialization

We want to create a good initial population to reduce the number of iterations needed to reach a segmentation we are content with. Although, we do not want to enforce our limitations strictly, as this might result in an undesirable initial state and push our segmentation toward a local maximum instead of a global maximum. We calculate the average Manhattan distance between the cells of each existing subset and a new cell to be inserted and filter out those not within an initial distance factor $d_{threshold}$. The average distance from a cell c to a subset g is calculated as shown in Algorithm 2, where $c.x$ and $c.y$ denotes the x-coordinate and y-coordinate of a cell c, respectively. The initial distance calculation is intended to keep cells in the same subset close during the initialization.

We also filter out subsets where the population limit would be surpassed if we were to insert the cell there. If no subsets remain after this filtering process, we begin a new subset and add it to the individual as a candidate for the subsequent insertion. If a limit of m_{max} subsets is reached, we insert it randomly if no subset satisfies the above criteria. In a typical setting

Algorithm 2: Average Manhattan Distance

input : Cell c, subset g

output: The average Manhattan distance from cell c to all cells in subset g

$d \leftarrow 0$

forall c' *of* g **do**

  $d \leftarrow d + abs(c'.x - c.x) + abs(c'.y - c.y)$

end

return $d/g.length$

of parameters, m_{max} subsets should never be reached. The process of a feasible insertion is described in Algorithm 3, which is performed once for every cell in the data set. This procedure is repeated n times to create n individuals, shuffling the cells to avoid deterministic initialization.

Parent Selection

Parent selection is performed by deterministic binary tournament selection. We select two individuals at random, compare their rank, and select the lowest rank of the two. If the two selected individuals have the same rank, we choose the one with the highest crowding distance to maintain diversity. If both have equal crowding distance, we select randomly between the two. This process is repeated $\frac{n}{2}$ times to create a list of parents to be used for creating the next generation, where the same parent may appear multiple times.

Crossover

Crossover is performed by selecting random parents from the previous step and applying crossover with a probability p_c. With a probability $1 - p_c$, we allow a copy of the parents to persist as offspring. Crossover is done by a modified two-point crossover where we represent the gene as a list of ordered cells based on the respective subset ordering. Cells included in the list before the cutpoint are removed from the opposite individual, then re-inserted in a randomly chosen subset from the neighboring subsets of each cell. If no neighboring subsets exist, we insert feasibly as illustrated in Algorithm 3. The crossover-process is illustrated in Figure 5.4, where parents and offspring are represented as individuals G_k with subsets g_i^k, where m_k is the number of subsets for individual k.

Algorithm 3: Feasible Insertion

input : Set G of subsets g_i, cell c to be inserted
output: A new set G'

$G' \leftarrow$ copy of G
$P \leftarrow \{\}$ `// Possible subsets for insertion`
forall g *of* G' **do**
  $d \leftarrow$ `AverageManhattanDistance` (c, g) `// Algorithm 2`
  if $d < d_{threshold}$ **then**
    $g.d \leftarrow d$
    add g to P
  end
end
sort P ascending by $g.d$
forall g *of* P **do** `// Iterate from lowest average distance to highest`
  if $g.population + c.population \leq p_{max}$ **then**
    $g' \leftarrow$ copy of g
    add c to g'
    exchange g with g' in G'
    return G' `// Insertion performed, exit`
  end
end
if $G'.length \geq m_{max}$ **then** `// Max subsets reached, insert randomly`
  $g' \leftarrow$ select random g from G'
  add c to g'
  exchange g with g' in G'
  return G'
end
$g' \leftarrow \{c\}$ `// No feasible subset found, create new subset`
add g' to G'
return G'

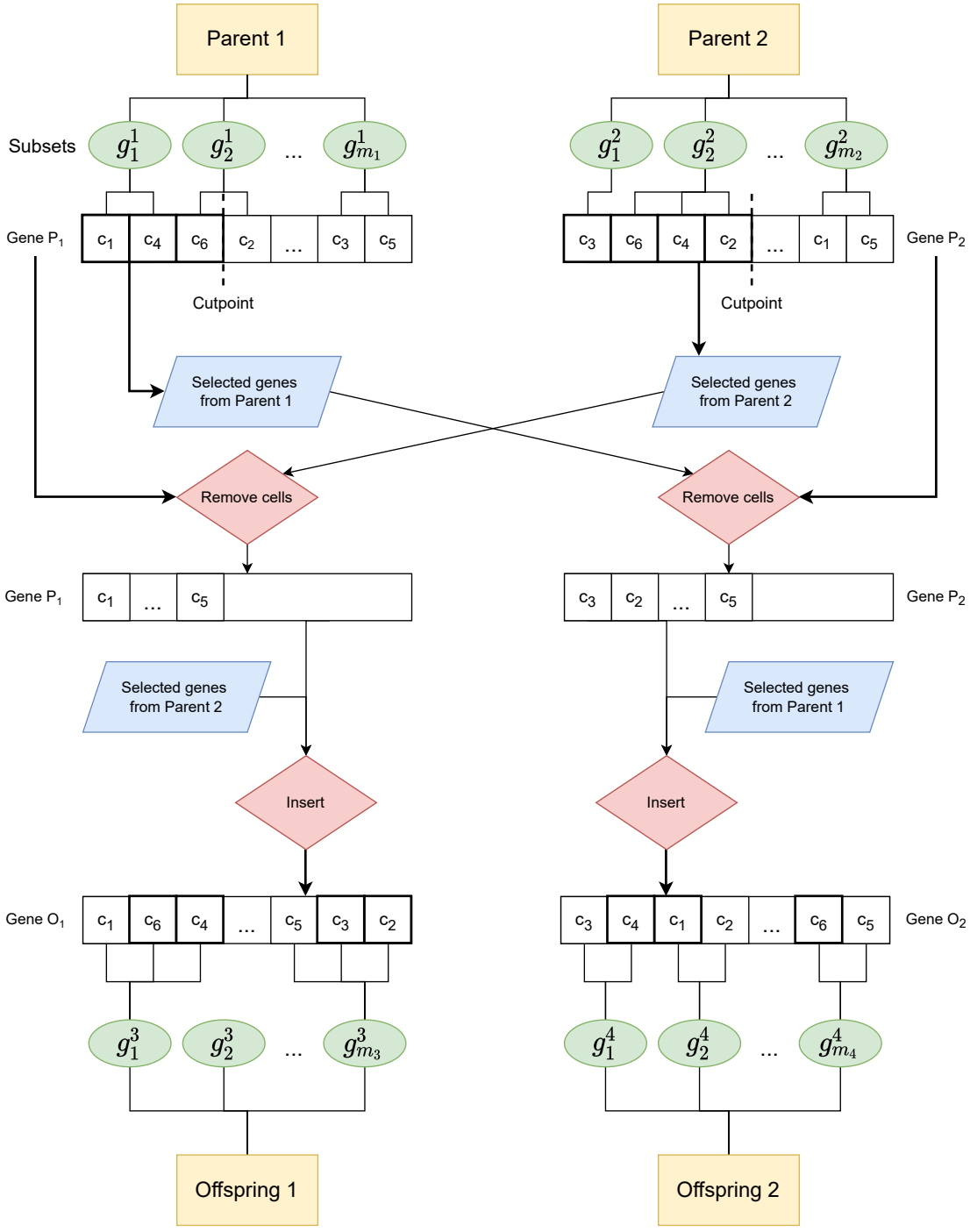

Figure 5.4: Crossover as performed in our MOEA-implementation. Cells are partitioned into subsets. Each parent has a certain number of subsets.

Mutation

We apply four different mutation functions. Two of them are used to reduce the number of subsets, and the other two are intended to increase the number of subsets. For each offspring, there is a probability of p_m that it should undergo mutation. Mutation in an individual will, at most, affect two subsets at a time. Reduction, split, and fraction mutation employ a random selection of a subset weighted by the respective objective's normalized values, this is shown by Equation (5.9) where $o_1(g_i), o_2(g_i), ...$ indicates the different objectives for subset g_i. This provides a probability P_i for selecting subset g_i for mutation, which results in a heuristic that more often chooses subsets where one or more objectives are higher than in the average subset.

$$P_i = \frac{\sum_{j=1}^{4} \frac{o_j(g_i)}{O_j(G)}}{4}. \tag{5.9}$$

The four different mutation functions we apply are as follows:

Reduction mutation *A single subset is selected at random, weighted by objectives (Equation* (5.9)*). The subset is emptied for cells. The cells previously contained within this subset are re-inserted feasibly, following Algorithm 3.*

Merge mutation *Two subsets are chosen at complete random, and the cells contained within each are joined together to create a new subset replacing the two subsets chosen.*

Split mutation *A single subset is selected at random, weighted by objectives (Equation* (5.9)*). A random cutpoint is selected. The cells in the selected subset are split into two new segments decided by the cutpoint, replacing the single selected subset.*

Fraction mutation *A single subset is selected at random, weighted by objectives (Equation* (5.9)*). The cells in this segment are iterated and assigned to a new subset depending on random chance. We begin with a single empty subset, and for each cell, with a probability of p_f, it is assigned to the current subset. Otherwise, a new subset is created. This mutation is presented in Algorithm 4.*

Algorithm 4: Fraction mutation

input : Set G of subsets g_i
output: A new set G'

```
G' ← copy of G
g ← SelectWeightedRandomSubset(G')                    // Equation (5.9)
remove g from G'
g' ← {}                                               // Assign empty set
forall c of g do
    r ← random decimal between 0 and 1
    if r < p_f then
        add c to g'
    end
    else
        add copy of g' to G'
        g' ← {c}          // Overwrite g', copy of previous g' still in G'
    end
end
add copy of g' to G'
return G'
```

Evaluation

The evaluation of our individuals is performed with non-dominated sorting as proposed for NSGA-II (Deb et al., 2002) and applying elitist selection based on the Pareto fronts of our population. Suppose an entire Pareto front does not fit within the given population size. In that case, the individuals are sorted by crowding distance, and the ones with the highest crowding distance are selected for the next generation. An example of how this is performed is illustrated in Figure 5.5.

Post-Processing

As discussed in Section 5.4.2, we want to minimize the number of separate segments, especially within the same subset. When our genetic algorithm finishes after g generations, we want to ensure that each separate segment is assigned to an actual subset as defined by our model to maintain the locality of our predictions. One problem we consider this to solve is a situation where two segments on each side of the map are considered to be in

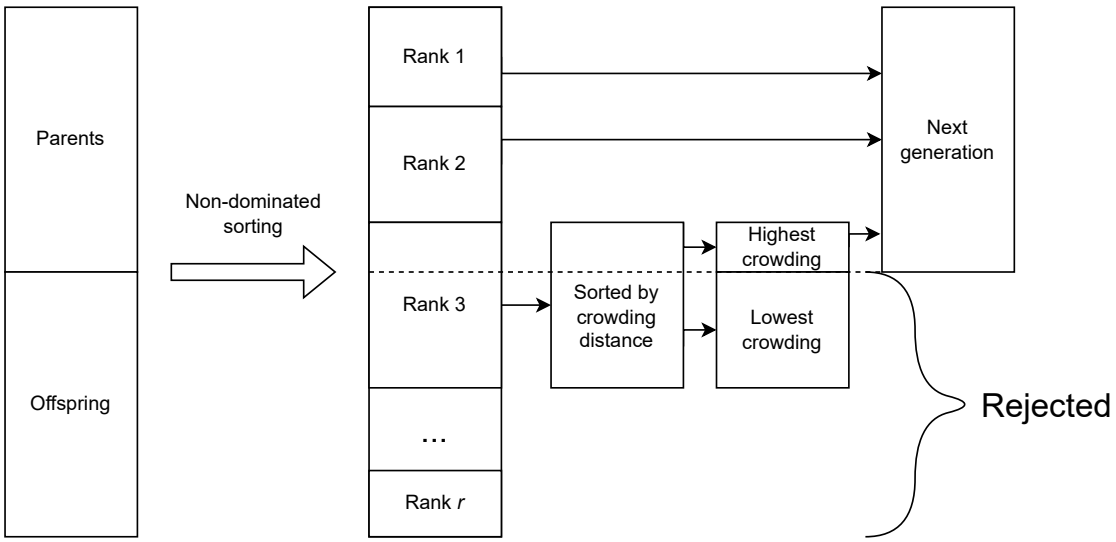

Figure 5.5: Selection phase of NSGA-II, performed with an elitist approach based on the Pareto rank and crowding distance of each individual. We apply the same method in our segmenting NSGA-II.

the same subset, and the same predictions will apply for both of these segments. This is not optimal in a realistic setting, as one would like to know which part of the map an incident happens. Having a 50/50 chance of an incident being on opposite sides of the map is highly problematic. When all our subsets have been converted to segments, we have a complete segmentation of Oslo and Akershus. This process is illustrated in Figure 5.6, where we go from two subsets, marked in red and green, to five segments. Each of the five segments is assigned to a subset to conform with our model, resulting in a complete segmentation of the area.

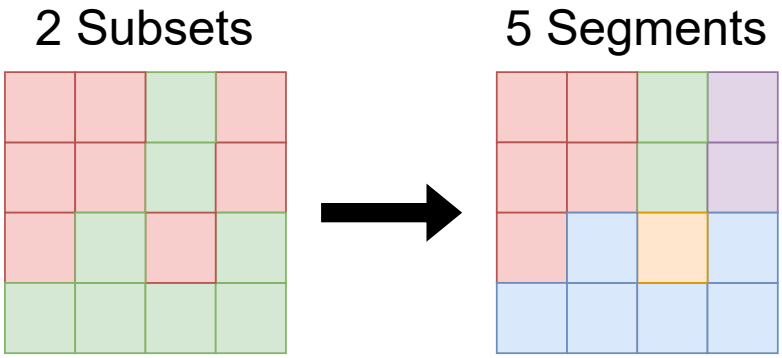

Figure 5.6: An example of how a set G of subsets g_i is converted to a set of segments in post-processing. Note that only cardinal neighbors can belong to the same segment.

The full segmenting NSGA-II algorithm is presented in Algorithm 5, from initialization to termination. The different probability parameters in the algorithm are set beforehand.

Predictions

For our spatial aggregation, we produce multiple segmentations, pick out superior ones in each objective, and some segmentations somewhere in between, which we believe are good all-around segmentations. We then sum the incidents per hour for each segment to create a new data set with different areas than before. Although we do not have strictly categorical data, we can still evaluate and compare the performance of different models using CCE. The output of our models corresponds to the number of areas in the segmentation we are looking at, and we use *softmax* on the output layer to create a distribution where the sum is 1. We can apply the predicted distribution to a predicted volume by simple multiplication to get a forecast for a given hour in a given area. We compare our models to the historical average with a k-fold cross-validation where we compute the average of four folds to score on the fifth fold. We choose this deterministic forecasting method as our baseline since we can not compare the score of two different segmentations directly, given the difference in the number of segments to forecast.

Algorithm 5: Segmenting NSGA-II

input : Set C of cells c, number of generations g, population size S
parameters: Probability of crossover p_c, probability of mutation p_m, probability of respective mutation functions p_{sm}, p_{rm}, and p_{mm}
output : A set of solutions I with rank

$I \leftarrow \{\}$ // Empty set of individuals
for $n = 0$ ***up to*** S **do** // Initialize S individuals
  $G \leftarrow \{\}$
  forall c *of* C **do**
    $G \leftarrow$ `FeasibleInsertion`(G, c) // Algorithm 3
  end
  add G to I
end
$R \leftarrow$ `RankPopulation`(I) // Fast non-dominated sort (Deb et al., 2002)
for $m = 0$ ***up to*** g **do** // Main loop
  $P \leftarrow$ `ParentSelection`(R) // Deterministic binary tournament
  $O \leftarrow \{\}$
  while $O.length < S$ **do** // Create S offspring
    $r_c \leftarrow$ random number between 0 and 1
    $i_1, i_2 \leftarrow$ random individuals from P
    if $r_c < p_c$ **then**
      $o_1, o_2 \leftarrow$ `Crossover`(i_1, i_2) // Figure 5.4
      add o_1, o_2 to O
    else add copy of i_1, i_2 to O
  end
  forall o *of* O **do**
    $r_m \leftarrow$ random number between 0 and 1
    if $r_m < p_m$ **then**
      $r_M \leftarrow$ random number between 0 and 1
      if $r_M < p_{rm}$ **then** `ReductionMutation`(o)
      else if $r_M < p_{rm} + p_{sm}$ **then** `SplitMutation`(o)
      else if $r_M < p_{rm} + p_{sm} + p_{mm}$ **then** `MergeMutation`(o)
      else `FractionMutation`(o) // Algorithm 4
    end
  end
  $I \leftarrow$ merge I and O
  $R \leftarrow$ `RankPopulation`(I)
  $I \leftarrow$ `NewPopulationFromRank`(R) // Figure 5.5
end
forall G *of* I **do** `SplitSubsetsToSegments`(G) // Figure 5.6
return I

Chapter 6

Experimental Results

In this chapter, we present the results and findings of our method and compare these to a set of different baselines. We also compare our results to those of Haugsbø Hermansen and Mengshoel (2021) to assess whether we can improve on the methods presented there.

6.1 Experimental Setup

Before presenting our experiments and results, we provide the setup and frameworks used to perform the experiments. First off, all experiments are performed with one of the two setups presented in Table 6.1.

Table 6.1: Machines used for the setup of experiments.

Setup	Operating System	Processor	RAM	Python version
Van De Weijer	MacOS Big Sur	Intel Core i7	16GB	3.9.7
Owren	MacOS Monterey	Apple M1	16GB	3.9.10

Rather than implementing neural networks from scratch, we use the Python library Keras (Chollet et al., 2015). This allows for simple architecture implementation, parameter tuning, and training of our models, allowing us to spend our time focusing on other parts of our research. We use the *statsmodels*-library for Python (Seabold & Perktold, 2010) to perform STL-decomposition.

6.2 Data Analysis and Preprocessing

Chapter 2 presents most of the results from our pilot project, but some of the resulting data from STL-decomposition is still interesting to present. As discussed in Section 5.1, we ended up decomposing our data at periods of 168 hours for the seasonal component and 365 days for the trend component. The seasonal component from the decomposition is presented in Figure 6.1 and the trend component is presented in Figure 6.2.

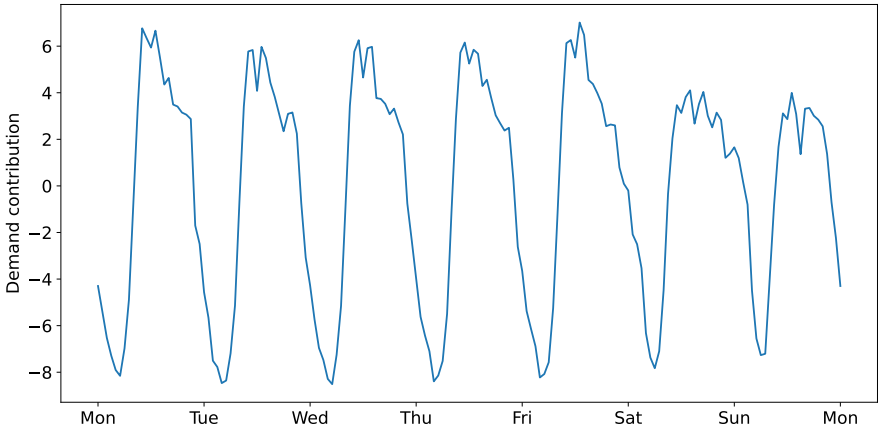

Figure 6.1: Seasonal component of STL-decomposition with a period of 168 hours.

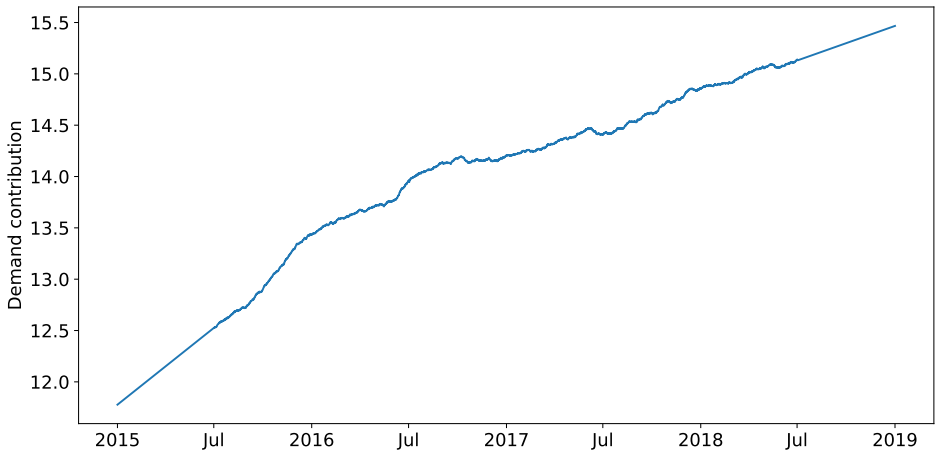

Figure 6.2: Trend component of STL-decomposition with a period of 365 days.

From Figure 6.2, we can see that the trend component is straightened out at the beginning, from January 2015 to July 2015, and at the end, from July 2018 to January 2019. This is because LOESS interpolation is a generalization of moving average and polynomial regression,

which is used to smooth the data after the seasonal component is removed until the trend remains. This means that the first and last $\frac{p}{2}$ observations are not part of the trend, where p is the period used for the decomposition. In this case $p = 365 \cdot 24$, i.e., a full year. The *statsmodels*-library includes methods to interpolate the trend based on a number of previous observations, which we use to fit our models to the trend. This interpolation is the straight areas at the beginning and end of the trend component.

6.3 Volume Forecasting

This section covers the results of the volume forecast experiments presented in Section 5.2. We use the first experimental results to decide on parameters for our models. The final results show the forecasting accuracy of our proposed models, constructed using information gained from our smaller experiments, compared to industry baselines.

6.3.1 Input Data Selection Results

We decide which combination of data inputs to use for training our models in our first experiment. Using a set consisting of the hour, day of the week, and the month was the highest performing in the research of Haugsbø Hermansen and Mengshoel (2021). We will refer to this combination of inputs as the *Basic* input set. Since we found an increasing trend in Section 5.1, we see if including year, both as a numerical value and one-hot encoding, will improve forecasts. Inspired by Haugsbø Hermansen and Mengshoel (2021), we use an architecture for our neural network consisting of two hidden layers with 16 nodes in each layer, using the *Adam* optimizer. We will also use the same architecture for all input combinations for consistency. We will refer to this architecture as MLP_{Adam}. From Table 6.2, we see that including the year as a numerical value improves our model performance. We will use the year encoded as a numeric value rather than one-hot encoding for our models.

Table 6.2: Validation error for the same neural network architecture, but with different input parameters without decomposition.

Data input sets	Average MSE
Hour, day of the week, month	22.6341
Hour, day of the week, month, year as numeric value	**21.7295**
Hour, day of the week, month, year as a one-hot encoding	24.6047

6.3.2 STL and SSA

Our main objective with decomposition with STL and SSA is to extract as much noise as possible from our time series, such that there are less features to handle for machine learning. This is essentially the same as achieving the lowest error using our metrics. The results from Table 6.3 show that SSA with w-correlation matrix and recurrent forecasting was the best performing parameters for SSA-decomposition.

Table 6.3: SSA parameter selection results.

Grouping method	Forecasting method	MSE
W-correlation	Bootstrap	25.3318
W-correlation	Vector	26.2989
W-correlation	Recurrent	**24.6047**
Periodgram	Bootstrap	25.5018
Periodgram	Vector	26.5796
Periodgram	Recurrent	25.3203

Results from Table 6.4 show that using a weekly period for seasonality scores better than a daily period. These results concur with our data analysis, where we found that the ambulance demand pattern is different for the days of the week.

Table 6.4: STL and polynomial regression parameter selection results.

Polynomial degree	Seasonal period	MSE
1	Daily	25.6604
2	Daily	25.0769
3	Daily	25.2445
4	Daily	25.9296
5	Daily	25.1184
1	Weekly	24.1701
2	Weekly	**23.5851**
3	Weekly	23.7534
4	Weekly	24.4396
5	Weekly	23.6269

Although using second-degree polynomial regression scores best on our validation data, we believe that it might not be best suited for prediction on our test data. Figure 6.3 shows the predicted trends from our experiments. As we can see from the figure, towards the end of the time series, both the second and fifth-degree versions have a negative gradient.

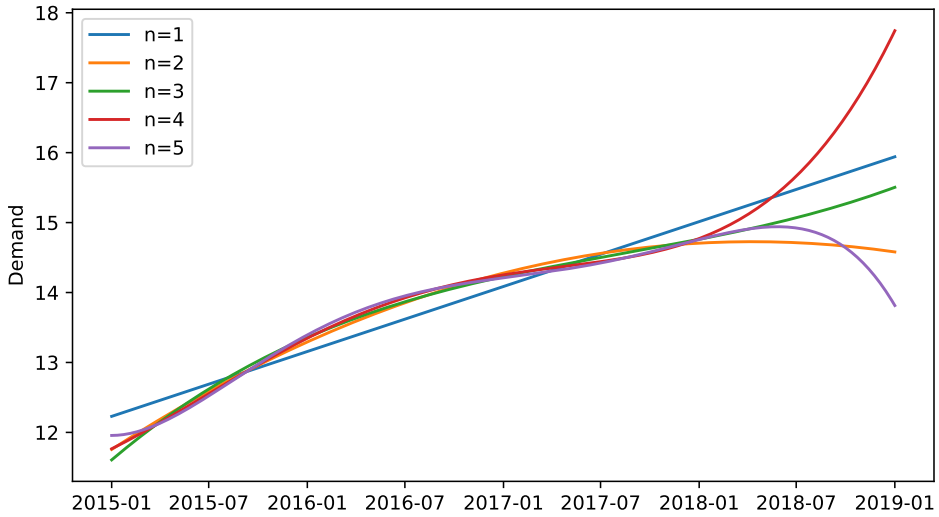

Figure 6.3: Predicted trend using different degrees of polynomial regression.

Figure 6.4 shows the negative gradient of $n = 2$ and $n = 5$ more clearly, as well as the trend with an interpolated period. We know from both discussions with OUH and our analysis that the trend is increasing annually. On this basis, we will also keep the best scoring STL-decomposition with an increasing trend towards the end of our time series. From Table 6.4, we can see that this is using a weekly seasonality and third-degree polynomial regression for predicting the trend. Considering how the predicted trend increases or decreases towards the end of our time series could be especially decisive for this study compared to a real-world application. This is because, in this study, the final evaluation is based on forecasting results several months ahead of time. It would probably suffice to predict a few days or weeks ahead of time in a real-world application. We will refer to STL-decomposition using a second-degree polynomial to predict trend as STL_2, and STL_3 when using a third-degree polynomial.

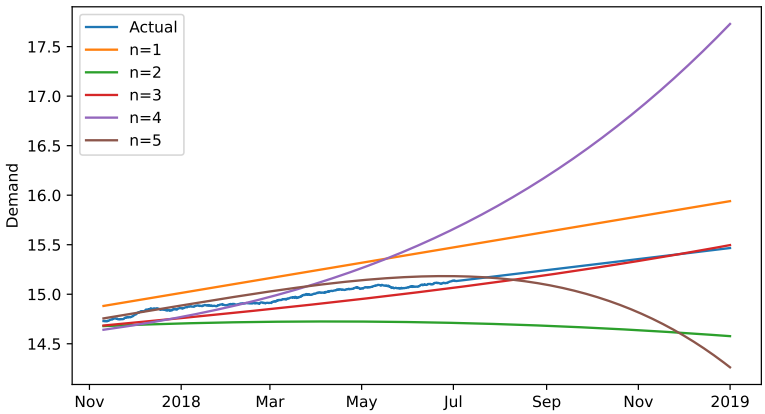

Figure 6.4: Predicted trend for the last 10 000 hours using different degrees of polynomial regression.

Figure 6.5 and Figure 6.6 shows the forecasted individual components, with the original residual, for experiments using STL and SSA-decomposition, respectively. Comparing these, we can see that the residual component r closely resembles each other. Where as Figure 6.5 has a predicted value for each of trend and seasonality, we can see that the grouped component in Figure 6.6 predicts one value for both. We can see that if we were to sum the trend and seasonal component in Figure 6.5, we would get a component relatively similar to the grouped component in Figure 6.6.

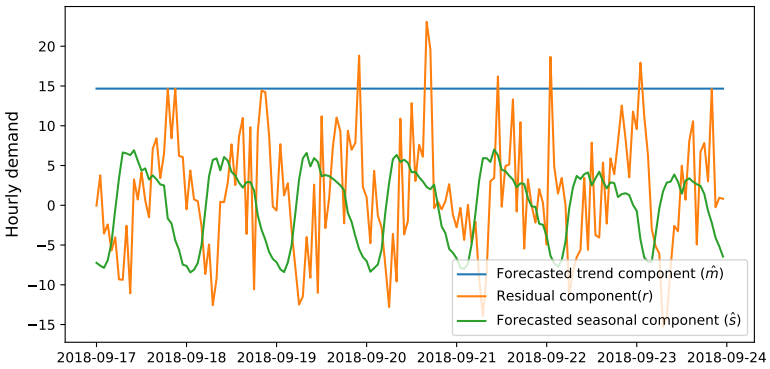

Figure 6.5: STL components from decomposition. Trend is forecast with polynomial regression, seasonality with a naïve forecast based on period.

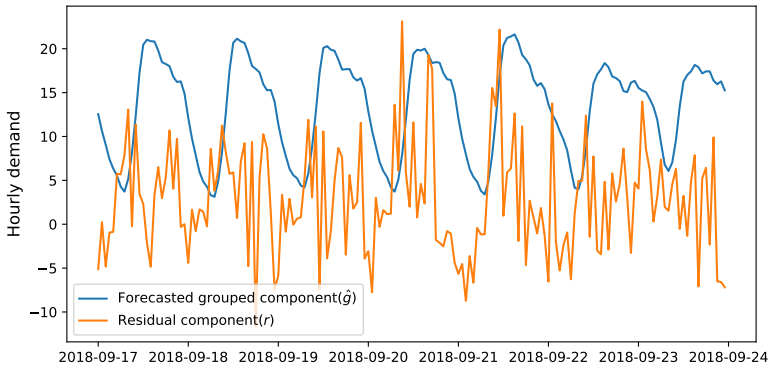

Figure 6.6: SSA components from decomposition. Grouped component is forecast with built-in methods for forecasting in SSA.

6.3.3 Weight Initialization with Genetic Algorithms

This experiment shows the result of our model adaptation of the PNN architecture proposed by Huang et al. (2019). The experiment aims to see if there are indications of whether we can use weight initialization with genetic algorithms for model improvement. Additionally, we want to see how using a linear activation function for the output layer measures up against exponential activation, as proposed in the original architecture by Huang et al. (2019).

The results from this experiment are presented in Table 6.5. Both models show improvement when genetic algorithms are used to initialize weights before model training. The results indicate that exponential activation for the final layer yields better forecasts. Based on these results, we will, in all cases, use genetic algorithms for weight initialization for this model. We will continue to use exponential activation when data is not decomposed. We will refer to this architecture as MLP_{GD+l} when a linear activation function is used, and MLP_{GD+e} when an exponential activation function is used.

Table 6.5: Weight initialization with genetic algorithm results.

Activation function	Weight initialization method	MSE
Exponential	Random	21.9757
Exponential	Genetic algorithm	**21.2325**
Linear	Random	21.8845
Linear	Genetic algorithm	21.5401

6.3.4 Test Results

This section covers the final test results from our volume predictions. We include baselines used in other similar studies covered in Chapter 4 to show how our models compare to those proposed by others. Table 6.6 presents the final hourly volume forecasting results. Note that these results are from the test set presented in Figure 5.1, while previously presented results are from the average of each fold in k-fold cross-validation on the training set. The model with the highest forecasting accuracy using MSE is the PNN inspired architecture, MLP_{GD+l}, with a linear activation function, *Basic* data set with year, and STL with a third-degree polynomial regression to approximate the trend. MLP_{Adam}, the MLP architecture with *Adam* optimizer, using STL-decomposition with a second degree polynomial to estimate the trend is the model that gave the best predictions using MAE as the error metric.

In Figure 6.7, we present a set of predictions from our best performing models in MSE and MSE on a week from our test set. We can see that both models follow each other closely, with the MAE model usually predicting a lower demand than the MSE model.

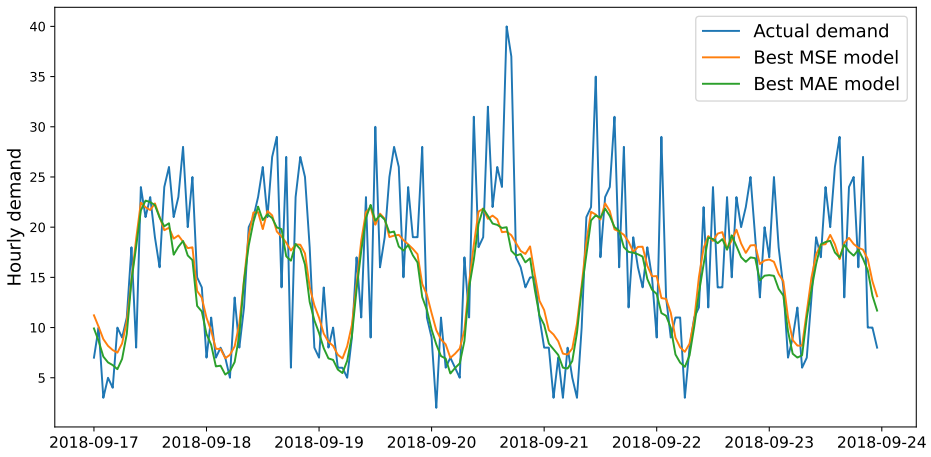

Figure 6.7: Forecasts of the best models using MSE and MAE for a week in September, compared to the actual demand that week.

One key takeaway from these results is the importance of capturing the trend in our time series. We deduce the importance of the trend by examining the only two models where the *Basic* input set was used, which does not include year, and there was no decomposition done. Both these models gave the two worst scores. Comparing decomposed *Basic* input sets with non-decomposed data with the year included further underlines this importance, as the

difference between the models where the year was included and not is reduced.

The results from decomposition indicate that using STL improves the forecasts of our MLP-models, as well as outperforming decomposition using SSA. SSA improved the accuracy of forecasts from MLP when year was not included in the input data, but produced similar results as not using decomposition at all when year was included.

Table 6.6: Comparison of different decompositions and models with baselines. The best results are in bold.

Input set	Decomposition method	Model	MSE	MAE
	None	SMA, $h = 6$	46.1227	5.4478
	None	Naïve forecast, $h = 168$	42.3716	5.0187
	None	Medic	26.3334	3.9072
Basic	None	MLP_{Adam}[1]	22.9464	3.6915
Basic	STL_2	MLP_{Adam}	22.0298	3.6297
Basic	STL_3	MLP_{Adam}	21.7391	3.6446
Basic	SSA	MLP_{Adam}	21.9294	3.6497
Basic	None	MLP_{GD+e}	25.3918	3.8465
Basic	STL_2	MLP_{GD+l}	21.9417	3.6289
Basic	STL_3	MLP_{GD+l}	21.7365	3.6456
Basic	SSA	MLP_{GD+l}	21.8490	3.6427
Basic + year	None	MLP_{Adam}	21.7709	3.6209
Basic + year	STL_2	MLP_{Adam}	21.7661	**3.6130**
Basic + year	STL_3	MLP_{Adam}	21.7315	3.6429
Basic + year	SSA	MLP_{Adam}	21.9193	3.6438
Basic + year	None	MLP_{GD+e}	21.9549	3.6561
Basic + year	STL_2	MLP_{GD+l}	21.9238	3.6248
Basic + year	STL_3	MLP_{GD+l}	**21.6835**	3.6413
Basic + year	SSA	MLP_{GD+l}	21.9820	3.6529

6.3.5 Alternative Error Metric

In this experiment we apply two MLP_{GD+e} models with weight initialization using GA and no decomposition, with the *Basic + Year* data set. We measure the error in two ways, with

[1]Best performing model from Haugsbø Hermansen and Mengshoel (2021)

standard MSE and *modified MSE* as presented in Equation (5.1). We define these models as such:

MSE-model *MSE as fitness function in DFO and as loss function in SGD.*

Alternative model *Modified MSE as fitness function in DFO, MSE as loss function in SGD.*

The results with different optimization techniques are presented in Table 6.7a and Table 6.7b, measured with MSE and modified MSE, respectively.

Table 6.7: Error metrics for alternative loss and MSE.

(a) MSE.

Model	DFO only	DFO + SGD (MSE loss)
MSE-model	35.2042	22.4471
Alternative model	40.5848	22.9199

(b) Modified MSE from Equation (5.1).

Model	DFO only	DFO + SGD (MSE loss)
MSE-model	56.9318	33.9856
Alternative model	53.2524	35.6131

From the results in Table 6.7a and Table 6.7b, we see that the modified MSE metric consistently yields a higher error. We also see that the alternative model results in a lower modified MSE error after DFO only, where as the MSE-model outperforms after SGD. For MSE error, we have that the MSE-model outperforms in both phases of training.

Although the quantifiable results give some insight into how the models perform, we also want to visualize and compare forecasts with a target to evaluate how the application of a higher loss for underestimation affects the forecast. In Figure 6.8, we present forecasts for the week of September 17th to 24th, 2018, from both the MSE model and the alternative model, compared to the target after only performing DFO. Similarly, Figure 6.9 presents the forecasts for the same week after applying SGD with MSE-loss to the models previously trained with DFO.

Looking at the forecasts after only performing DFO, we see a clear tendency of the model trained using the proposed error metric consistently forecasting higher demand than the model with MSE as the error metric. Further, applying SGD to these two models produces forecasts that are more aligned with each other, which is not that surprising given that we perform the gradient descent step with the same loss for both models.

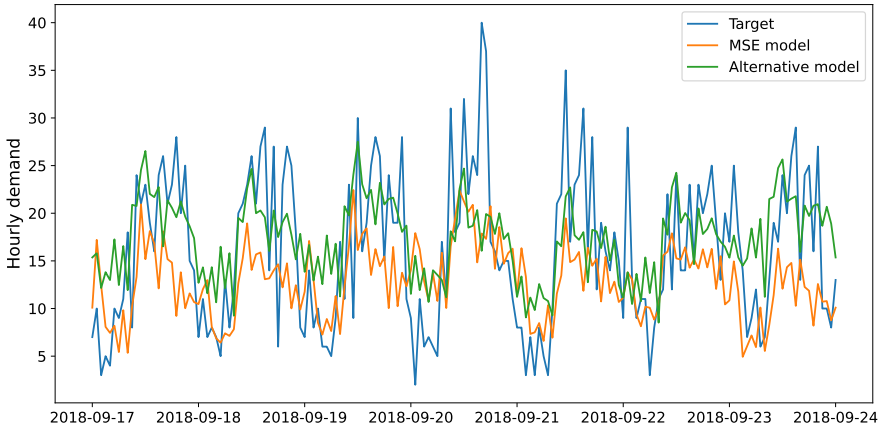

Figure 6.8: Forecasts from MSE model and alternative model after DFO compared to target for the week of September 17th to 24th 2018.

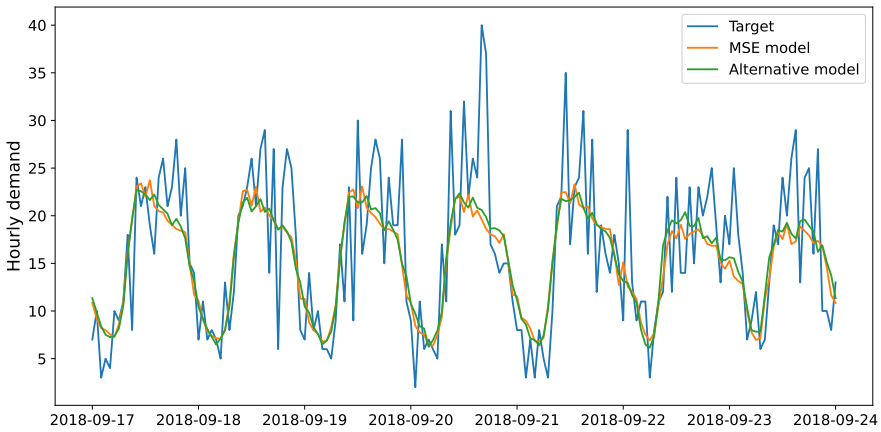

Figure 6.9: Forecasts from MSE model and alternative model after DFO and SGD compared to target for the week of September 17th to 24th 2018.

6.4 Distribution Forecasting

This section presents the results from our two different approaches to forecasting at the highest spatial resolution of our data set.

6.4.1 Wasserstein Generative Adversarial Network

The first experiment for WGAN consists of creating a model able to perform well on the MNIST data set introduced by (Lecun et al., 1998) before adapting it to our problem. Our implementation of WGAN for the MNIST data set is inspired by Atienza (2020). After getting

some good results from our experiments with MNIST, thereby verifying our implementation, we adapt the model to work with our data. To simplify the model, we aimed to maintain a quadratic resolution of our data. We also want to be able to resize by a factor of 4 in the process of generation. With the area of Oslo and Akershus having a maximum height of 122 cells and a maximum width of 83 cells, we ended up with data consisting of 124 x 124 cells. We normalize the number of incidents of each cell to be between 0 and 1, summing to 1.

The implementation with the MNIST data set performed adequately at generating realistic samples after about 5 000 iterations. We then modified the model to work with the higher resolution of our data and repeated the process. An important factor is an increase in the number of weights to train in the model, as the generator and discriminator have a lot more nodes within the layers of the network. The increase in resolution does not only make the weight update slower, but the model also needs to run for more iterations. In total, this makes the complete training process more tedious than for the MNIST data set, limiting how many iterations we could run.

We present four samples from our actual distribution in Figure 6.10, and we can see how sparse the data is when scaled to 124×124. Figure 6.11 shows 16 generated distributions from WGAN, and even though we can see some tendency of the locality around the high-demand areas, the general distribution area is too large to be of practical use.

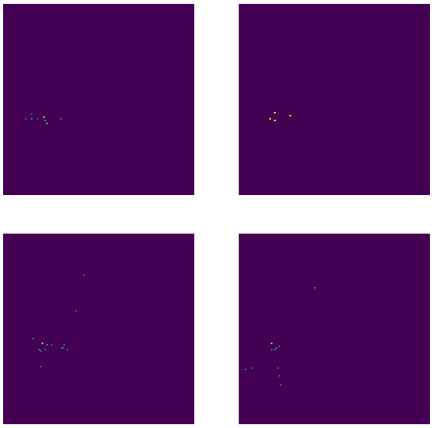

Figure 6.10: Four examples of true distributions after preprocessing in the format needed for WGAN.

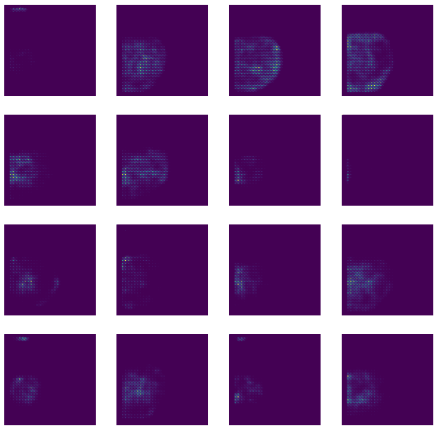

Figure 6.11: 16 generated scenarios from WGAN.

6.4.2 Distributional Multi-Layer Perceptron

For our distributional MLP, we build upon the findings of Haugsbø Hermansen and Mengshoel (2021) and look at how the input to the network affects the precision of our predictions. We compare the best input set of the model with the historical average and a modified all 0s, which forecasts an equal distribution for each cell as CCE is undefined for $\hat{y}_i = 0$.

As we can see from Table 6.8, hour, week and numeric year within the interval $[0, 1]$ gives us the lowest CCE. Generally speaking, we see that all inputs containing the hour outperform those without the hour.

Table 6.8: Average CCE for each combination of input variables.

Single	Avg. CCE	Double	Avg. CCE	Triple	Avg. CCE	Quadruple	Avg. CCE
H	5.8287	H,D	5.8293	H,D,W	5.8289	H,D,W,Y+O	5.8298
						H,D,W,Y+N	5.8282
				H,D,Y+O	5.8306		
				H,D,Y+N	8.8284		
		H,W	5.8295	H,W,Y+O	5.8315		
				H,W,Y+N	**5.8280**		
		H,Y+O	5.8283				
		H,Y+N	5.8300				
D	5.8362	D,W	5.8361	D,W,Y+O	5.8343		
				D,Y+N	5.8355		
		D,Y+O	5.8350				
		D,Y+N	5.8358				
W	5.8359	W,Y+O	5.8355				
		W,Y+N	5.8352				
Y+O	5.8350						
Y+N	5.8332						

Table 6.9: Average CCE for our best performing input, compared to historic average and all 0s.

Model	Average CCE
MLP (H,W,Y+N)	5.8280
Historical average	**5.8263**
All 0s	8.6250

We compare our best performing model to the historical average and modified all 0s. The results are presented in Table 6.9. The model with the best performing input set is unable to

outperform the historical average of the data. We also see that there is a maximum 0.0082 difference of CCE between the best (H,W,Y+N) and the worst (D) input variables for the MLP-model. Although there is a difference between the different input variables, it is not highly significant.

6.5 Spatial Aggregation

We run our segmenting NSGA-II for 200 generations, with a population size of 40. After the algorithm finishes, we have a set of different segmentations of our distributional data. The algorithm was also run multiple times with different seeds for randomization. Visualizations of all segmentations and results presented in this section are shown in Appendix A.

Figure 6.12 shows an objective space plot for our two most important objectives, population above threshold (O_1) and the number of segments (O_3), both subject to minimization. Here we can see how our solutions converge towards the minimum for both objectives, and we can see the Pareto front of the final generation.

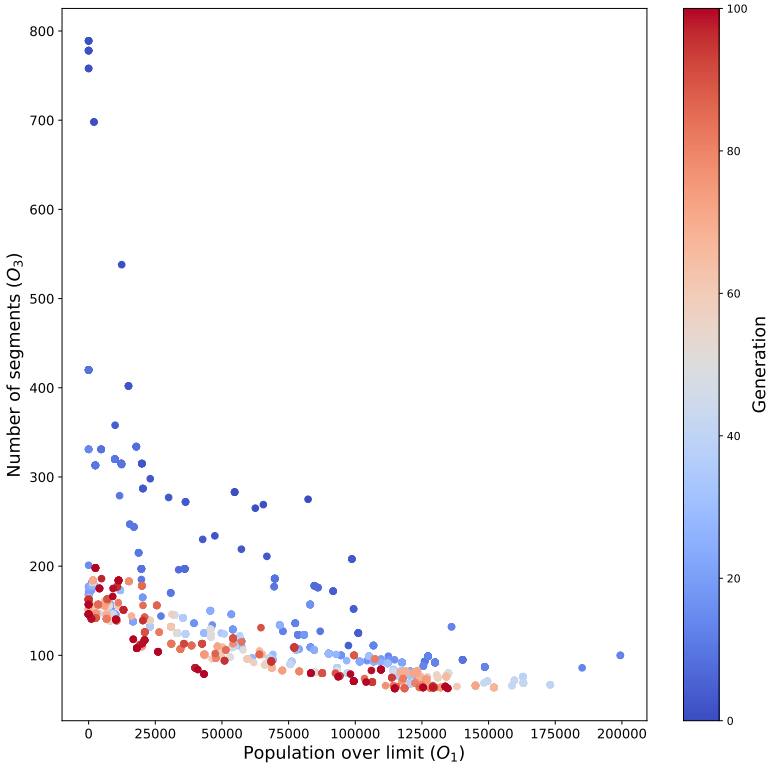

Figure 6.12: Objective space plot for objectives O_1 and O_3 of an arbitrary run of our algorithm with 100 generations.

As a simple naming convention, we give each segmentation a name of vX.Y, where X is the number of the full run of the algorithm and Y is the segmentation within the different runs. After the first run of our algorithm, we made a subjective evaluation of the different segmentations and landed on selecting our v1.0. In later runs, we saved different segmentations, e.g., the best performing in each objective alongside segmentations subjectively deemed feasible. All objectives of the segmentations evaluated are presented in Tables 6.10a to 6.10f.

Table 6.10: Objectives for different segmentations.

(a) v1.0.

Objective	Value
O_1	4 050
O_2	0
O_3	177
O_4	4 892
Visualization	Figure 6.13

(b) v2.0.

Objective	Value
O_1	175 879
O_2	16
O_3	56
O_4	3 242
Visualization	Figure A.1

(c) v2.1.

Objective	Value
O_1	0
O_2	0
O_3	85
O_4	3 668
Visualization	Figure A.3

(d) v2.2.

Objective	Value
O_1	177 599
O_2	0
O_3	60
O_4	2 982
Visualization	Figure A.5

(e) v2.3.

Objective	Value
O_1	9 290
O_2	0
O_3	72
O_4	3 668
Visualization	Figure A.7

(f) v2.4.

Objective	Value
O_1	0
O_2	0
O_3	108
O_4	3 576
Visualization	Figure A.9

Segmentations v2.0, v2.1, and v2.2 have the best objectives in the number of segments (O_3), population over the limit (O_1), and total circumference (O_4), respectively. The area above the limit (O_2) as an objective was 0 for most segmentations in the first rank of iteration 2, so we decided not to use one specific segmentation with this objective. Segmentations v2.3 and v2.4 are two subjectively good segmentations with a balanced distribution of the different objectives. We see from the other segmentations that having a few separate segments gives a high population over the limit, which is in accordance with the objective space plot in Figure 6.12. In contrast, a low population over the limit yields a higher total circumference. Similarly, a low total circumference also pushes the population over the limit and reduces the number of segments. Presented in Figure 6.13 is an illustration of segmentation v1.0, where we can see that the segments in the center of Oslo have a higher resolution than those further out in the rural areas.

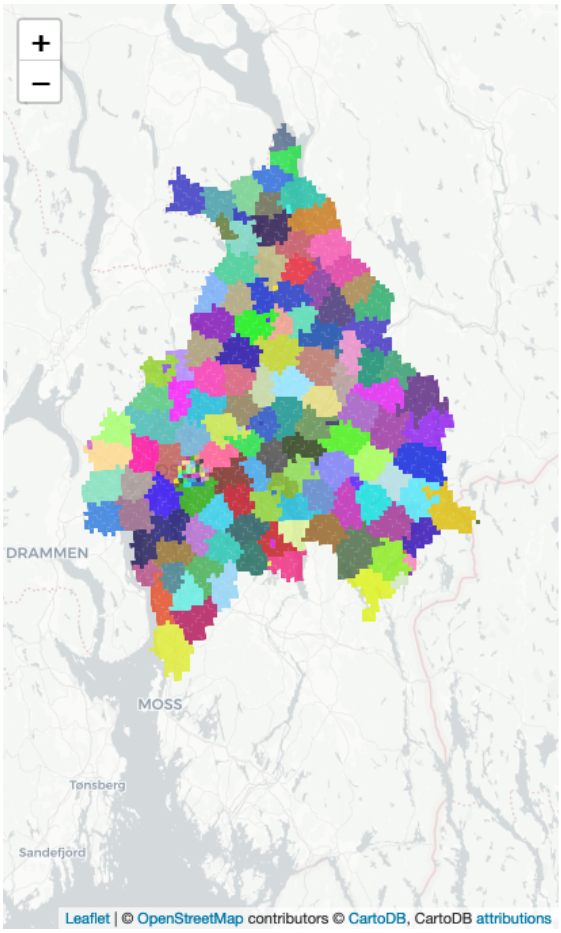

Figure 6.13: Segmentation v1.0. Each color is a separate segment, in total 177 segments.

After creating our different segmentations, we aggregate the incidents per hour for each segment and create new data sets for each segmentation. This new data is then the basis for an MLP_{Adam} model to train on and forecast, where the input to the model is the best-performing input set from Section 6.4.2. The model is evaluated using k-fold cross-validation with $k = 5$. Early stopping is also applied to each fold, with a patience of 5. We use CCE as our performance metric to compare distributional methods, as this is a relatively good measure when comparing two distributions. Here we compare our model to the historical average and an adapted all 0s that forecasts an equal distribution for each segment. From previous results in Section 6.4.2, we have that the historical average is difficult to outperform, as our model for predicting the highest resolution was unable to outperform the historical average previously. We include modified all 0s to illustrate how the sparsity of our data changes with our aggregation.

Table 6.11: Average CCE for each segmentation from each model.

Segmentation	MLP	Historical Average	All 0s	Visualization
v1.0	**3.7447**	3.7497	5.1761	Figure 6.14
v2.0	**2.5126**	2.5136	4.0254	Figure A.2
v2.1	**3.1926**	3.1947	4.4427	Figure A.4
v2.2	**2.4714**	2.4735	4.0943	Figure A.6
v2.3	**3.1265**	3.1295	4.2767	Figure A.8
v2.4	**3.4698**	3.4739	4.6821	Figure A.10

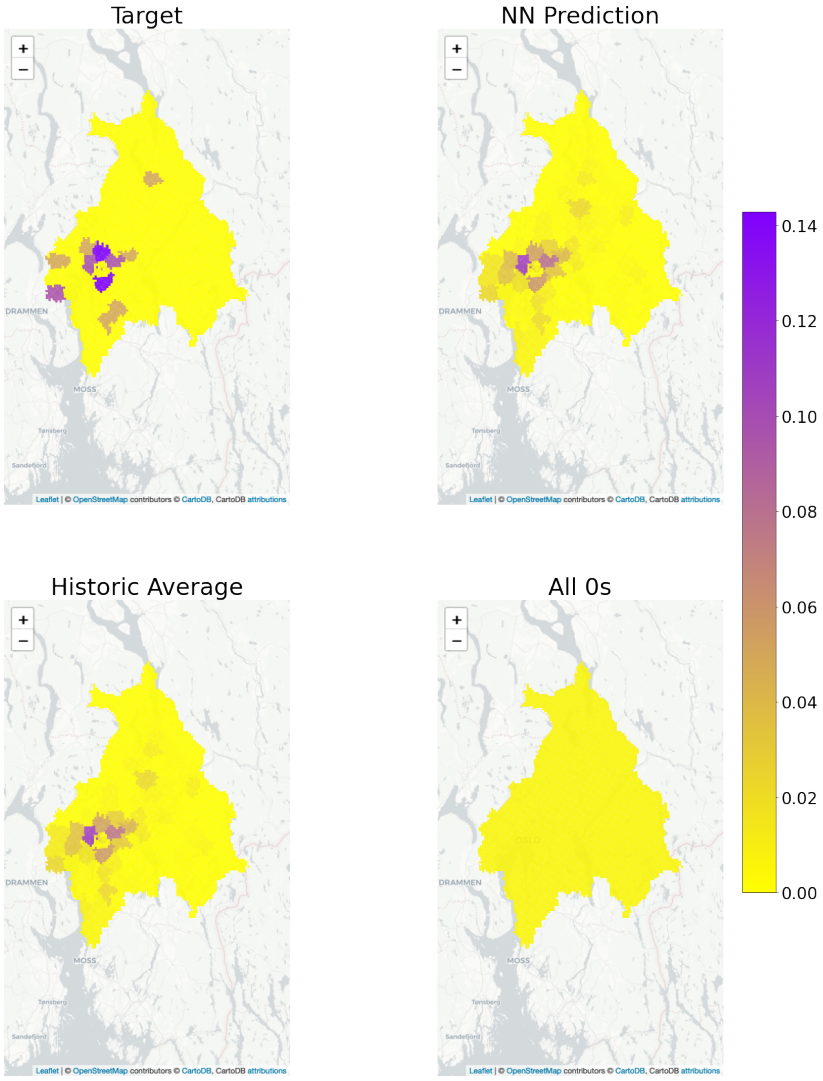

Figure 6.14: Predicted distribution using segmentation v1.0 for the hour 2015-04-21 09:00 - 10:00.

Figure 6.14 illustrates a predicted distribution from the three different models, as well as the target for segmentation v1.0. The results for each of the models on each of the segmentations are presented in Table 6.11. Illustrations of all results are presented in Appendix A. We randomly select a single hour of our data set to produce illustrations of our predicted distributions from different models and segmentations. From the figures in Figure 6.14, we see that the MLP-model prediction and historical average are strikingly similar.

6.6 Discussion

Most of our results are quantifiable, and we compare them with different approaches from related research. We are still interested in reflecting on the findings and how our results relate to the operations at OUH.

Volume Forecasts

Section 6.3 proposes models that improve volume forecasts from previous work within the field by decomposing the time series and applying machine learning and regression to forecast the volume demand based on the components. We believe these results can be improved further by applying more extensive hyperparameter-tuning of both decomposition and machine learning models.

One of the approaches for capturing the increasing trend in our time series was including the year in our input data. We did this by representing the number of years, after the start of our time series, as a value between 0 and 1. Although including the year in this way improved forecasting accuracy, a finer value, such as the number of weeks since the start of our time series, could further improve forecasting accuracy.

Another finding of our volume forecasts is through the application of a modified MSE error metric to give a higher loss for underestimation of volume. Although this gives a higher MSE score, the actual forecasts seem in line with the proposed goal of the error metric when applied as a loss function in DFO. For the operations at OUH and planning of resources, having forecasts that help satisfy a minimum of available resources at any given incident is sometimes more important than having a low mean error from a target.

Distribution Forecasts

In Section 6.4, we apply two different approaches at the highest resolution for distribution forecasting. Our first model, WGAN, did not yield the best results, and we believe this might

be due to a combination of two factors. Firstly, our theoretical background with generative adversarial networks may not be strong enough to apply WGAN in this setting properly. Secondly, we believe that the issue with sparsity remains highly relevant when attempting to generate realistic scenarios. The expansion from 5 569 cells to a 124 x 124 grid (15 376 cells) makes the sparsity issue even more prevalent. There are at most 36 distinct cells with incidents in a given hour, which makes up only 0.23% of the total possible cells when scaled to 124 x 124.

We apply an MLP_{Adam} model, previously proven to give good results in distribution forecasting, with an experiment to assert how the input variables relate to the forecasts. The primary factor of the input variables seems to be the inclusion of the hour of the day. All scores from inputs that include the hour outperform those without the hour. This difference is minor, which leads us to believe that the distributional shift of where incidents happen during a day is weaker than first assumed and might indicate that the part of the population most frequently involved in incidents requiring medical attention stays within a limited area throughout the day.

Aggregation and Distribution Forecasts

Finally, in Section 6.5, we apply GAs to create segmentations of Oslo and Akershus to overcome the issue of demand sparsity. One of the significant results here is the ability of MLP_{Adam} to outperform the historical average, which it was not able to do at the highest resolution. Again, the difference is minor, and as seen from Figure 6.14, the forecast of MLP_{Adam} and the historical average are similar. This similarity further underlines that there seems to be little distributional shift through the day, week, and year.

Chapter 7

Conclusion

This chapter concludes our research by reiterating our research goals and discussing how the results from the previous chapter answer them. We present limitations to our research and introduce suggestions for future work.

7.1 Evaluation

In this section, we evaluate our results in light of our research questions presented in Chapter 1.

7.1.1 Research Question 1

Research question 1 *How can we forecast demand at a high spatio-temporal resolution?*

In Section 6.3, we present the accuracy of our proposed volume forecasting models, along with industry baselines. The results show that we can produce more accurate forecasts than industry baselines using all of our models. Our results suggest that machine learning models can outperform baselines and that these models can further be improved by tailoring the input data, the architecture of the model, and using decomposition to pre-process the data.

We found that the increasing yearly trend should be incorporated somehow into the data. We found success in this using both STL, SSA, and including the year as a numerical value in our input data set. The year having such an effect would suggest that forecasting methods based on a moving average sampled from previous years, such as Medic and Setzler, cannot capture an essential feature in predicting ambulance demand.

Although our distribution forecasting models did not perform as well as we had hoped at a high resolution, we could extract some information about the data based on the results. The

most prevalent indication from these experiments is the assumption that the population most frequently in need of ambulances seems to remain roughly within the same area throughout the day, week, and year. The way for an ANN to outperform the historical average is through the inclusion of variables that correlate to the target values, which also vary throughout the time series. Otherwise, gradient descent will converge toward the historical average of the training data due to the lack of a relationship between input and output.

7.1.2 Research Question 2

Research question 2 *Can time series decomposition be used in combination with machine learning models to produce improved total hourly ambulance demand?*

Our results indicate that decomposition with STL and SSA can improve model accuracy. Our intuition that extracting seasonal and trend components of the time series will simplify the problem for the machine learning model to learn seems to be correct.

Although we got improved results using decomposition and machine learning, perhaps these models could be further improved with decomposition using more finely tuned parameters or methods for our problem area.

7.1.3 Research Question 3

Research question 3 *Can aggregation of spatial data improve forecasts of EMS demand?*

From our distributional forecasts, we found that an MLP-model was unable to outperform the historical average at the highest resolution of our data set. To reduce the sparsity of our data, we applied a GA to aggregate distributional data, which made it possible for our MLP-model to outperform the historical average by a thin margin. Most of our results indicate that the aggregation of spatial data improves the precision of the forecasts at the cost of resolution. Although we can not compare different aggregations due to the different number of segments affecting the CCE-score, comparing to a deterministic forecasting method in historical average makes it possible to compare the performance of models on different aggregations.

7.2 Future Work

Volume forecasts could be improved by using special flags in the input data for special events. This is because events such as big concerts, sports events, new years eve, and national holi-

days might be directly related to an increase in medical emergencies.

Given our indication of the population in need of ambulances staying roughly within the same area, we believe that some additional data set exploration could further investigate this indication. Some additional information about the incidents would be interesting, such as the age group of the people involved in the incident, but this might be problematic concerning anonymity. We believe that age is an essential factor in the need for ambulances and the urgency level. Given an assumed correlation between age and urgency, it is interesting to look at how each urgency level affects the distribution of incidents. Separating distributions on an urgency level will increase the sparsity of the data even more, so the aggregation techniques presented in this thesis could be relevant to further apply and improve for future research. Other ways of aggregating data are also an interesting approach to reducing sparsity.

Another approach we believe would improve distributional forecast is the inclusion of external data about the distribution of the population throughout the day, week, and year. Including external data is also suggested by Martin et al. (2021), who also proposes including domain knowledge from EMS professionals about the optimal levels of spatial resolution when aggregating spatial data. More generally, we have that the way for an ANN to outperform the historical average is to include highly correlated variables that change throughout the time series, as mentioned previously.

Martin et al. (2021) proposes k-means to create a few large areas to perform time series analysis and forecasting within each area instead of forecasting distributions. We believe this approach is interesting to explore further for our data set, with different techniques for aggregation or using areas suggested by the EMS themselves, sufficiently covering Oslo and Akershus.

This thesis focuses on forecasting demand volume and distribution of medical emergencies. Forecasts could be an important tool in the planning of resource allocation. Another approach to optimizing resource allocation is to look at this problem directly using optimization models. Creating a simulation model to evaluate different combinations of resource allocation and using heuristic search algorithms to find optimal solutions is one approach. We believe this can be combined with our forecasting models to find optimal resource allocations for future events.

Bibliography

Aggarwal, C. (2003). A framework for diagnosing changes in evolving data streams, 575. https://doi.org/10.1145/872824.872826

Anderson, R. L. (1953). Recent advances in finding best operating conditions. *Journal of the American Statistical Association*, *48*(264), 789–798. https://doi.org/10.1080/01621459.1953.10501200

Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein GAN. https://doi.org/10.48550/ARXIV.1701.07875

Atienza, R. (2020). *Advanced deep learning with TensorFlow 2 and Keras*. Packt Publishing Ltd.

Aytug, H. & Saydam, C. (2002). Solving large-scale maximum expected covering location problems by genetic algorithms: A comparative study. *European Journal of Operational Research*, *141*(3), 480–494. https://doi.org/https://doi.org/10.1016/S0377-2217(01)00260-0

Basak, A., Mengshoel, O. J., Kulkarni, C., Schmidt, K., Shastry, P. & Rapeta, R. (2017). Optimizing the decomposition of time series using evolutionary algorithms: Soil moisture analytics. *Proceedings of the Genetic and Evolutionary Computation Conference*, 1073–1080. https://doi.org/10.1145/3071178.3071191

Braun, O., McCallion, R. & Fazackerley, J. (1990). Characteristics of midsized urban ems systems. *Annals of Emergency Medicine*, *19*(5), 536–546. https://doi.org/https://doi.org/10.1016/S0196-0644(05)82186-9

Brentan, B. M., Luvizotto Jr., E., Herrera, M., Izquierdo, J. & Pérez-García, R. (2017). Hybrid regression model for near real-time urban water demand forecasting. *Journal of Computational and Applied Mathematics*, *309*, 532–541. https://doi.org/https://doi.org/10.1016/j.cam.2016.02.009

Brockwell, P. J. & Davis, R. A. (2002). *Introduction to time series and forecasting*. Springer.

Brunsdon, C., Corcoran, J. & Higgs, G. (2007). Visualising space and time in crime patterns: A comparison of methods [Extracting Information from Spatial Datasets]. *Computers,*

Environment and Urban Systems, *31*(1), 52–75. https://doi.org/https://doi.org/10.1016/j.compenvurbsys.2005.07.009

Chen, A. Y., Lu, T.-Y., Ma, M. H.-M. & Sun, W.-Z. (2016). Demand forecast using data analytics for the preallocation of ambulances. *IEEE Journal of Biomedical and Health Informatics*, *20*(4), 1178–1187. https://doi.org/10.1109/JBHI.2015.2443799

Chen, K.-Y. & Wang, C.-H. (2007). Support vector regression with genetic algorithms in forecasting tourism demand. *Tourism Management*, *28*(1), 215–226. https://doi.org/https://doi.org/10.1016/j.tourman.2005.12.018

Chen, Y., Xu, P., Chu, Y., Li, W., Wu, Y., Ni, L., Bao, Y. & Wang, K. (2017). Short-term electrical load forecasting using the support vector regression (SVR) model to calculate the demand response baseline for office buildings. *Applied Energy*, *195*, 659–670. https://doi.org/https://doi.org/10.1016/j.apenergy.2017.03.034

Chollet, F. et al. (2015). *Keras*. https://github.com/fchollet/keras

Cleveland, R. B., Cleveland, W. S. & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Fattaheian-Dehkordi, S., Fereidunian, A., Gholami-Dehkordi, H. & Lesani, H. (2014). Hour-ahead demand forecasting in smart grid using support vector regression (SVR). *International Transactions on Electrical Energy Systems*, *24*(12), 1650–1663. https://doi.org/https://doi.org/10.1002/etep.1791

Golyandina, N. & Korobeynikov, A. (2014). Basic singular spectrum analysis and forecasting with R. *Computational Statistics & Data Analysis*, *71*, 934–954. https://doi.org/https://doi.org/10.1016/j.csda.2013.04.009

Golyandina, N., Korobeynikov, A. & Zhigljavsky, A. (2010). *Singular Spectrum Analysis with R*.

Haugsbø Hermansen, A. & Mengshoel, O. J. (2021). Forecasting ambulance demand using machine learning: A case study from Oslo, Norway. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 01–10. https://doi.org/10.1109/SSCI50451.2021.9659837

Herbert, Z. C., Asghar, Z. & Oroza, C. A. (2021). Long-term reservoir inflow forecasts: Enhanced water supply and inflow volume accuracy using deep learning. *Journal of Hydrology*, *601*, 126676. https://doi.org/https://doi.org/10.1016/j.jhydrol.2021.126676

Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/https://doi.org/10.1016/0893-6080(89)90020-8

Huang, H., Jiang, M., Ding, Z. & Zhou, M. (2019). Forecasting emergency calls with a Poisson neural network-based assemble model. *IEEE Access*, *7*, 18061–18069. https://doi.org/10.1109/ACCESS.2019.2896887

Jiang, S., Chin, K.-S., Wang, L., Qu, G. & Tsui, K. L. (2017). Modified genetic algorithm-based feature selection combined with pre-trained deep neural network for demand forecasting in outpatient department. *Expert Systems with Applications*, *82*, 216–230. https://doi.org/https://doi.org/10.1016/j.eswa.2017.04.017

Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kingma, D. & Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Martin, R. J., Mousavi, R. & Saydam, C. (2021). Predicting emergency medical service call demand: A modern spatiotemporal machine learning approach. *Operations Research for Health Care*, *28*, 100285. https://doi.org/https://doi.org/10.1016/j.orhc.2021.100285

Matteson, D. S., McLean, M. W., Woodard, D. B. & Henderson, S. G. (2011). Forecasting emergency medical service call arrival rates. *The Annals of Applied Statistics*, *5*(2B), 1379–1406. https://doi.org/10.1214/10-AOAS442

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In G. I. Marchuk (Ed.), *Optimization techniques IFIP technical conference novosibirsk, july 1–7, 1974* (pp. 400–404). Springer Berlin Heidelberg.

Nakaya, T. & Yano, K. (2010). Visualising crime clusters in a space-time cube: An exploratory data-analysis approach using space-time kernel density estimation and scan statistics. *Transactions in GIS*, *14*(3), 223–239. https://doi.org/https://doi.org/10.1111/j.1467-9671.2010.01194.x

Olsen, S., Ilkka, L., Berlac, P. A. et al. (2019). *The Nordic emergency medical services* (tech. rep. DIS-2750). Norwegian Directorate of Health.

Procopiuc, C. & Procopiuc, O. (2005). Density estimation for spatial data streams. *3633*, 109–126. https://doi.org/10.1007/11535331_7

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Rpy2. (n.d.). Retrieved 28th April 2022, from https://rpy2.github.io/doc/latest/html/index.html

Seabold, S. & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*.

Setzler, H., Saydam, C. & Park, S. (2009). EMS call volume predictions: A comparative study. *Computers & Operations Research*, *36*, 1843–1851. https://doi.org/10.1016/j.cor.2008.05.010

Tascikaraoglu, A., Sanandaji, B. M., Chicco, G., Cocina, V., Spertino, F., Erdinc, O., Paterakis, N. G. & Catalão, J. P. (2016). Compressive spatio-temporal forecasting of meteorological quantities and photovoltaic power. *IEEE Transactions on Sustainable Energy*, *7*(3), 1295–1305. https://doi.org/10.1109/TSTE.2016.2544929

Tieleman, T., Hinton, G. et al. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26–31.

Ulrich, M., Jahnke, H., Langrock, R., Pesch, R. & Senge, R. (2021). Distributional regression for demand forecasting in e-grocery. *European Journal of Operational Research*, *294*(3), 831–842. https://doi.org/https://doi.org/10.1016/j.ejor.2019.11.029

Van Rossum, G. & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

Venkatesh, K., Ravi, V., Prinzie, A. & den Poel, D. V. (2014). Cash demand forecasting in atms by clustering and neural networks. *European Journal of Operational Research*, *232*(2), 383–392. https://doi.org/https://doi.org/10.1016/j.ejor.2013.07.027

Wilesmith, J., Stevenson, M., King, C. & Morris, R. (2003). Spatio-temporal epidemiology of foot-and-mouth disease in two counties of Great Britain in 2001. *Preventive Veterinary Medicine*, *61*(3), 157–170. https://doi.org/https://doi.org/10.1016/j.prevetmed.2003.08.002

Wong, H. T. & Lai, P. C. (2012). Weather inference and daily demand for emergency ambulance services. *Emergency Medicine Journal*, *29*(1), 60–64. https://doi.org/10.1136/emj.2010.096701

Wu, C., Chau, K. & Fan, C. (2010). Prediction of rainfall time series using modular artificial neural networks coupled with data-preprocessing techniques. *Journal of Hydrology*, *389*(1), 146–167. https://doi.org/https://doi.org/10.1016/j.jhydrol.2010.05.040

Xu, C., Ji, J. & Liu, P. (2018). The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets. *Transportation Research Part C: Emerging*

Technologies, *95*, 47–60. https://doi.org/https://doi.org/10.1016/j.trc.2018.07.013

Xu, D., Zhang, S., Zhang, H. & Mandic, D. P. (2021). Convergence of the rmsprop deep learning method with penalty for nonconvex optimization. *Neural Networks*, *139*, 17–23. https://doi.org/https://doi.org/10.1016/j.neunet.2021.02.011

Yamashita, G. H., Fogliatto, F. S., Anzanello, M. J. & Tortorella, G. L. (2022). Customized prediction of attendance to soccer matches based on symbolic regression and genetic programming. *Expert Systems with Applications*, *187*, 115912. https://doi.org/https://doi.org/10.1016/j.eswa.2021.115912

Yu, B., Yin, H. & Zhu, Z. (2017). Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. *CoRR*, *abs/1709.04875*. http://arxiv.org/abs/1709.04875

Zanc, R., Cioara, T. & Anghel, I. (2019). Forecasting financial markets using deep learning. *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 459–466. https://doi.org/10.1109/ICCP48234.2019.8959715

Zhang, Y., Li, G., Muskat, B. & Law, R. (2021). Tourism demand forecasting: A decomposed deep learning approach. *Journal of Travel Research*, *60*(5), 981–997. https://doi.org/10.1177/0047287520919522

Zhang, Z., Chen, D., Liu, W., Racine, J. S., Ong, S., Chen, Y., Zhao, G. & Jiang, Q. (2011). Non-parametric evaluation of dynamic disease risk: A spatio-temporal kernel approach. *PLOS ONE*, *6*(3), 1–8. https://doi.org/KDEdisease

Zhou, Z. & Matteson, D. S. (2015). Predicting ambulance demand: A spatio-temporal kernel approach. https://doi.org/10.48550/ARXIV.1507.00364

Appendix A

Segmentation Results and Predictions

A.1 Segmentation v2.0

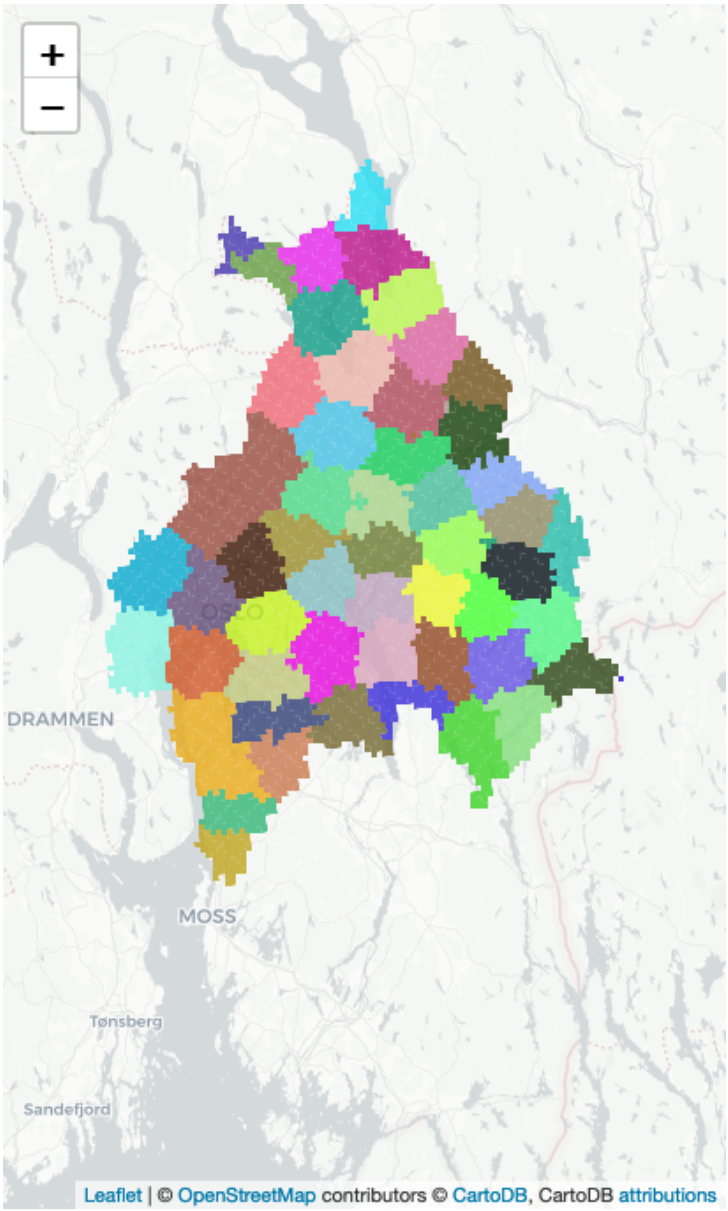

Figure A.1: Segmentation v2.0. Each color is a separate segment, in total 56 segments.

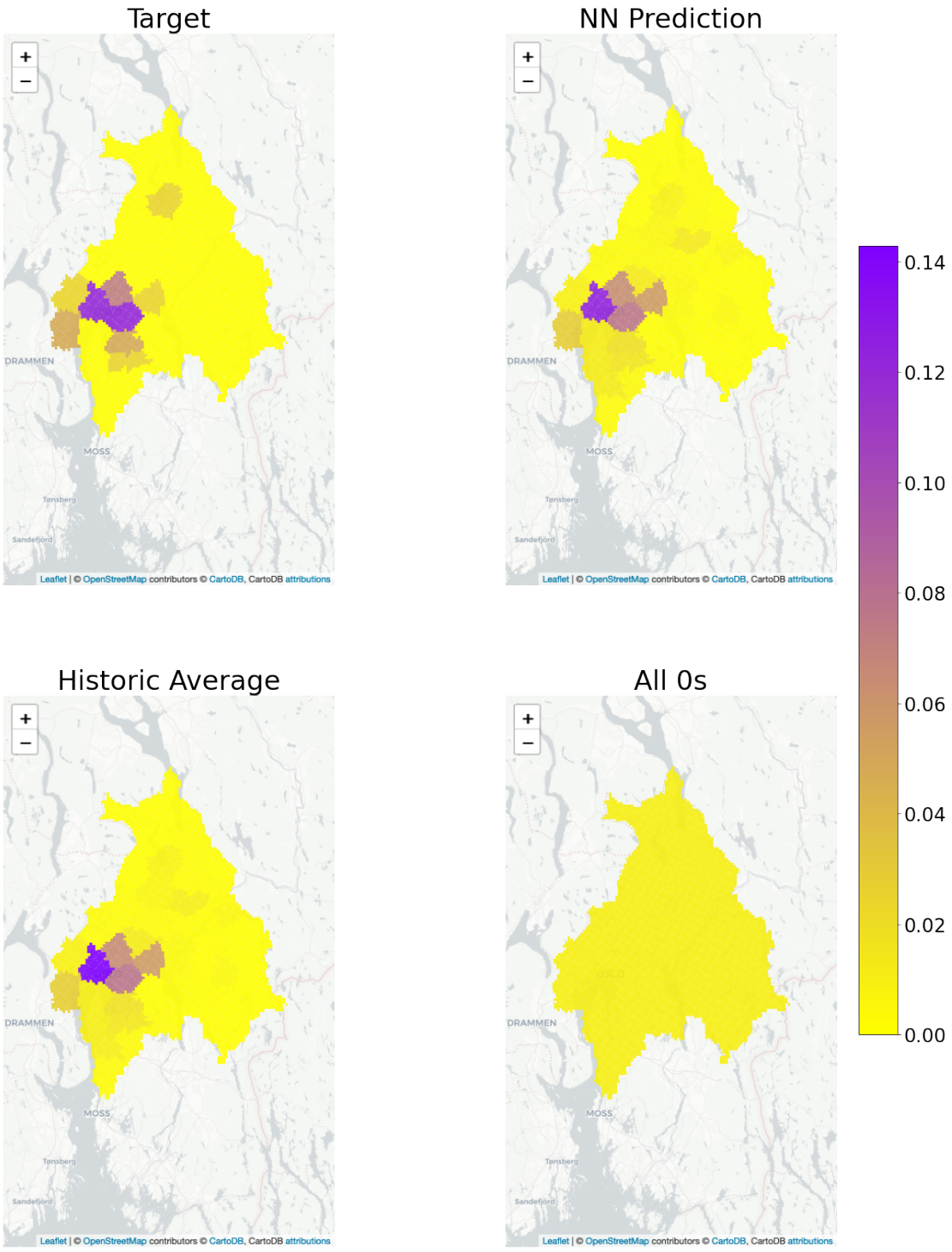

Figure A.2: Predicted distribution using segmentation v2.0 for the hour 2015-04-21 09:00 - 10:00.

A.2 Segmentation v2.1

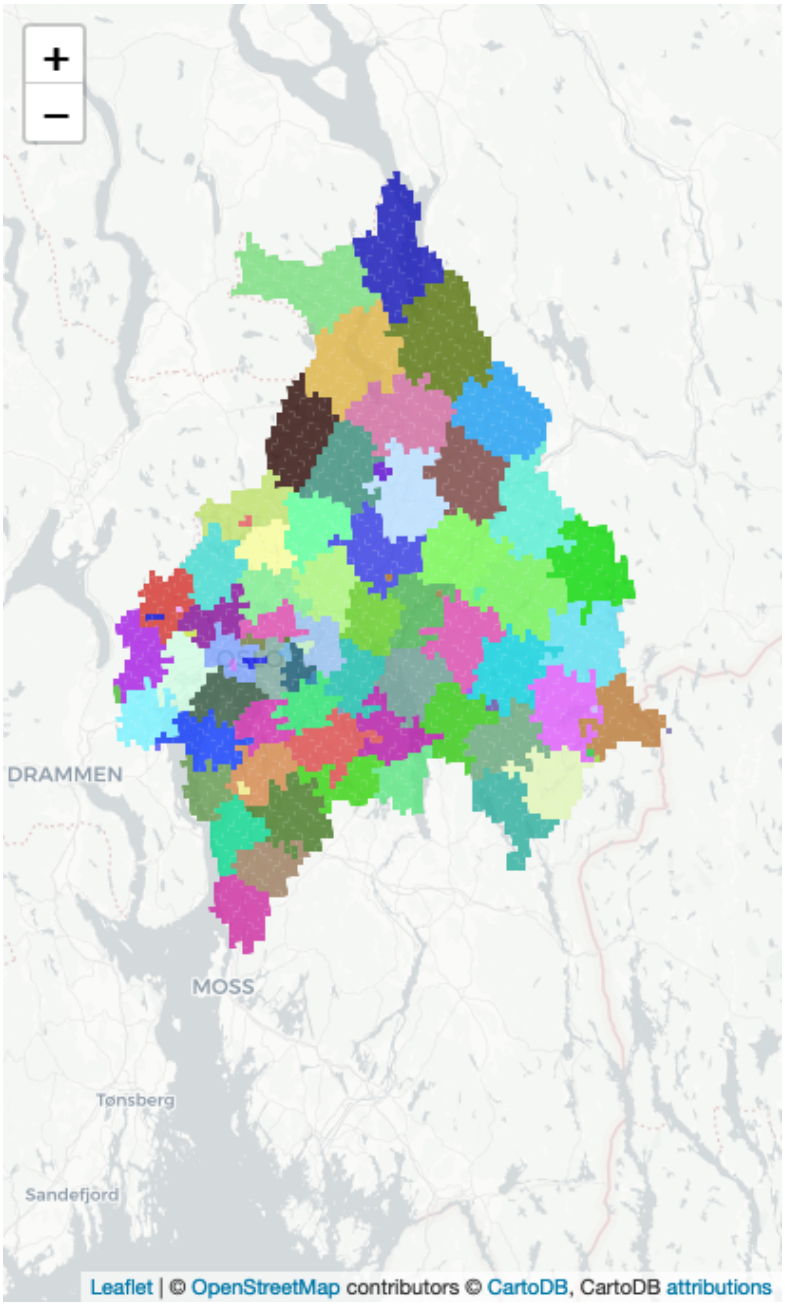

Figure A.3: Segmentation v2.1. Each color is a separate segment, in total 85 segments.

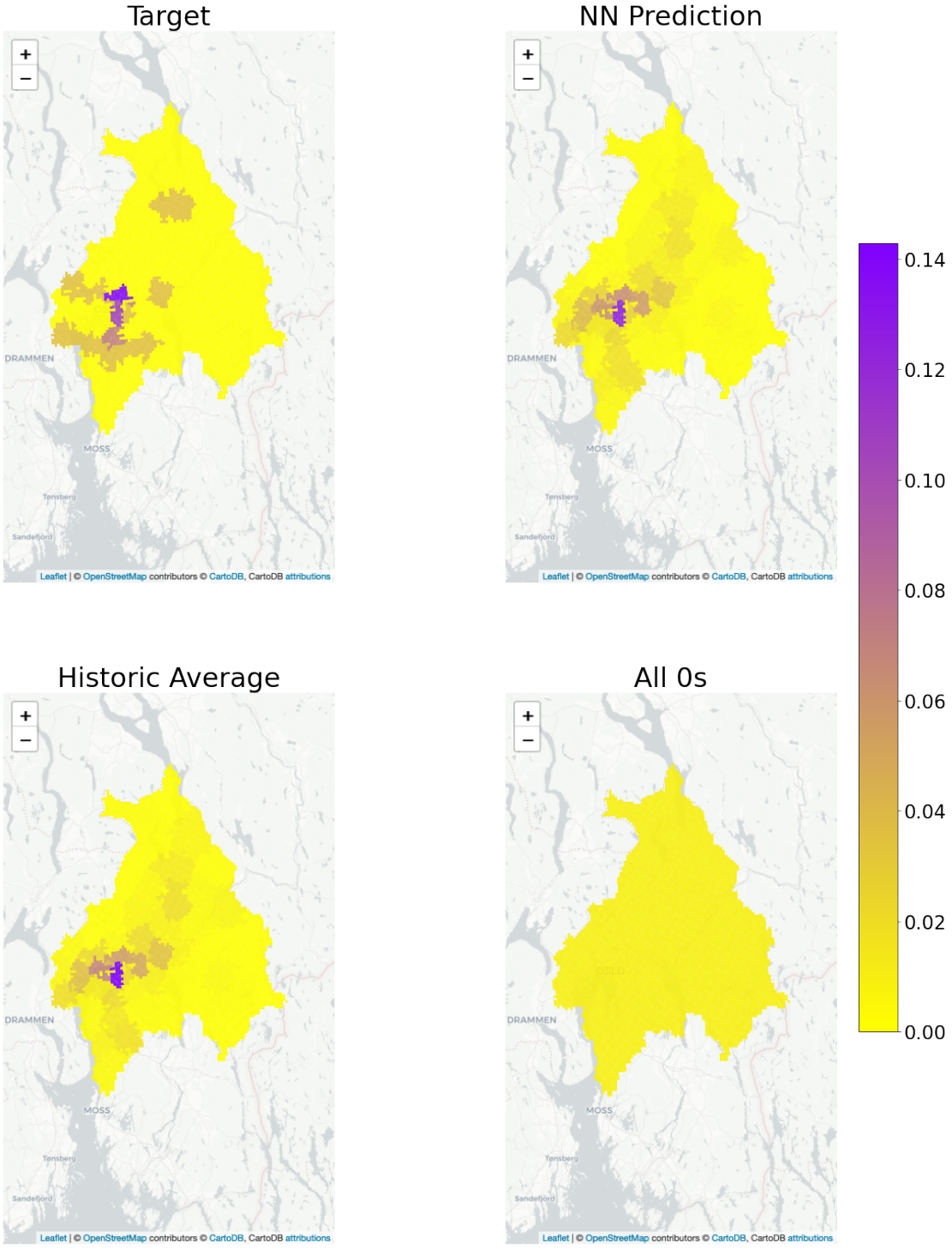

Figure A.4: Predicted distribution using segmentation v2.1 for the hour 2015-04-21 09:00 - 10:00.

A.3 Segmentation v2.2

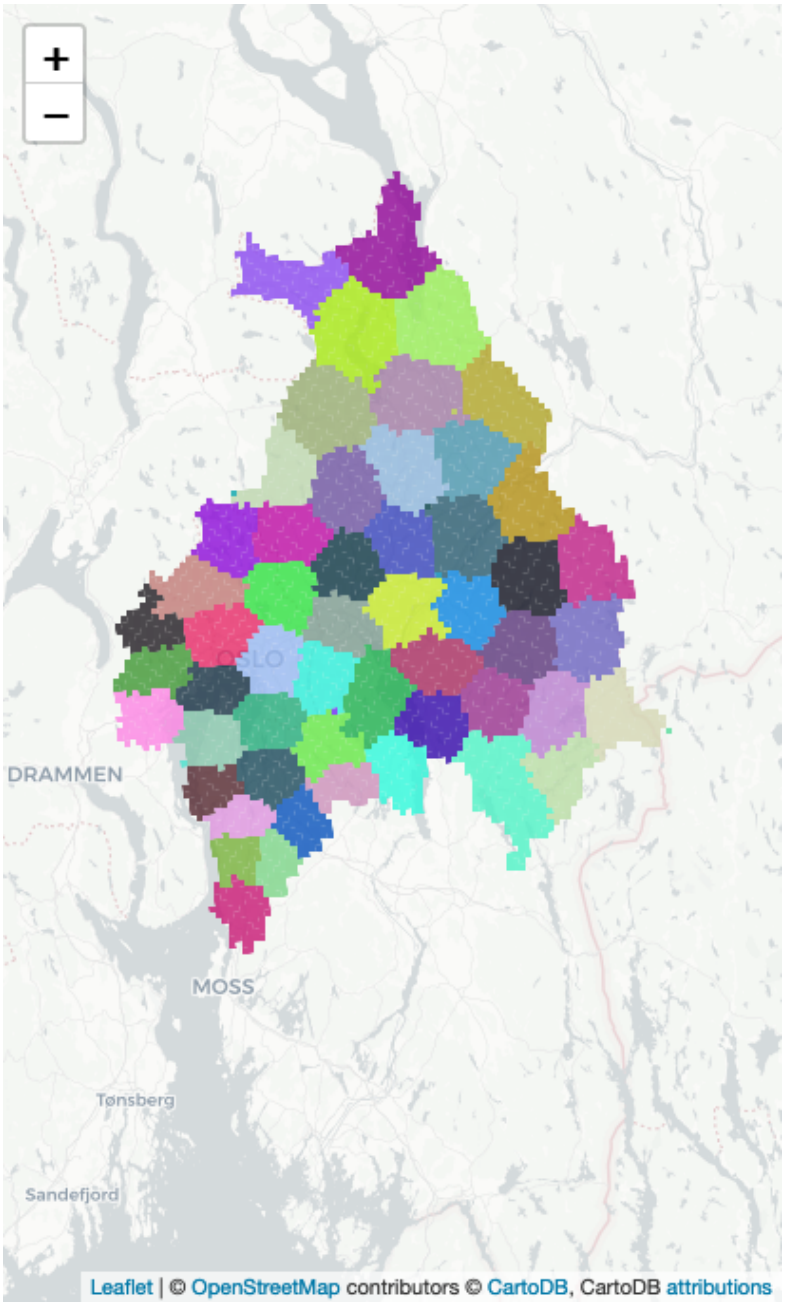

Figure A.5: Segmentation v2.2. Each color is a separate segment, in total 60 segments.

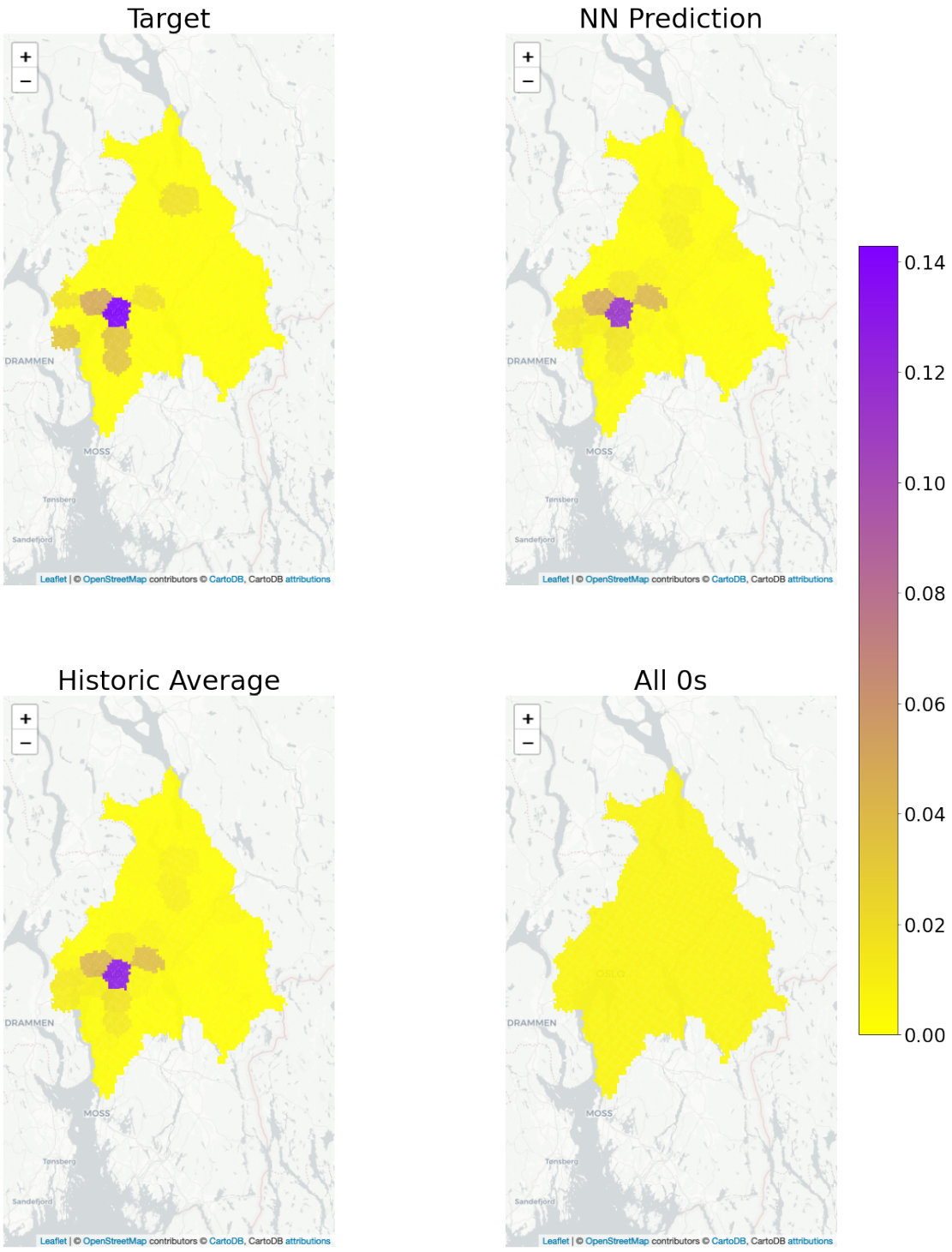

Figure A.6: Predicted distribution using segmentation v2.2 for the hour 2015-04-21 09:00 - 10:00.

A.4 Segmentation v2.3

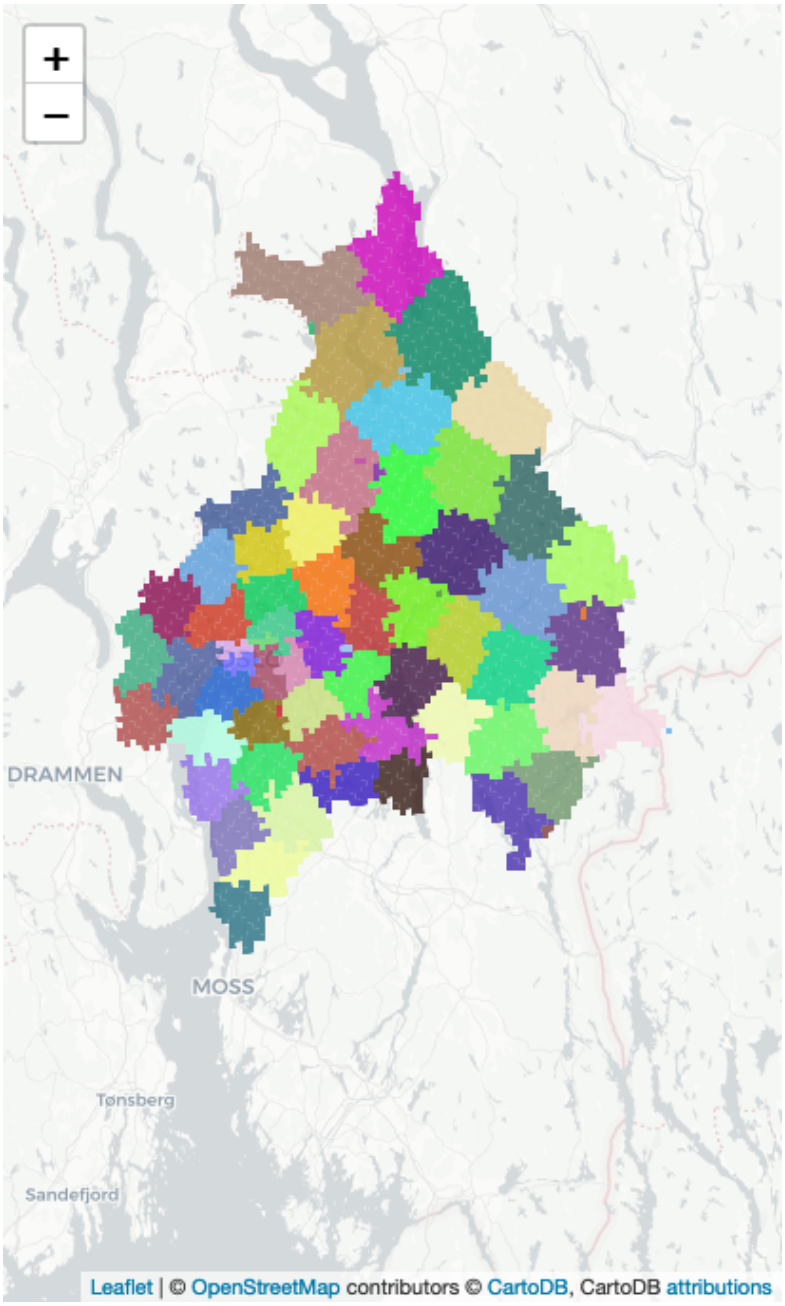

Figure A.7: Segmentation v2.3. Each color is a separate segment, in total 72 segments.

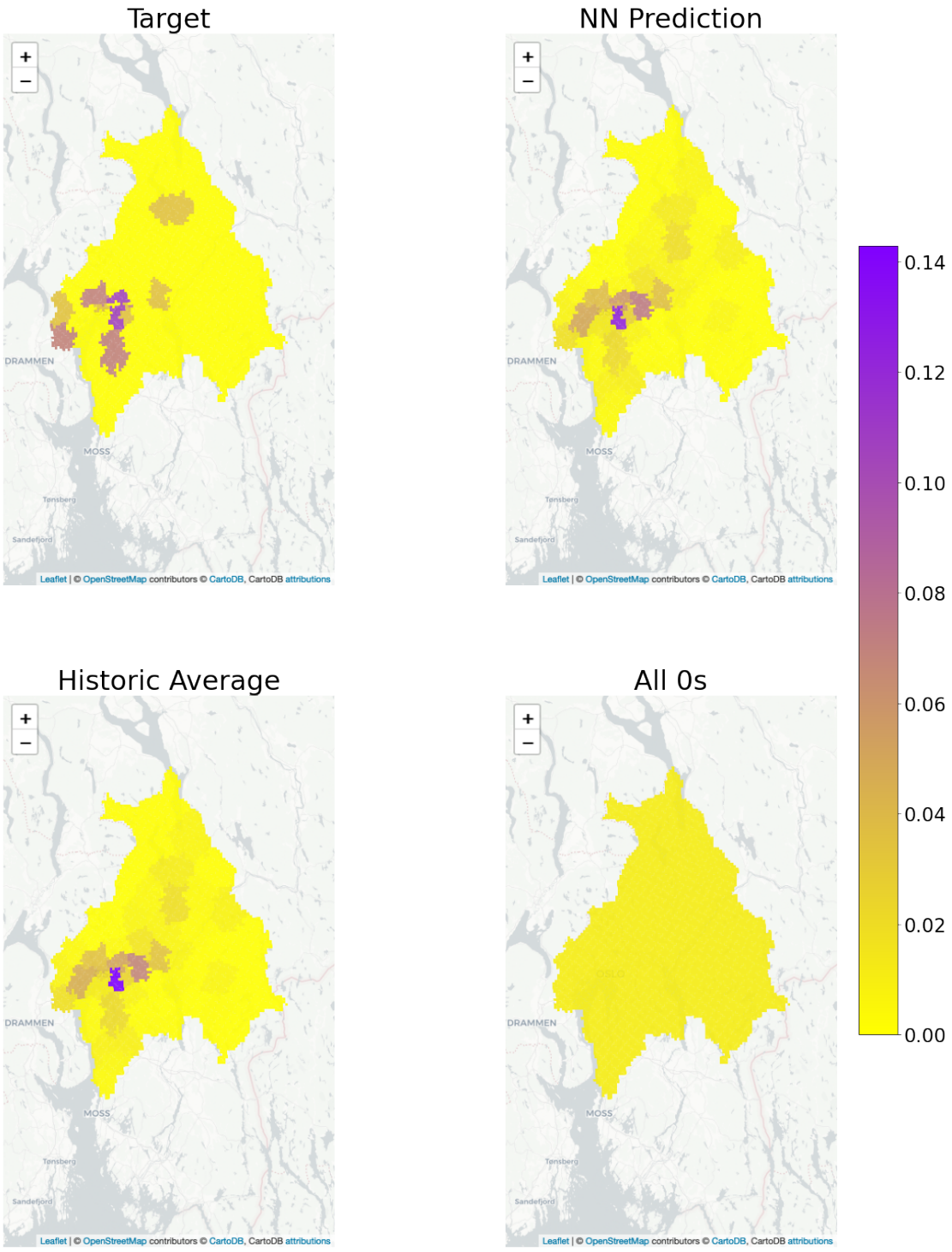

Figure A.8: Predicted distribution using segmentation v2.3 for the hour 2015-04-21 09:00 - 10:00.

A.5 Segmentation v2.4

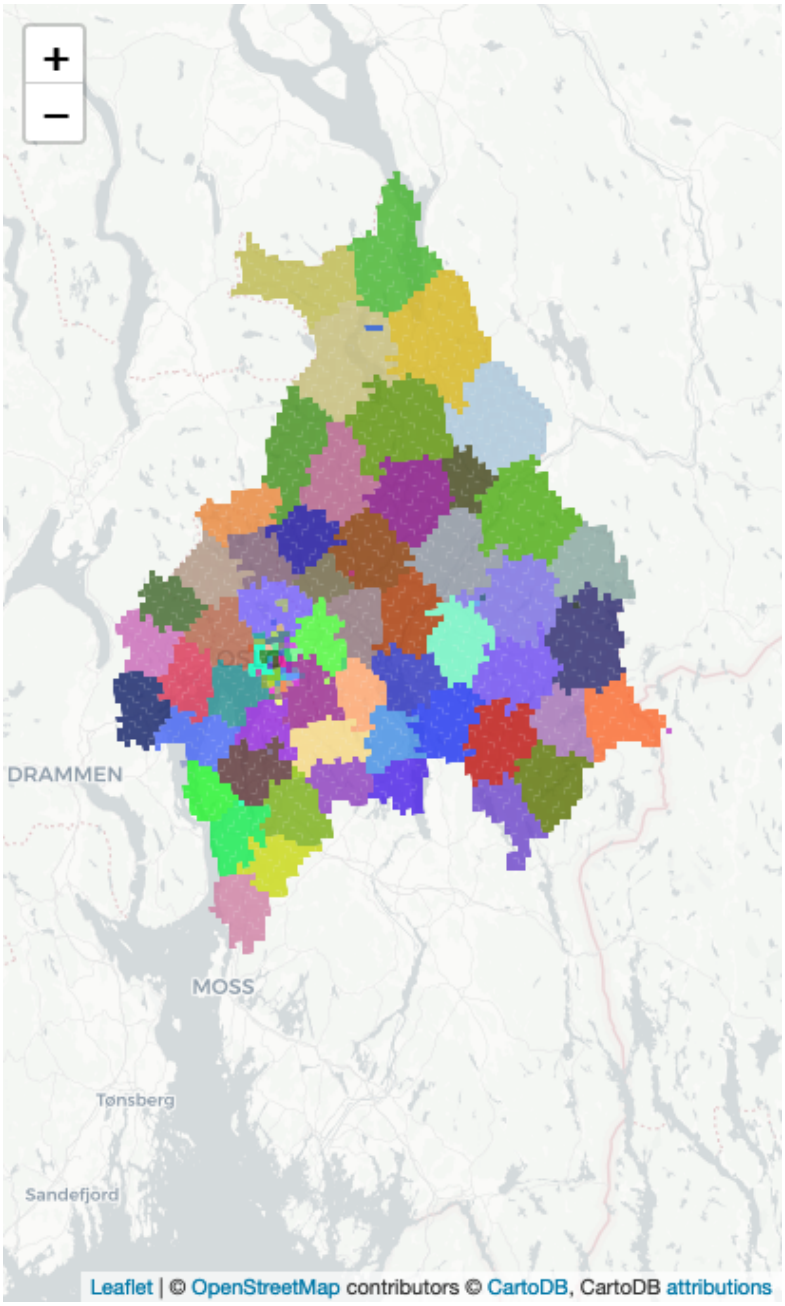

Figure A.9: Segmentation v2.4. Each color is a separate segment, in total 108 segments.

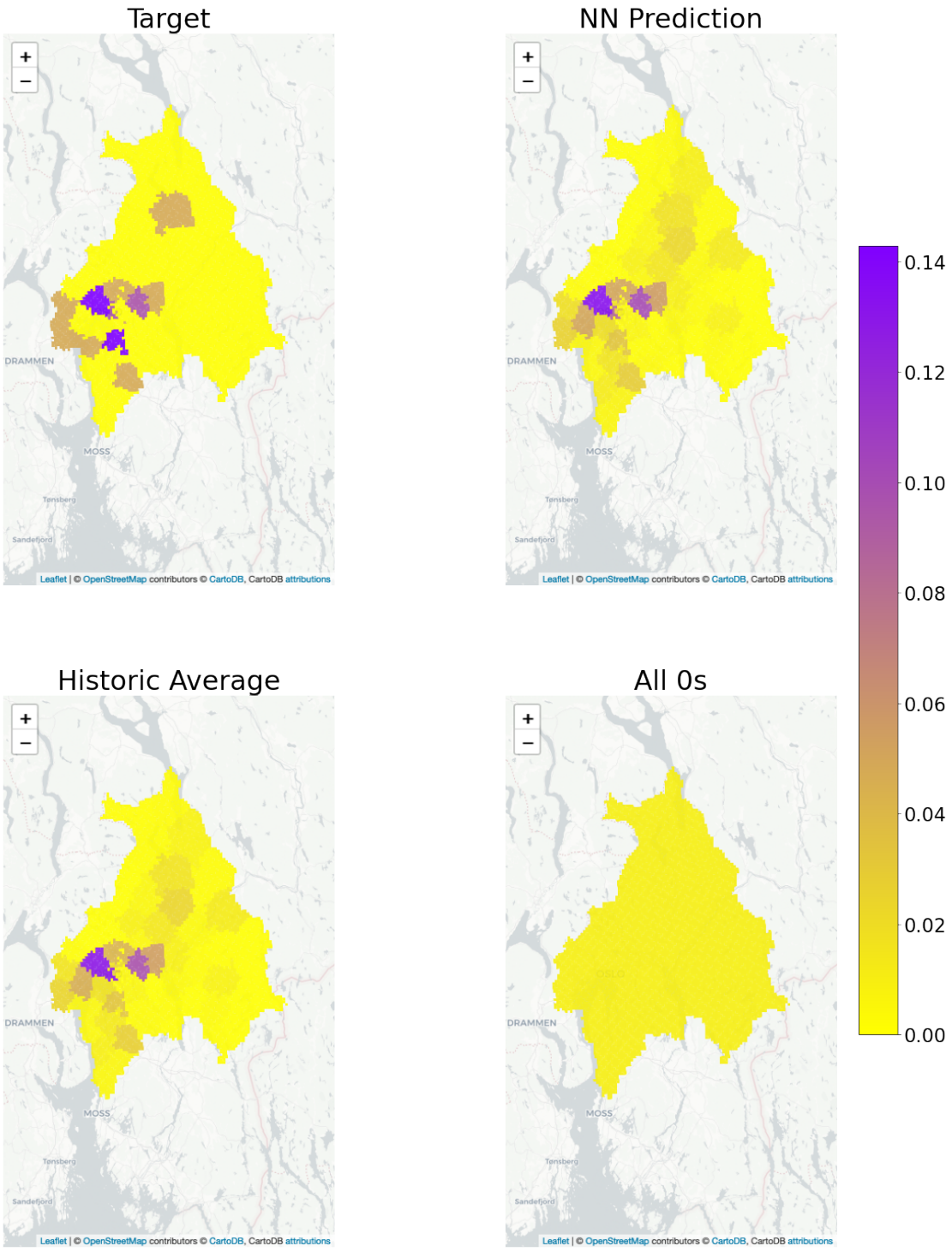

Figure A.10: Predicted distribution using segmentation v2.4 for the hour 2015-04-21 09:00 - 10:00.

Owren, O. & Van De Weijer, E.

Forecasting Ambulance Demand in Oslo and Akershus

NTNU
Norwegian University of
Science and Technology