Master's thesis
Master's Programme in Data Science

Ambient Drivers of Local Intermediate Ion Concentration in Boreal Environment

Mikko Ahro

May 20, 2025

Supervisor(s):	Professor Keijo Heljanko and Docent Ekaterina Ezhova
Examiner(s):	Professor Keijo Heljanko
	Docent Ekaterina Ezhova

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Degree programme Master's Programme in Data Science

Tekijä — Författare — Author

Mikko Ahro

Työn nimi — Arbetets titel — Title

Ambient Drivers of Local Intermediate Ion Concentration in Boreal Environment

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year May 20, 2025	Sivumäärä — Sidantal — Number of pages 59

Tiivistelmä — Referat — Abstract

This thesis examines the feasibility of predicting 2–2.3 nm negative ion concentrations using meteorological and environmental parameters from the Värriö SMEAR I station. The work focuses on building a reproducible data analysis pipeline and evaluating multiple machine learning models under different train-test splitting strategies.

The study evaluates a dataset including temperature, relative humidity, wind speed and direction, photosynthetically active radiation (PAR), friction velocity, and SO_2 concentration. A key finding was that rain events significantly distort negative ion measurements. Including these events led to a significant overestimation of model performance, and filtering them out was necessary.

Several modelling approaches were tested, including linear regression, polynomial regression, decision trees, random forest, gradient boosting (XGBoost and LightGBM), and neural network. Models trained on randomly split data produced promising results, but performance collapsed when time-aware splits were used. This dependency on the train-test split type indicates that models captured temporal proximity rather than causal relationships.

The analysis suggests that the available features are insufficient for predicting the target variable reliably. Feature engineering and transformation strategies provided only limited improvements. Recommendations for future work include adding total VOC measurements, higher time resolution, and further testing on data from other SMEAR stations.

ACM Computing Classification System (CCS):
Applied computing → Physical sciences and engineering → Earth and atmospheric sciences → Environmental sciences
Computing methodologies → Machine learning → Machine learning approachest
Mathematics of computing → Probability and statistics → Statistical paradigms → Regression analysis

Avainsanat — Nyckelord — Keywords

SMEAR station data, intermediate ions, machine learning, time series modelling, feature engineering

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Understanding how natural ecosystems influence atmospheric aerosol formation and concentration is essential for predicting the climate system and climate feedback mechanisms. One important mechanism is the formation of aerosol particles from biogenic emissions and their growth through atmospheric processes. Atmospheric aerosols are a key component of the Earth's climate system due to their role in scattering radiation and modulating cloud properties. Biogenic aerosols are formed mainly in forested environments either through direct emission or via gas-to-particle conversion driven by natural emissions and photochemical reactions [5]. The boreal region, in particular, has received considerable attention as a globally significant source of biogenic aerosol precursors [28]. Within this research area, the formation and growth of molecular clusters into climatically relevant particle sizes remain central questions. Various studies have identified intermediate ions as promising indicators of early-stage particle formation, bridging the gap between gas-phase chemistry and observable aerosol dynamics [19, 32].

Despite extensive research on aerosol formation, many studies have focused on regional new particle formation (NPF) events, where formation and continuous growth of aerosol particles to climatically relevant sizes is detectable [18, 8, 32]. However, intermediate ions, charged particles in the 2–7 nm size range, are also present outside such events and may provide insight into more continuous or background particle formation activity [13]. Systematic modeling including these background-level intermediate ion concentrations and aerosol bursts in relation to environmental variables remains limited, especially in high-latitude forest environments.

The study aims to model the dependence of intermediate ion concentration on various environmental features. These features include ambient temperature, relative humidity, wind speed and direction, friction velocity, SO_2 concentration, and photosynthetically active radiation. This study models the relationship between intermediate ion concentrations and selected environmental variables using Värriö SMEAR I station data. The analysis includes both statistical techniques and machine learning models, with particular attention to feature engineering and model validation.

The key research questions are:

- How do ambient environmental parameters influence local intermediate ion formation as inferred from negative ion concentration data and ecosystem observations?
- Can machine learning approaches reliably predict local intermediate ions under different environmental conditions?
- What machine learning methods and feature engineering options are most suitable for this data?

This study contributes to a more comprehensive understanding of aerosol dynamics under natural conditions by analyzing intermediate ion behavior beyond discrete formation events. The results can support the development of improved representations of particle formation processes in environmental models and provide reusable analytical tools for researchers working with SMEAR stations data or similar observational platforms.

The chapters in this thesis are organized as follows: Chapter 2 presents intermediate ions' role and formation mechanisms at a level relevant to this study. Chapter 3 outlines the dataset and details the statistical and machine learning methodologies employed. Chapter 4 presents the results of the statistical analyses and machine learning methods. Finally, chapter 5 summarizes the findings of this study.

2. Background

This chapter describes the impact of atmospheric aerosols on climate, the role of intermediate ions, and their formation mechanisms at a level relevant to this study. Aerosol formation and impact on climate are extremely complicated and widely researched subjects, and several other factors affect these besides those described below. Only the relevant features included in the dataset are discussed.

2.1 Atmospheric aerosols and climate effects

According to Kulmala et al. [27], forests can have both cooling and warming effects on the climate. A decreased surface albedo compared to grassland has a warming effect. Albedo refers to the reflectivity of a surface; lower albedo means less sunlight is reflected and more is absorbed, contributing to warming. On the other hand, forests act as a CO_2 sink due to photosynthesis. Forests also act as a source of aerosols by emitting biogenic volatile organic compounds (BVOCs), which contribute to the formation of secondary organic aerosols (SOA).

Atmospheric aerosols influence the Earth's radiative balance via multiple pathways. First, they act as cloud condensation nuclei (CCN), increasing cloud droplet number and cloud albedo: a process known as the indirect aerosol effect, which leads to cooling. Second, aerosols have a direct radiative effect by interacting with solar radiation through scattering and absorption. Scattering aerosols, such as sulfates and organics, contribute to cooling by reflecting solar radiation back to space, while absorbing aerosols contribute to warming by trapping solar radiation in the atmosphere.

Furthermore, aerosols increase the fraction of diffuse solar radiation, which can enhance forest photosynthesis. Ezhova et al. [10] found that increased aerosol loading in boreal and hemiboreal forests raised the diffuse fraction of radiation from approximately 0.11 to 0.27. This diffusion change led to a 6–14 % increase in gross primary production (GPP). The enhancement is attributed to enhanced light penetration into the canopy and increased light-use efficiency under diffuse conditions.

Overall, aerosol-climate interactions include warming and cooling components, with their net impact depending on aerosol type, ecosystem characteristics, and meteorological conditions.

2.2 Aerosol formation mechanisms

The formation of atmospheric aerosol particles proceeds through complex nucleation and growth mechanisms. This section highlights key gas-phase processes and environmental factors relevant to this study.

2.2.1 Sulfuric acid formation

The gas-phase oxidation of sulfur dioxide (SO_2) to sulfuric acid (H_2SO_4) proceeds as follows [30]:

$$\begin{aligned} SO_2 + OH + M &\longrightarrow HOSO_2 + M \\ HOSO_2 + O_2 &\longrightarrow HO_2 + SO_3 \\ SO_3 + H_2O &\longrightarrow H_2SO_4, \end{aligned} \tag{2.1}$$

where:

- M denotes a third body (usually N_2 or O_2) that stabilizes the intermediate species by absorbing excess energy.
- OH is the hydroxyl radical, the primary atmospheric oxidant.
- $HOSO_2$ is an intermediate radical formed during the oxidation of SO_2.

Reactions 2.1 are initiated by photo-oxidation, as the formation of OH radicals requires solar radiation.

2.2.2 Pathway from sulfuric acid to intermediate ions

Sulfuric acid can contribute to the formation of intermediate ions through clustering mechanisms, particularly when stabilized by other low-volatility vapors. One such pathway involves interactions between H_2SO_4 and highly oxygenated organic molecules (HOMs) [31].

In forest environments, these HOMs are formed from the oxidation of biogenic volatile organic compounds (VOCs), such as monoterpenes [31]. Their formation proceeds via photo-oxidation reactions, typically involving the hydroxyl radical (OH) or ozone (O_3) [30]:

$$\text{VOCs} + \text{OH}\,(\text{or } O_3) \longrightarrow \text{HOMs}. \tag{2.2}$$

2.2.3 Other environmental factors impacting intermediate ions formation

Relative humidity has multiple impacts on new particle formation [14], typically high RH inhibiting new particle formation. Higher humidity limits sulfuric acid concentration due to reduced hydroxy radical concentrations, so humidity impacts NPF early precursor formation. Also, coagulation scavenging of small ion clusters is increased with higher relative humidity. Scavenging refers to the removal of newly formed or nucleating clusters by collisions with larger pre-existing particles, which prevents them from growing further and contributing to new particle formation [8]. Condensation sink also increases because of the hygroscopic growth of preexisting particles. The condensation sink quantifies how efficiently pre-existing aerosol particles remove condensable vapors from the atmosphere, thereby suppressing new particle formation [8].

Turbulence is highly relevant for new particle formation [38], due to several proposed mechanisms: mixing of nucleation precursor gases across air layers, dilution of condensation sink by cleaner air, and adiabatic cooling in updrafts boosting nucleation rate. Also, one additional possible effect is that photochemically induced particle formation has started in the air layer, not vapor-rich enough to continue to significant particle growth; if such a layer with new particles is mixed with a vapor-rich layer, this will enhance particle growth. Similar mechanisms are also studied by Wehner et al. [47], with more explicit pointers towards the relevance of mechanical turbulence or shear. This study quantifies turbulence using friction velocity, which reflects mainly mechanical mixing processes such as shear, but is also influenced by buoyancy-driven turbulence under strong surface heating.

Negative ion concentration is influenced by air mass transport. Wind speed and direction affect the spatial distribution of intermediate ions source, and therefore wind speed and direction are highly relevant factors [44].

Atmospheric temperature is also a key factor in new particle formation [25], but the direction and magnitude of the effect remain unclear. One reason may be that temperature influences both NPF-inhibiting and NPF-enhancing processes. For example, higher temperatures increase the emissions of BVOCs exponentially, enhancing aerosol precursors' availability and promoting particle formation [12]. At the same time, higher temperatures reduce the stability of molecular clusters, making it less likely for nucleation to proceed [2]. It may also increase the condensation sink due to enhanced growth of pre-existing particles.

2.2.4 Intermediate ions as indicators of local particle formation

Intermediate ions, typically defined as charged particles in the 2–7 nm size range, are a strong indicator of atmospheric new particle formation (NPF) events [32, 18]. Their formation coincides with that of larger particles, giving them indirect climate relevance.

Tuovinen et al. [45] evaluated the suitability of using atmospheric ion concentrations to identify and quantify local intermediate ion formation (LIIF) as a proxy for NPF. They argue that forming intermediate ions, charged particles with diameters roughly between 2 and 7 nm, is necessary for NPF to proceed, as particle growth beyond approximately 2 nm is required to ensure particle survival.

Ions smaller than 2 nm are present continuously due to ionization by cosmic rays and natural radioactivity, resulting in high background concentrations that mask growth-related signals. In contrast, intermediate ions only appear in elevated concentrations during active formation events, making them more specific indicators of NPF.

Tuovinen et al. [45] further show that the size range of 2.0–2.3 nm is optimal for identifying LIIF. This range is large enough to reflect early growth from molecular clusters, but small enough that the ions are likely formed locally, within 500 m to 1 km of the measurement site, minimizing confounding effects from regional transport. Negative ion concentrations in this range were more sensitive to LIIF than positive ions, due to stronger contrasts between background and event periods.

As a result, the concentration of 2.0–2.3 nm negative ions measured by a neutral cluster and air ion spectrometer (NAIS) can be used as a robust indicator of local particle formation activity under varying environmental conditions.

2.2.5 Intermediate ions and variability in formation conditions

Earlier long-term observations by Hõrrak et al. [13] support the use of intermediate ions as indicators of particle formation. They identified bursts of intermediate ions that coincided with new particle formation and found correlations with environmental parameters such as temperature and relative humidity. However, these variables alone did not fully explain the observed variability, pointing to additional, unidentified factors influencing the process. Subsequent study by Tammet et al. [43] used the high-sensitivity symmetric inclined grid mobility analyzer (SIGMA) to show that intermediate ions are not limited to burst events but are also present during quiet periods. This persistent background concentration suggests continuous low-level nucleation that does not always develop into detectable growth events.

These findings indicate that a modeling approach focusing only on distinct bursts may miss a significant portion of the particle formation process. Instead, treating intermediate ion concentrations in the 2.0–2.3 nm size range as a continuous variable can help capture both prominent and subtle nucleation activity. This approach supports using data-driven models trained on complete time series data to better represent the influence of ambient conditions on early-stage particle formation.

3. Materials and methods

This chapter presents the dataset and the methods used in this study. It begins with an overview of the measurement site and variables included in the dataset, followed by data preprocessing, transformations, and normalization descriptions. Statistical techniques and several machine learning models are introduced, along with the evaluation and validation strategies used to assess model performance.

3.1 Dataset

Värriö measurement station [16], situated at 67°46′ N and 29°35′ E in eastern Lapland [40], is far from roads and pollution sources. The nearest small road lies approximately 8 km away, and the closest major road is around 100 km from the station. The nearest major anthropogenic pollution sources are metallurgy facilities located 150 km east and 190 km north of the site. Therefore, Värriö station is an ideal location to research local background aerosol formation. The station measurement tower is 390 m above sea level, and the major tree species surrounding the station are Scots pines. The dataset consisted of target variable 2–2.3 nm negative ion concentration, and features: wind speed and direction, SO_2 concentration, air temperature, relative humidity, and friction velocity. Rain intensity was added later, due to observations described in Section 4.4. The details of measurement parameters are described in Table 3.1. The period studied was 2019–2024, the summer months of July and August. The target variable was recorded as 30-minute averages; thus, other parameters were queried as 30-minute averages. As a total, the dataset had 17568 data points.

Based on earlier research, all parameters in the dataset are relevant to negative ion concentration as described in chapter 2.

3.1.1 Description of measurement methods

This study's environmental and meteorological data were collected from standard instrumentation at the Värriö station. These include wind speed and direction, air temperature, relative humidity, photosynthetically active radiation (PAR), rain inten-

name	description	unit	source
WS00	Wind speed at 16 m	m/s	Thies Ultrasonic Anemometer 2D
PAR	Photosynthetically active radiation	$\frac{\mu\text{mol}}{m^2 s}$	Li-Cor LI-190SB quantum sensor
SO2_1	SO_2 concentration at 9 m	ppb	TEI 43 CTL Pulsed Fluorescence analyzer
TDRY0	Air temperature at 15 m	°C	Rotronic MP103
WDIR	Wind direction at 16 m	°	Thies Ultrasonic Anemometer 2D
rainint	Rain intensity	mm/h	Thies Laser Disdrometer
RH0	Relative humidity at 15 m height	%	Rotronic MP103A
u_star	Friction velocity, tower at 16.6 m height	m/s	Metek USA-1 anemometer/thermometer
2-2.3nm	Negative ions concentration	ions/cm^3	Airel NAIS spectrometer

Table 3.1: Physical measurements for Värriö dataset.

sity, friction velocity (u_*), and sulfur dioxide (SO_2) concentration. All variables were measured using established commercial sensors and analyzers commonly used in atmospheric monitoring networks. Details on sensor types and measurement heights are listed in Table 3.1, and comprehensive metadata is available through the SMEAR data portal [22].

Neutral cluster and air ion spectrometer

The neutral cluster and air ion spectrometer (NAIS) is an instrument for monitoring atmospheric nanometer-sized aerosol particles [33]. NAIS can measure the size distribution of both ions and neutral particles in the size range of approximately 2–40 nm. The device features a dual-column design, with 21 electrometer channels per polarity, enabling simultaneous measurement of positive and negative ions.

NAIS operates in two modes. It detects naturally charged air ions in *ion mode*. In *particle mode*, neutral particles are charged by a corona charger in the sample inlet, allowing their detection alongside ions.

The instrument does not directly measure particle size. Instead, it records particle electrical mobility, a function of the sample flow, applied voltage, electrode geometry, and the particle's electrical charge [34]. Since electrical mobility depends on particle size, the instrument can infer size distributions by converting mobility spectra to size spectra. This conversion is achieved through a matrix inversion technique that maps signals from 21 physical electrometer channels into 27 partially overlapping mobility fractions, each corresponding to a specific size range, while accounting for the instrument response.

NAIS's key advantages include high time resolution, sensitivity, and suitability for long-term field measurements in diverse environments. Its primary limitation is sensitivity to background noise, particularly in the smallest size channels [33].

3.2 Data pipeline and tools

The data processing was done using Python as the primary programming language. The main Python libraries and software tools are listed in Table 3.2. Environmental variables were downloaded from the AVAA Smart SMEAR API using a public Python module from the University of Helsinki [26]. The module implements helper functions for accessing and formatting time series data and metadata from SMEAR stations.

The pipeline was managed using Hydra, a lightweight configuration management library for Python applications [50]. Hydra enabled modular control over pipeline components and experiment configurations. It enabled structured management of transformation, normalization, and hyperparameter options through its built-in sweeper, which allows automatic generation and execution of multiple experiment combinations based on hierarchical configuration files.

Library	**Version**	**Purpose**
Python	3.12.2	Programming language
Conda	24.5.0	Dependencies management
JupyterLab	4.3.4	Interactive notebooks and analysis
NumPy	1.26.4	Array and mathematical operations
Pandas	2.2.3	Dataframe manipulation
Matplotlib	3.10.0	Plotting
Seaborn	0.13.2	Statistical plots
Scikit-learn	1.6.1	ML models and preprocessing
LightGBM	4.5.0	LightGBM model
XGBoost	2.1.4	XGBoost model
Statsmodels	0.14.4	Statistical modeling (e.g., VIF)
Hydra	1.3.2	Experiment configuration and combinations management
OmegaConf	2.3.0	Hydra configuration backend
MLflow	2.20.2	Experiment tracking and logging

Table 3.2: Main tools and libraries used.

Target variable 2–2.3 nm ion concentration was available as a CSV-file, and the rest of the parameters were downloadable from Smart SMEAR API [22]. The data loading and merging process is presented in Figure 3.1.

After data loading, the dataset was fed into a pipeline controlled by Hydra. The pipeline was divided into two parts: data preparation, which included selecting the data range, dropping unnecessary columns, missing value processing, filtering out rain

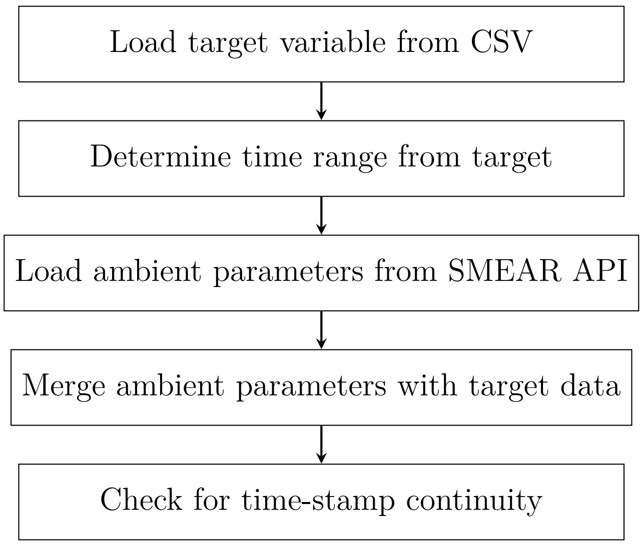

Load target variable from CSV
Determine time range from target
Load ambient parameters from SMEAR API
Merge ambient parameters with target data
Check for time-stamp continuity

Figure 3.1: Data loading.

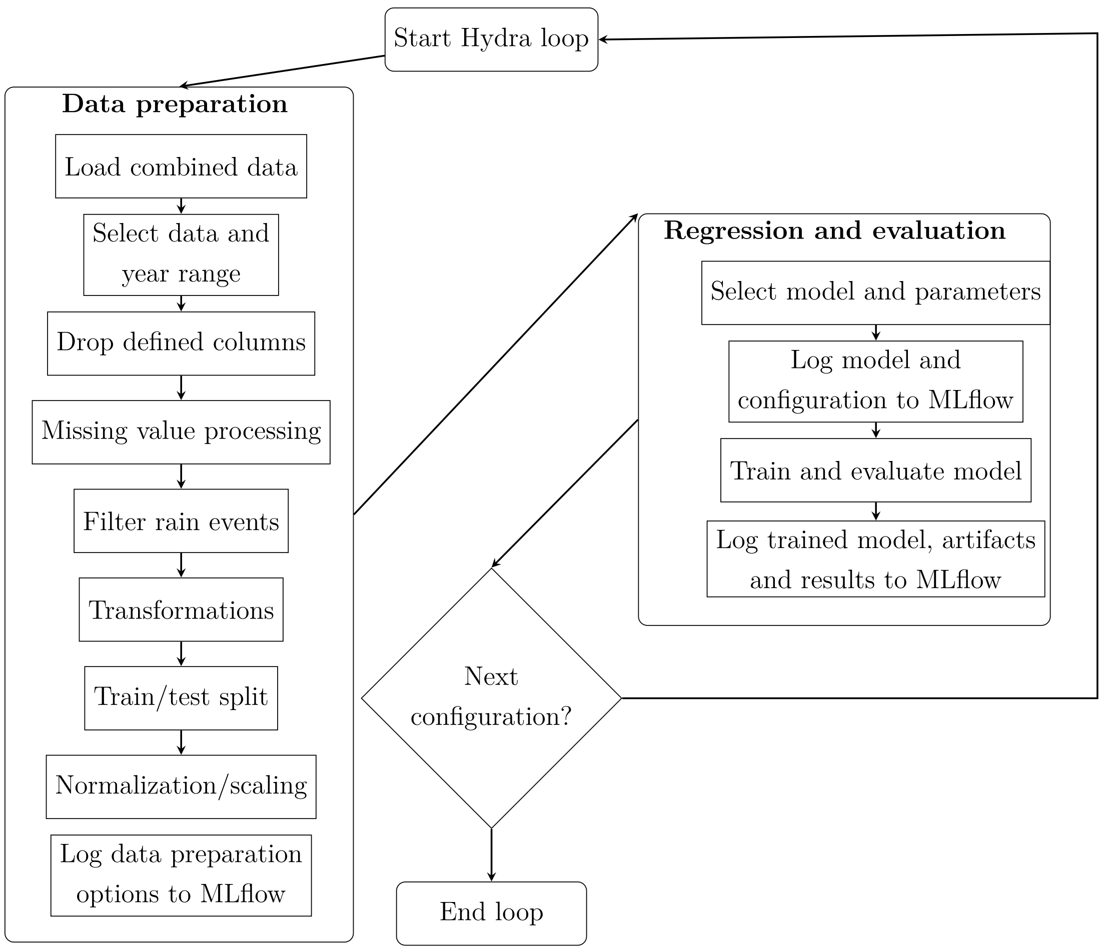


Figure 3.2: Data processing and regression pipeline.

events, transformations, train-test split, normalization and scaling, and logging data preparation options to MLflow. The second part of the pipeline was regression and evaluation, which included selecting a model with hyperparameters, logging model configuration to MLflow, training and evaluating the model, and logging results into MLflow. The pipeline details are described in Figure 3.2.

3.3 Transformations and normalizations

Water vapor concentration (calculated from RH and temperature) was added as a feature, as it offers a more robust representation than relative humidity and implicitly captures the interaction between humidity and temperature. Water vapor concentration in volume-% was calculated from relative humidity by calculating saturated water vapor pressure e_w using two alternative formulas: the Tetens and Goff-Gratch formulas [23]. Tetens formula:

$$e_w = 0.61078 \exp \frac{17.27t}{t + 273.3}, \tag{3.1}$$

where t is temperature in °C. Goff-Gratch formula for logarithmic saturated water vapor pressure is:

$$\begin{aligned} \log_{10} e_w = & -7.90298 \left(\frac{373.16}{T} - 1 \right) \\ & + 5.02808 \log_{10} \left(\frac{373.16}{T} \right) \\ & - 1.3816 \times 10^{-7} \left(10^{11.344\left(1 - \frac{T}{373.16}\right)} - 1 \right) \\ & + 8.1328 \times 10^{-3} \left(10^{-3.49149\left(\frac{373.16}{T} - 1\right)} - 1 \right) \\ & + \log_{10}(1013.246), \end{aligned} \tag{3.2}$$

where T is the temperature in Kelvin. The saturated water vapor pressure in hPa is converted into concentration in volume-%:

$$C_{H2O} = \frac{\text{RH}}{100} \cdot e_w \cdot \frac{1}{P_{\text{total}}} \cdot 100, \tag{3.3}$$

where RH is relative humidity and P_{total} is normal pressure 1013.25 hPa.

Wind measurement was represented by two parameters in the dataset: wind speed v (m/s) and wind direction θ as an angle (degrees). Angular features are unsuitable for machine learning or statistical modelling, so wind was converted into a more usable form. Three alternative transformations were used. Wind was transformed into x- and y-components:

$$\text{wind}_x = v \cdot \cos(\theta) \tag{3.4}$$

$$\text{wind}_y = v \cdot \sin(\theta). \tag{3.5}$$

As a second alternative, the wind direction was converted into sine- and cosine-components, and the wind speed was kept unchanged:

$$\text{wind}_{\text{cos}} = \cos(\theta) \tag{3.6}$$

$$\text{wind}_{\text{sin}} = \sin(\theta). \tag{3.7}$$

Wind was converted into x- and y-components as a third alternative, and wind speed was kept as an additional feature.

Time components were added as features by taking the time of day as minutes from data timestamps. Also, days from period start (i.e., the 1st day of each summer was day 1) was added as a feature. Time features, as cyclical features, were transformed into sine- and cosine-features:

$$\text{time}_{\text{cos}} = \cos\left(\frac{2\pi \cdot \text{minutes}}{1440}\right) \tag{3.8}$$

$$\text{time}_{\text{sin}} = \sin\left(\frac{2\pi \cdot \text{minutes}}{1440}\right). \tag{3.9}$$

The time of day was always transformed. Days from period start were used as a linear feature or similarly transformed into cyclical form:

$$\text{days}_{\text{cos}} = \cos\left(\frac{2\pi \cdot \text{minutes}}{\text{period length}}\right) \tag{3.10}$$

$$\text{days}_{\text{sin}} = \sin\left(\frac{2\pi \cdot \text{minutes}}{\text{period length}}\right), \tag{3.11}$$

where period length was 61 days for Värriö (1st July - 31st August).

Min-max normalization maps a value v_i to new scaled value v_i' according to range of feature in dataset [15]:

$$v_i' = \frac{v_i - \min_A}{\max_A}(\max - \min) + \min, \tag{3.12}$$

where $\min_A$ and $\max_A$ are the minimum and maximum values of the feature in the dataset, and min and max are the minimum and maximum of the scaled range. Min-max scaling is not necessary for most machine learning methods, but was used with a neural network to scale relative humidity to range 0–1 from the original 0–100 %.

Z-score normalization or standard scaling is used to normalize the values towards normal distribution [15]:

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A}, \tag{3.13}$$

where $\bar{A}$ is the mean of feature A in the dataset and σ_A is the standard deviation of feature A in the dataset.

Logarithmic transformation can be used to compensate for extremely skewed data. In this case, the skewed variables were extremely zero-skewed (target variable negative ion concentration and SO_2 concentration, and hence $\log(1+x)$ transformation was used:

$$v' = \log(1 + v). \tag{3.14}$$

3.4 Statistical analysis methods

In this section, the term predictor is used in line with statistical modeling terminology, though it corresponds conceptually to feature in the machine learning sections.

The principal measure of fit between the target variable and the features in a given model is the coefficient of determination, R^2, defined as [21]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{3.15}$$

Correlation between two random variables is defined as [46]:

$$\rho(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}, \tag{3.16}$$

where covariance is defined as:

$$\mathrm{Cov}(X, Y) = \mathbb{E}\left[(X - \mu_X)(Y - \mu_Y)\right]. \tag{3.17}$$

The variance inflation factor (VIF) is a measure used to quantify multicollinearity between features [21]. For each feature $\hat{\beta}_j$, it is defined as:

$$\mathrm{VIF}(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}, \tag{3.18}$$

where X_j is the predictor of interest, X_{-j} denotes all other predictors, and $R^2_{X_j|X_{-j}}$ is the coefficient of determination from regressing X_j on all other predictors. Interpretation of VIF values depends on context, but a typical rule of thumb is summarized in Table 3.3.

VIF	Interpretation
1	No correlation with other variables(ideal case).
1–5	Low to moderate multicollinearity; generally acceptable.
5–10	Moderate to high multicollinearity; may need attention depending on the context and modeling goals.
>10	Severe multicollinearity; problematic and likely to affect regression estimates significantly.
inf	Perfect collinearity; indicates a variable is a linear combination of other predictors.

Table 3.3: Interpretation of variance inflation factors.

3.5 Cross-validation

Cross-validation is a model evaluation strategy that involves repeated training and validation to estimate generalization performance on unseen data. In this study, k-fold cross-validation was used, where the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is partitioned into k disjoint and approximately equal-sized subsets $\mathcal{D}_1, \dots, \mathcal{D}_k$. For each fold $j = 1, \dots, k$, the model is trained on the union of all other folds $\mathcal{D}_{-j} = \bigcup_{i \neq j} \mathcal{D}_i$ and validated on $\mathcal{D}_j$. The model's performance is averaged over all k validation rounds:

$$R^2_{\text{CV}} = \frac{1}{k} \sum_{j=1}^{k} R^2_j.$$

This approach provides a more robust estimate of model performance compared to a single train-test split, as it reduces the variance associated with a particular partitioning of the data [21, 17]. It is especially useful for model selection and hyperparameter tuning.

3.6 Machine learning methods

3.6.1 Linear and polynomial regression

The most basic regression method is linear regression. In linear regression, a linear relationship is assumed between the target variable and features [21]:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon, \tag{3.19}$$

where ε is the error term. Thus, the optimization task is to find coefficients $\beta_0, \dots, \beta_p$ so that the residual sum of squares (RSS) is minimized:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{3.20}$$

Polynomial regression extends linear regression by including higher-degree terms of features, allowing for nonlinear relationships:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \cdots + \beta_d X^d + \varepsilon. \tag{3.21}$$

For multiple features, the equation is:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \sum_{j=1}^{p} \sum_{k=1}^{d} \beta_{jk} X_j^k + \varepsilon. \tag{3.22}$$

And in case interaction terms between various features are included, the model expands to:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \sum_{j=1}^{p} \sum_{k=2}^{d} \beta_{jk} X_j^k + \sum_{j=1}^{p} \sum_{m=j+1}^{p} \beta_{jm} X_j X_m + \varepsilon. \tag{3.23}$$

The optimization problem remains the same in all extended cases: finding coefficients β such that RSS is minimized.

A significant advantage of linear and polynomial regression is interpretability: The model provides explicit feature weights, allowing insight into how features influence predictions. The disadvantages of linear and polynomial regression include that linear and low-degree polynomial regression cannot model very complex relationships. Higher-degree polynomial regression increases computational complexity and is prone to overfitting, capturing noise rather than underlying patterns. Overfitting can be mitigated using regularization techniques like Lasso (L1 penalty) and Ridge (L2 penalty), which constrain the magnitude of coefficients [21].

3.6.2 Decision tree

A decision tree is a recursive method that divides the multidimensional feature space into non-overlapping regions. The split is done over various dimensions recursively, and each split grows the decision tree. The tree has a root node representing the full dataset, internal decision nodes corresponding to split conditions, and terminal leaf nodes providing the final output [41]. An illustrative example of a two-dimensional dataset split and resulting decision tree is presented in Figure 3.3.

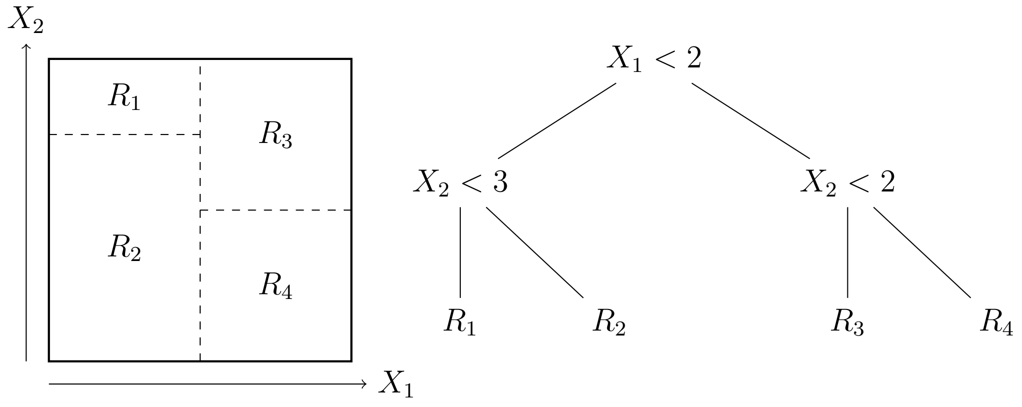


Figure 3.3: Example of a 2D decision tree split. The left side shows the feature space divided into regions, while the right side represents the corresponding decision tree.

Recursion continues until the predefined stop condition, for example, maximum tree depth, minimum number of samples per node, and minimum number of samples required to split. Data are split into J distinct regions $R_1, R_2, ...R_J$, and for every region R_i, the prediction for regression trees is the mean of the training target variable in that region, and for classification trees, the majority class in the region. Therefore, the optimization problem is to find such region splits that RSS is minimized [21]:

$$\mathrm{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2. \tag{3.24}$$

In practice, however, if all possible splits are considered, the optimization problem is computationally too expensive. Therefore, practical decision tree implementations rely on a greedy top-down algorithm. The split of predictor space is done based on criteria that RSS is minimized at that split without considering further splits down the tree [21].

$$\mathrm{RSS}(j, s) = \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2. \tag{3.25}$$

The above-described process leads, in practice, to too complex trees with too detailed split rules, which consequently means overfitting. Tree pruning is used [36] for better generalization. A more common method is post-pruning, where a large tree is first grown, and then the split rules are tested by predicting holdout data. Rules contributing little or no performance improvement are removed, reducing tree complexity and improving generalization. Another alternative for pruning is pre-pruning, where splitting stops when further splits do not provide sufficient information gain.

One main advantage of decision trees is due to the non-parametric nature of the method [41]: decision trees require no assumptions on distributions. On the other hand, the significant disadvantage of decision trees is a tendency to overfit, capturing noise rather than true patterns. Even though decision tree regressors are seldom optimal for practical problems, their basic theory is an essential building block for state-of-the-art tree-based regressor methods.

3.6.3 Random forest

The inherent overfitting tendency of decision trees can be reduced by bagging trees. Bagging reduces variance by training multiple decision trees on different bootstrapped samples from the training data [3]. The final prediction of bagged trees is the mean of the predictions of individual trees.

While bagging is effective in reducing variance, it remains a problem that all bagged trees are highly correlated because they are built from the same training data. The positive pairwise correlation for identically distributed, but not necessarily independent random variables is given by [17]:

$$\mathrm{Var}(\bar{Z}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{3.26}$$

where:

- $\bar{Z}$ is the mean of B identically distributed but correlated random variables.
- ρ is the pairwise correlation.
- σ^2 is the variance of each random variable.

As $B \to \infty$, the second term vanishes, and the variance approaches $\rho\sigma^2$. This means that if trees are highly correlated, variance reduction is limited.

Random forest is designed to cope with high variance problems associated with bagged trees [4]. Random forest draws bootstrapped samples from training data in the same manner as bagged trees. Instead of considering all p features for each split, Random forest randomly selects a subset of m features ($m < p$) and chooses the best split only among them [17]. This is recursively repeated until the stop condition. By limiting the number of features available for each split, random forest effectively decorrelates the resulting trees, reducing the variance of the ensemble predictor.

The advantage of random forests over decision trees and bagged trees is reduced variance and, hence, reduced overfitting tendency, especially if the number of features is large. However, random forests are not beneficial if only a few predictors exist. Also, random forests may perform suboptimally when there are many features but only a few relevant ones; the probability of randomly picking relevant features is low, leading to poor prediction performance.

3.6.4 Boosting

Boosting is an ensemble learning method designed to enhance the predictive performance of weak learners, typically decision trees [21]. Boosting builds trees sequentially, unlike bagging, where trees are trained independently on bootstrap samples. Each new tree is trained to correct the residual errors of the previous trees, allowing the model to learn iteratively from past mistakes.

The boosting process follows a slow-learning approach to prevent overfitting. At each step, a tree is trained on the residuals of the previous model, and a scaled version of this new tree is added to the overall prediction. This iterative refinement continues until a predefined number of trees is reached.

The general boosting algorithm has three key hyperparameters [21]:

1. Number of trees (B): Unlike bagging, boosting can overfit if too many trees are used, though overfitting tends to occur gradually. Limiting the number of trees can reduce the tendency to overfit.

2. Shrinkage parameter (λ): A small positive value (e.g., 0.01 or 0.001) that controls the learning rate, ensuring incremental improvements and reducing overfitting.

3. Tree depth (d): Defines the complexity of each tree. A common choice is $d = 1$, meaning each tree is a simple decision stump, leading to an additive model structure.

This iterative framework allows boosting to capture complex patterns in the data while maintaining a balance between flexibility and generalization.

Gradient boosting

Gradient boosting is a boosting algorithm that improves stability and mitigates overfitting issues associated with general boosting [37]. Unlike standard boosting, which directly fits new trees to the residuals, gradient boosting finds the most efficient correction direction by computing residuals as the negative gradient of the loss function:

$$\nabla L = \frac{\partial L(y, F)}{\partial F}, \tag{3.27}$$

where:

- $L(y, F)$ is the chosen loss function that measures prediction error.
- y is the true target value.
- $F(x)$ is the current ensemble model estimate at iteration t, combining all previously fitted trees.

By taking the negative gradient, gradient boosting determines the steepest descent direction in loss space, ensuring the most optimal step for updating the model in the next iteration. This approach results in more efficient error correction and better generalization than standard boosting.

The gradient boosting algorithm was first introduced by Friedman [11] and is presented in Algorithm 1.

Algorithm 1 Friedman's Gradient Boosting Algorithm

```
Require: Training data $(x_i, y_i)_{i=1}^N$, number of iterations $M$, loss function $L(y, f)$, base
    learner $h(x, \theta)$, parameters of base learner $\theta$
Ensure: Trained model $F_M(x)$
1: Initialize $F_0(x)$ as a constant:
       $F_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$
2: for $t = 1$ to $M$ do
3:     Compute the negative gradient (pseudo-residuals):
           $g_t(x_i) = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{t-1}(x)}$
4:     Fit a new base learner to the residuals:
           $h_t(x, \theta_t) \approx g_t(x)$
5:     Compute the optimal step size:
           $\rho_t = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{t-1}(x_i) + \rho h_t(x_i, \theta_t))$
6:     Update the model:
           $F_t(x) = F_{t-1}(x) + \rho_t h_t(x, \theta_t)$
7: end for
```

3.6.5 XGBoost

A highly efficient and popular tree-based ensemble method is XGBoost [6]. The important features of XGBoost can be roughly classified into three categories: 1) result-optimizing features, 2) computing cost-optimizing features, and 3) hybrid features optimizing both accuracy and efficiency.

Result-optimizing features include adding regularization; original XGBoost included L2 regularization, but modern versions have added L1 regularization as well [49]. The second-order gradient is used in addition to the first-order gradient for faster convergence. XGBoost includes a sparsity-aware split-finding algorithm that efficiently handles missing and sparse data. It automatically learns a default direction for missing and non-informative zero values, reducing unnecessary splits.

Computing cost-optimizing features includes the approximate algorithm for split finding. While XGBoost, by default, uses the exact greedy algorithm for split finding, the exact greedy algorithm requires sorting of the data. Sorting is computationally expensive in distributed settings, so the approximate algorithm for split finding reduces this cost by using candidate split points. XGBoost stores data in a compressed column (CSC) format instead of a row-based format to allow effective parallelization. This allows accessing specific feature values across all samples, computing splits for individual features in parallel, and reducing memory consumption. This data format is beneficial for both single-machine setups, where it allows cache-aware access and speeds up split finding, and for distributed systems, which is essential for enabling parallel computation. XGBoost has also been further developed. CUDA-based graphics processor unit (GPU) support has been added [35].

There are also hybrid features that improve both accuracy and computing efficiency. XGBoost grows trees depth-first and prunes weak branches during training based on gain thresholds. Compared to traditional tree boosting, this method improves generalization and reduces computational costs by avoiding unnecessary splits. Pruning weak branches lowers overfitting risks and helps reduce computational costs by avoiding unnecessary branches.

3.6.6 LightGBM

LightGBM [24] is another tree-based ensemble method based on gradient boosting. LightGBM was developed to address the scalability limitations of XGBoost and other tree-based gradient boosting methods. Despite multiple optimizations implemented in XGBoost, the limitation is that XGBoost still needs to scan all data instances to estimate gains for all possible split points, which is slow when the feature space and data are large. LightGBM addresses these limitations with two key innovations: gradient-based on-side sampling (GOSS) to reduce data size and exclusive feature bundling (EFB) to reduce the number of features.

GOSS takes further advantage of gradients rather than just determining the most efficient direction: GOSS calculates and sorts gradients for all data instances. Data instances are divided into two categories based on their gradients and are processed differently. Instances with large gradients are sampled fully, but instances with small gradients are only randomly sampled. The distribution change due to random sampling is balanced by weight factors given to low-gradient instances.

The EFB algorithm addresses a sparse feature space in which many features are mutually exclusive, such as when categorical features are one-hot encoded. EFB is a specific algorithm for identifying and merging mutually exclusive features. The resulting feature space compression significantly reduces the training speed and memory consumption without sacrificing predictive power.

Additionally, LightGBM uses a histogram-based algorithm to determine split points instead of finding exact ones. The histogram-based algorithm bundles features into bins, significantly reducing the split point comparisons as only bin boundaries need to be compared.

Like XGBoost, LightGBM includes several engineering optimizations, such as parallelization and cache-aware access.

Overall, LightGBM can speed up training time by a factor of 20 compared to XGBoost, with only a slight loss of accuracy.

3.6.7 Feature importance measures in tree-based methods

One significant advantage of decision trees is interpretability; the tree resulting from training can be visualized and is almost intuitively interpretable. As more advanced tree-based methods use bagging and boosting, this interpretability is reduced. When the resulting model is an ensemble of several trees, the contribution of individual features cannot be easily visualized or tabulated for feature-wise interpretability [21].

Feature importance can, however, be calculated based on the contribution of each feature to the model's performance. In practical implementations like XGBoost and LightGBM, feature importance is typically assessed using one of the following alternative metrics [1]:

- Gain is the total reduction in the loss function, such as residual sum of squares or log-loss, attributable to a feature across all splits where that feature is used.
- Split count is the number of times a feature is used in splits across all trees.
- Cover is the number of samples affected by splits based on that feature.

This provides a relative ranking of feature importance. However, it should be noted that these measures reflect the contribution to predictive power but do not indicate the direction of a feature's effect on the target variable. It is also noticeable that feature importance is a different concept from correlation. A feature may have a strong linear or nonlinear relationship with the target but still be deemed less important if its predictive value is redundant, or if the model does not benefit from using it in splits. Conversely, features with low correlation may be significant if they reduce residual error in key regions of the input space.

3.6.8 Neural networks

Artificial neural networks (ANNs) are computational models inspired by the structure and function of the human brain [7]. They consist of an input layer, one or more hidden layers, and an output layer. The layers are composed of nodes (neurons) connected through weighted links. Each input is multiplied by a weight, and the weighted sum is passed through an activation function, determining whether the neuron is activated. The general structure of a neural network is illustrated in Figure 3.4.

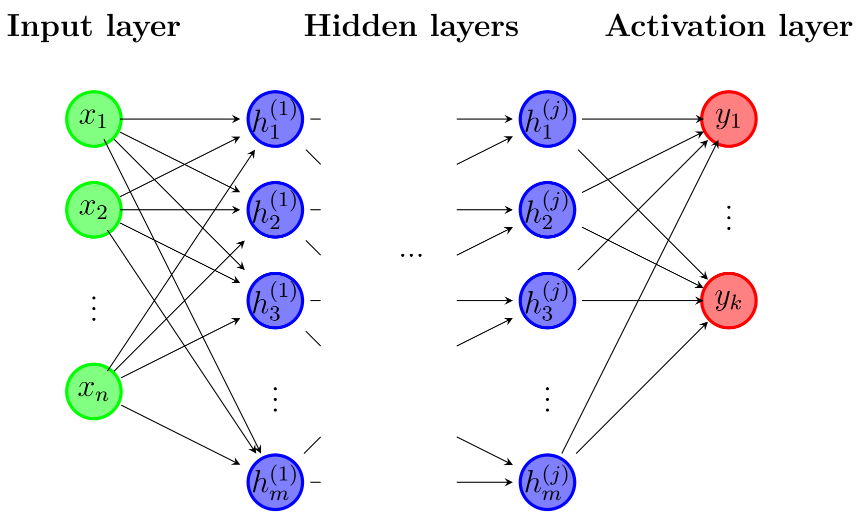


Figure 3.4: Basic structure of a neural network.

Mathematically, the input to a hidden layer neuron h_i is calculated as a weighted sum of the inputs x_j [9]:

$$h_i = f\left(\sum_{j=1}^{n} w_{ij} x_j\right), \tag{3.28}$$

where w_{ij} are the connection weights and f is the activation function. A simple activation function is the step function:

$$f(x) = \begin{cases} 0, & \text{if } x < \theta \\ 1, & \text{if } x \geq \theta, \end{cases} \tag{3.29}$$

where θ is the threshold value. More commonly, smooth activation functions like the sigmoid function are used:

$$f(x) = \frac{1}{1 + e^{-\frac{x-\theta}{T}}}, \tag{3.30}$$

where T is a smoothing parameter.

Training a neural network involves adjusting the weights using a learning algorithm, such as backpropagation. In supervised learning, the network is provided with input-output pairs, and the error between the predicted and true output is calculated. The weights are updated iteratively based on this error until the network output matches the expected result with sufficient accuracy.

Neural networks are powerful for capturing complex patterns and nonlinear relationships, making them highly effective in image and speech recognition tasks. However, they require large datasets and computational resources for training, and the training process can be prone to overfitting if not properly regularized.

3.7 Validation and evaluation strategy

Model evaluation was based on a structured train-test split strategy and k-fold cross-validation. The choice of split method was critical, as experiments showed that model performance varied substantially depending on how the data was partitioned. All data splits and experiment configurations were managed using Hydra, and model metrics were logged to MLflow.

Four types of train-test splits were used:

- Random split: the dataset was split using scikit-learn's `train_test_split` with `shuffle=True`, meaning that training and test samples were selected randomly from the entire dataset. This approach does not respect time ordering and serves as a performance upper bound.

- Regular split: a non-shuffled `train_test_split` with `shuffle=False`, resulting in a chronologically ordered split where the first fraction of the dataset is used for training and the last portion for testing. This reflects a more realistic temporal separation.

- Day-based split: the data was grouped by full calendar days, and GroupKFold was applied using day identifiers as groups. This ensures that all samples from the same day fall entirely in either the training or test set, preventing leakage due to partial day overlap.

- Year-based split: one full year was used as the test set, while the remaining years were used for training. This setup was used to assess model generalization across seasonal and interannual variation.

A summary of the split types used for training-test data is visualized in Figure 3.5.

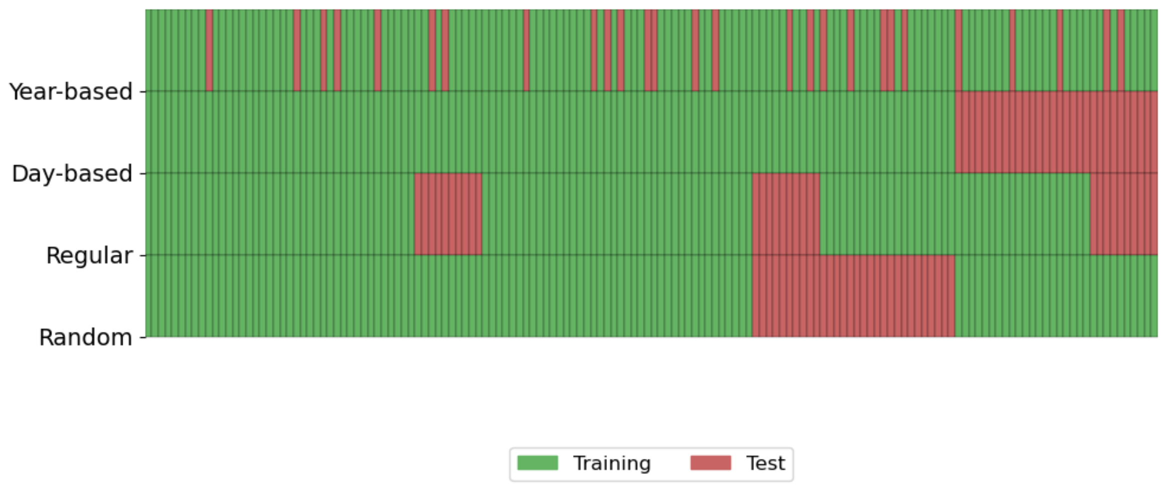


Figure 3.5: Illustrative example of split types. Note that in the real data set, there are 48 data points per day, 61 days per two-month period for each year, and six years. The number of data points has been scaled lower than in the actual data for presentation purposes. From top: random split means the test set is selected as totally random data points from the whole dataset. Non-randomized or regular split means that the test data is from the end of the dataset. Day-based split means the train-test split is random, but complete days are kept together. A year-based split means that one year (between 2019 and 2024) is selected as the test set, and the rest are used as training data.

k-fold cross-validation was used within the training set to assess generalization and guide model selection. In all experiments, the number of folds was set to $k = 5$, a commonly used default that balances bias and variance in error estimation. Two modes were applied:

- Standard *k*-fold: training data was randomly partitioned into *k* equally sized folds.

- TimeSeriesSplit: folds respected temporal order by successively expanding the training window and using subsequent observations for validation. This was applied by setting `use_time_series_cv=True` in the configuration.

In all cases, performance was evaluated using the coefficient of determination (R^2), computed for both the cross-validation folds (CV R^2) and the final held-out test set (validation R^2). These metrics evaluated model generalization under different data continuity and temporal structure assumptions. CV R^2 reflects model performance on subsets of the training data and is used for model selection, while validation R^2 measures performance on a completely unseen test set. In time-aware splits, these values can differ substantially, as temporal patterns seen during training may not generalize to the held-out period.

The splits were implemented using `train_test_split`, `GroupKFold`, and `TimeSeriesSplit` functions from scikit-learn [39].

4. Results and discussion

4.1 Dataset cleaning and filtering

In dataset preparation, timestamp continuity was checked for years 2019-2024. Environmental data downloaded from SmartSMEAR had continuous timestamps, while a separate target variable was missing 48 timestamps. Merging of data frames resulted in continuous timestamps with missing target variable values replaced with NaN values. This amount of missing target data was not deemed to be critical.

The next step was to check missing values in the dataset. A total of 1133 datapoints had one or more missing values out of 17568, which corresponds to 0.6 % of the total data and was not seen as critical. Missing data is plotted as a heatmap in Figure 4.1. Missing periods are mostly scattered small fractions. Only friction velocity has a more extended missing period, at 2024 data is missing from August 22 to August 31, a total of 476 data points, approximately 16 % of the 2024 summer period.

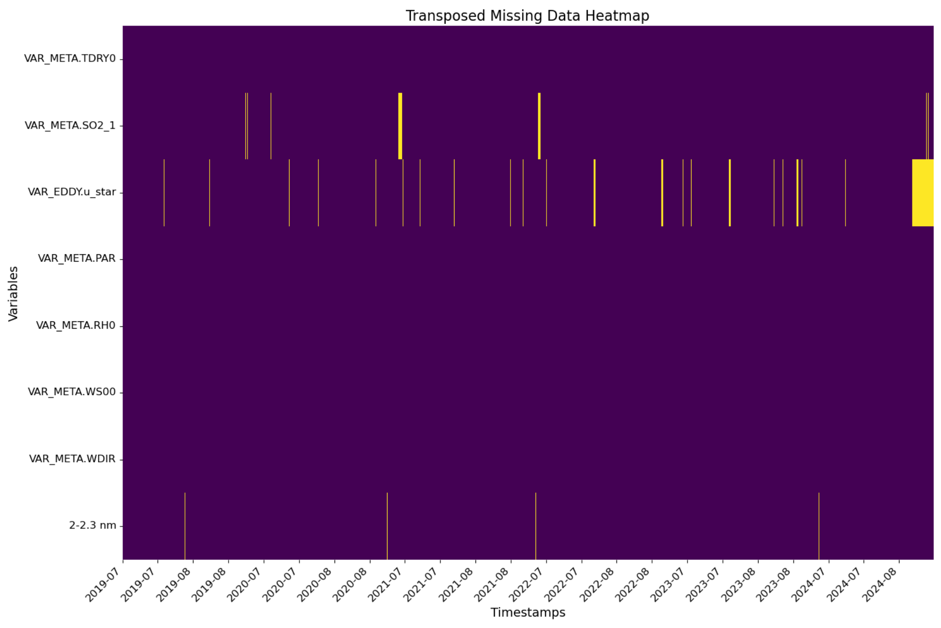


Figure 4.1: Missing data heatmap for Värriö summer periods

4.2 Exploratory analysis

In the exploratory analysis phase, the pairwise correlation between variables was investigated. The results are presented in Figure 4.2.

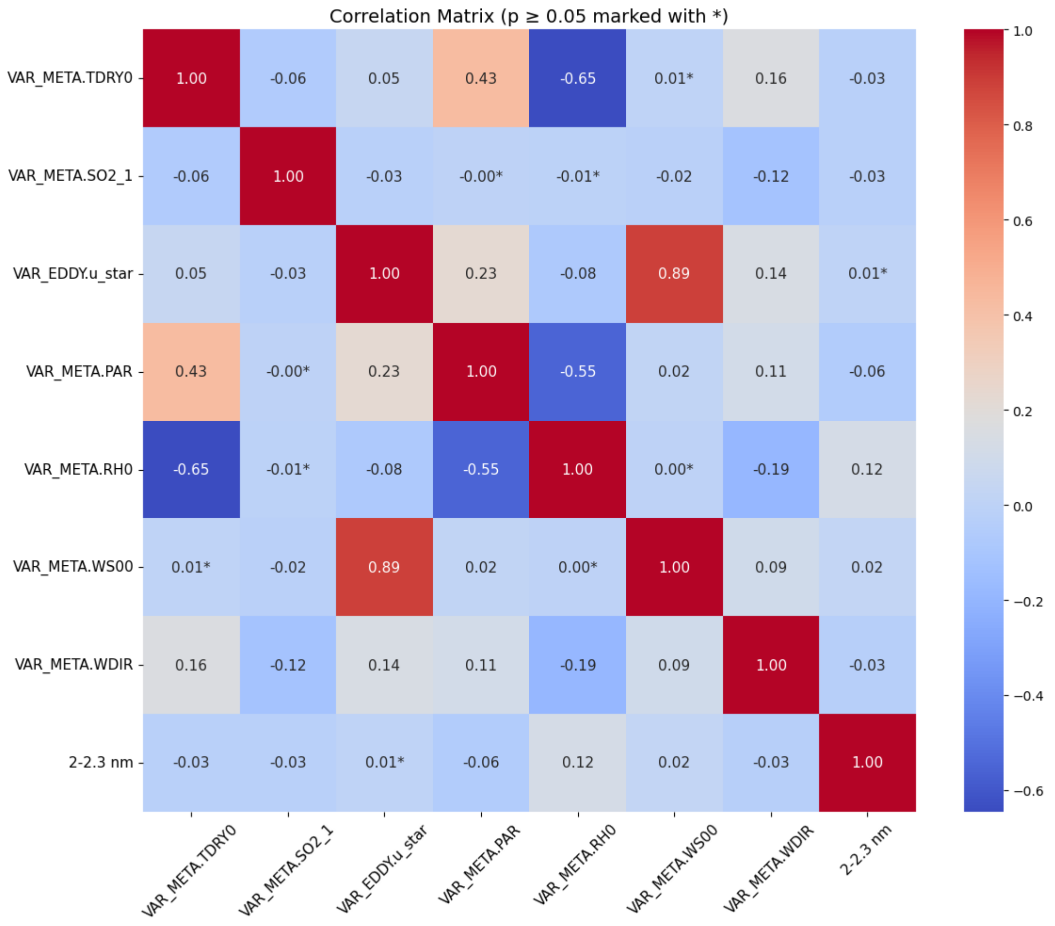


Figure 4.2: Pairwise variable correlations in Värriö dataset. Asterisks (*) indicate pairwise correlations with p-values ≥ 0.05, suggesting statisitically non-significant correlations at the 5 % level.

A few high correlations observed in the data can be explained by known physical mechanisms. For instance, temperature and photosynthetically active radiation (PAR) correlate positively. Relative humidity (RH) is negatively correlated with temperature due to diurnal boundary layer dynamics, where surface warming deepens the boundary layer and mixes in drier air. RH also negatively correlates with solar radiation, reflecting their opposing diurnal cycles. Due to shear- and buoyancy-driven turbulence, friction velocity positively correlates with both wind speed and solar radiation. Naturally, wind speed and friction velocity are also positively correlated. No other variables showed strong pairwise correlations in this dataset.

Variance inflation factors between features were also calculated; the results are presented in Table 4.1. There were a few high (>5) and extremely high (>10) variable inflation factors, which was not surprising, taking into account the high pairwise correlations. Notably, there were no infinite variable inflation factors, indicating that the dataset did not contain redundant features.

Variable	VIF
VAR_META.TDRY0	7.65
VAR_META.SO2_1	1.04
VAR_EDDY.u_star	38.58
VAR_META.PAR	2.91
VAR_META.RH0	7.22
VAR_META.WS00	43.60
VAR_META.WDIR	5.73

Table 4.1: Variable inflation factors for Värriö

Distributions of features and target were investigated with the aid of histogram plots. The distributions are presented in Figure 4.3. The distributions of ambient temperature, friction velocity, and windspeed were close to normal. SO_2 concentration and the target variable were extremely zero-skewed. Solar radiation was zero-skewed, and relative humidity was skewed toward 100 %. Wind direction seemed relatively uniformly distributed, except northern winds were rare.

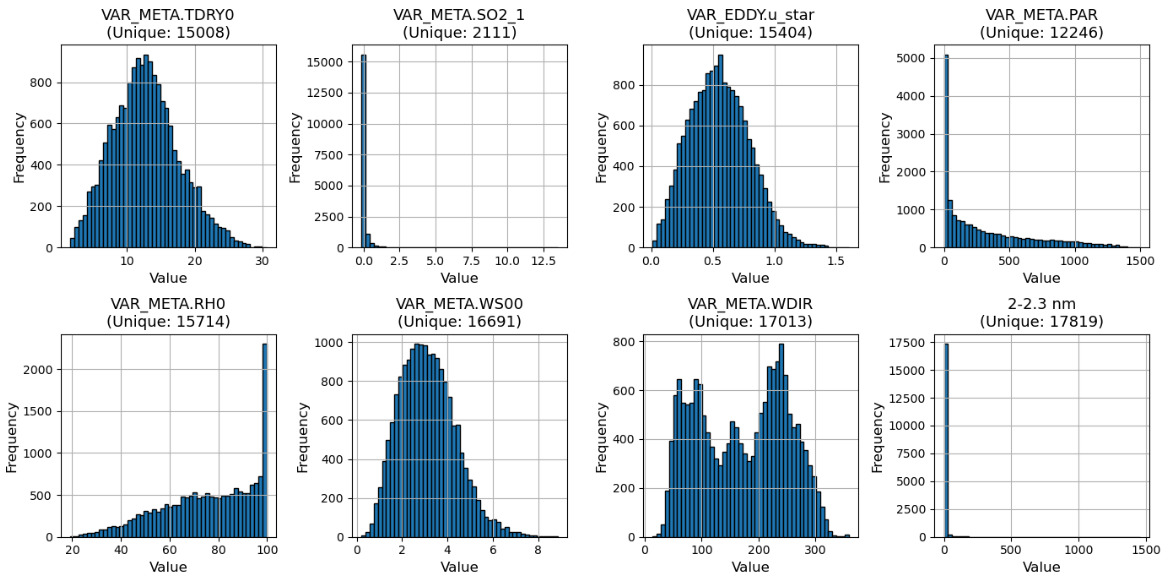


Figure 4.3: Distributions of features and target in Värriö datasete

4.3 Baseline models

As a baseline comparison, the following machine learning models were evaluated:

- Linear regression
- Polynomial regression ($d = 2$)
- Decision tree regressor
- Random forest regressor
- XGBoost regressor
- LightGBM regressor
- Multilayer perceptron regressor

The dataset used in this test was all Värriö data 2019–2024, July-August. All parameters listed in Table 3.1 were used in the test, except rain intensity, which was added later. A randomized train-test split was used.

Each model was lightly optimized for hyperparameters, transformations, and normalizations. The configurations used in the final comparison are listed in Table 4.2

model	wind	time components	H_2O	log(1+x)	standard scaling
Linear regression	xy	cos-sin	tetens	SO_2, target	-
Polynomial regression	xy	cos-sin	tetens	SO_2, target	-
Decision tree	xy	cos-sin	tetens	SO_2, target	-
Random forest	xy	cos-sin	tetens	SO_2, target	-
XGBoost	xy	cos-sin	tetens	SO_2, target	-
LightGBM	xy	cos-sin	tetens	SO_2, target	-
MLPRegressor	xy	cos-sin	tetens	SO_2, target	T, u_star, PAR, RH, H_2O, windX, windY

Table 4.2: Configurations used in model comparison

Linear, polynomial regression, and decision tree regressors did not reach good R^2 scores. Polynomial regression was also tested up to 4th degree polynomials, but 3th and 4th degrees did not improve the model. In fact, training scores remained poor, so applying regularization was not deemed worthwhile. Random forest, XGBoost, LightGBM, and MLP regressors all achieved over 50 % R^2 scores. The results for various models are summarized in Figure 4.4.

Tree-based ensemble methods XGBoost and LightGBM performed best and were virtually identical. Based on these results, it was decided to continue with LightGBM; its speed made it more suitable for testing various combinations related to feature engineering, transformations, and normalizations.

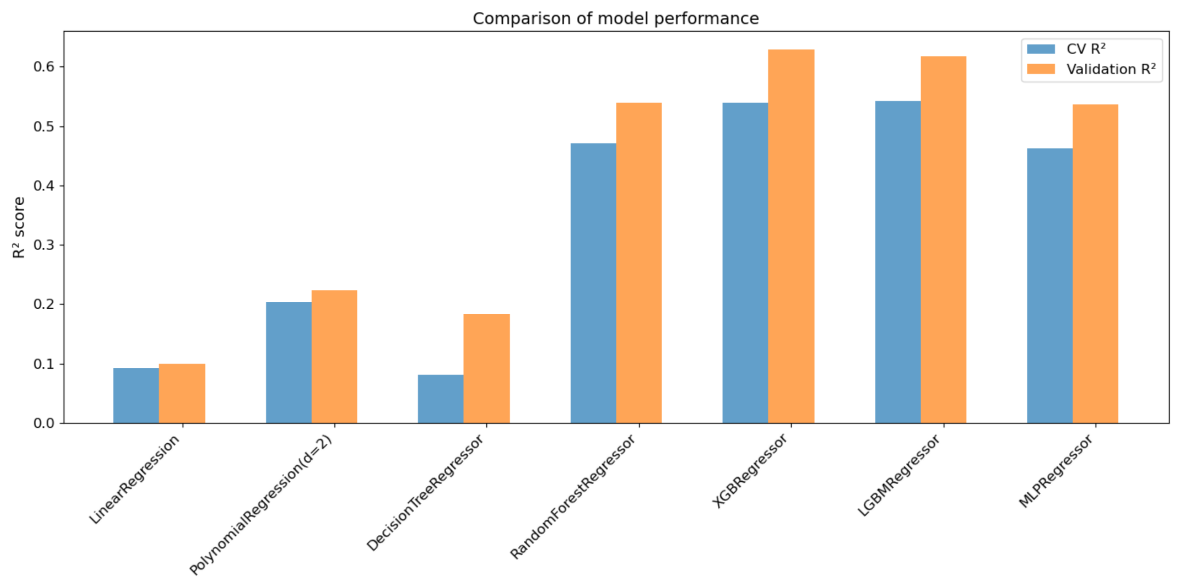


Figure 4.4: Comparison of various models.

4.4 Impact of rain events

Rain intensity was added to the dataset because liquid-phase water may affect aerosol formation in various ways, such as scavenging. Relative humidity was part of the original feature set. Still, relative humidity does not reliably indicate whether it rains or not, and in particular, it does not convey any information about rain intensity.

Adding rain intensity surprisingly affected model performance; the model CV R2 increased from 0.54 to 0.79, and test R2 increased from 0.63 to 0.79. The unexpectedly large performance improvement from including a single feature warranted further investigation. As a first step, pairwise correlations were calculated between target, relative humidity, and rain intensity. These showed an extremely high correlation between target and rain intensity. The pairwise correlations between the target variable, rain intensity, and relative humidity are shown in Figure 4.5.

This correlation was further validated by a scatter plot with a trendline between the target variable and rain intensity, as shown in Figure 4.6. Even though the data mass was heavily centered towards zero rain intensity and the data was sparse for higher rain intensities, the dependence was clear.

Rain intensity was further classified. The categories were determined according to the World Meteorological Organization scale [48], except that no rain category limit was determined based on the estimated rain measurement noise level, as shown in Table 4.3. The target variable was plotted as a box plot in different rain categories in Figure 4.7. Even though the bin counts for moderate ($n = 147$) and heavy ($n = 12$) were low, the phenomenon is evident: negative ion concentration increased by orders

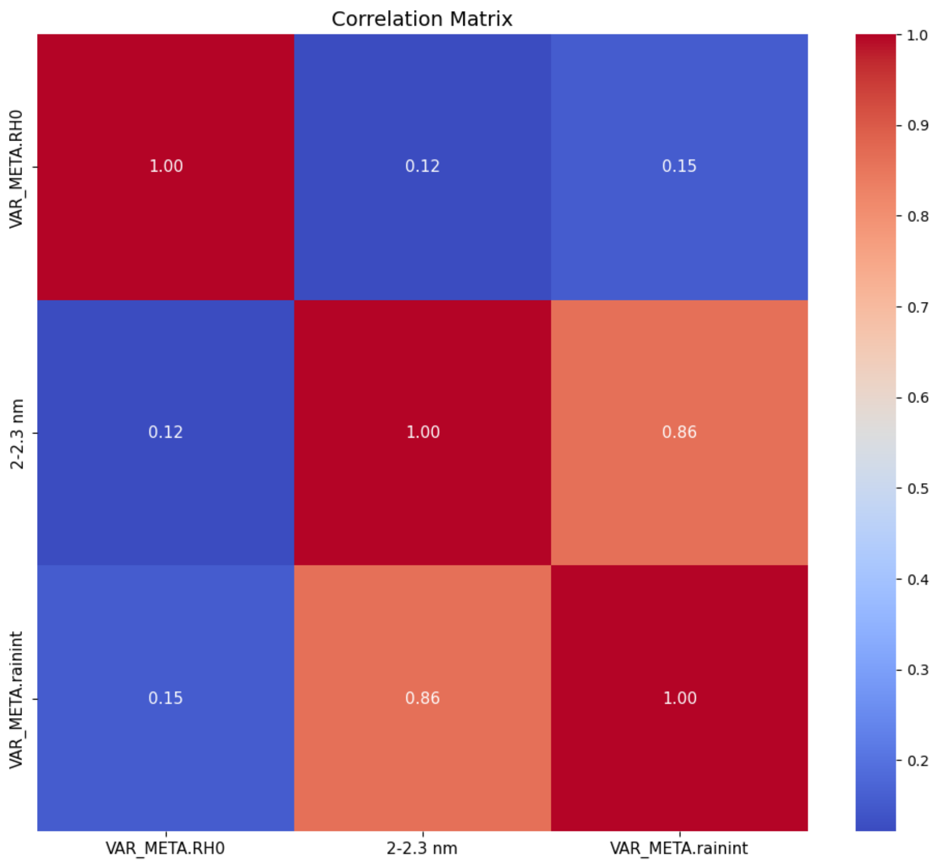


Figure 4.5: Correlation matrix for Värriö dataset for target variable, relative humidity and rain intensity. All correlations were statistically significant (p < 0.05).

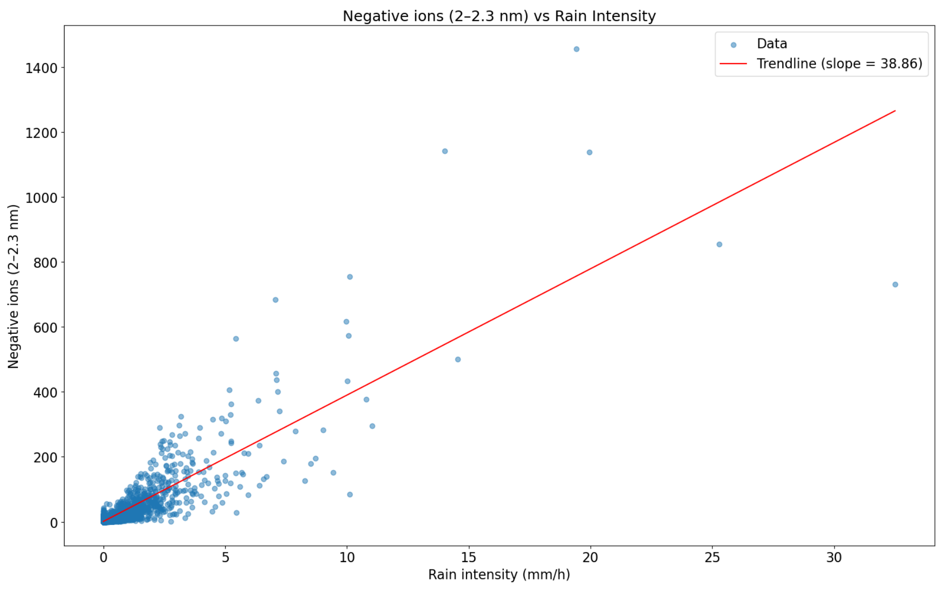


Figure 4.6: Target variable vs. rain intensity as scatter plot and trend line.

of magnitude when rain intensity increased.

Rain intensity (mm/h)	Category
≤ 0.01	No rain
> 0.01 and ≤ 2.5	Slight rain
> 2.5 and ≤ 10	Moderate rain
> 10 and ≤ 50	Heavy rain
> 50	Violent rain

Table 4.3: Rain intensity classification.

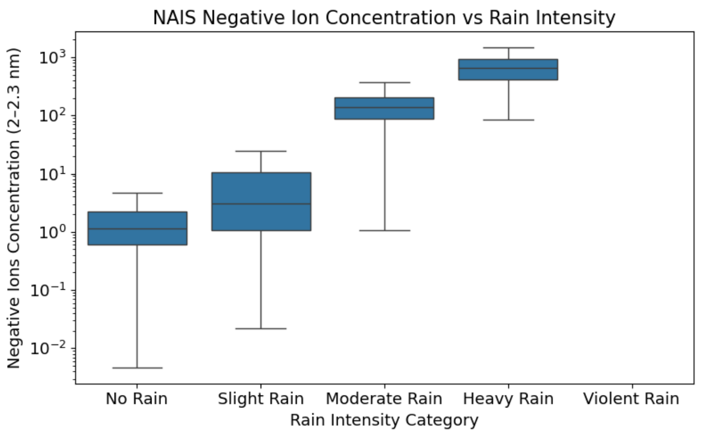


Figure 4.7: Box plots for target variable distributions in various rain categories for Värriö dataset.

To investigate the temporal effect of rain intensity on target variable measurement, negative ion concentration correlation with rain intensity and time-lagged rain intensity was plotted in Figure 4.8. The results show that the effect of rain vanishes very quickly. The fact that the correlation did not drop to zero immediately could be explained by rain events lasting typically more than one 30-minute sampling period.

The temporal closeness of the target variable and rain intensity was also shown by plotting five randomly selected days in categories no rain, slight rain, moderate rain, and heavy rain. The days were chosen by maximum rain intensity during the day. These example days are presented in Figure 4.9. The plots showed that during heavy and moderate rain, the target variable peaks follow rain intensity peaks without a delay or lag effect. During slight rain days, the target peaks also followed rain intensity, although the strength of the response varied. On dry days, there were naturally only very low rain intensities detected (<0.01 mm/h based on earlier definition), and therefore there was no similar trend.

The results showed clearly that the target variable was strongly impacted by rain intensity. The impact was not delayed or lagged, but the target variable peaked im-

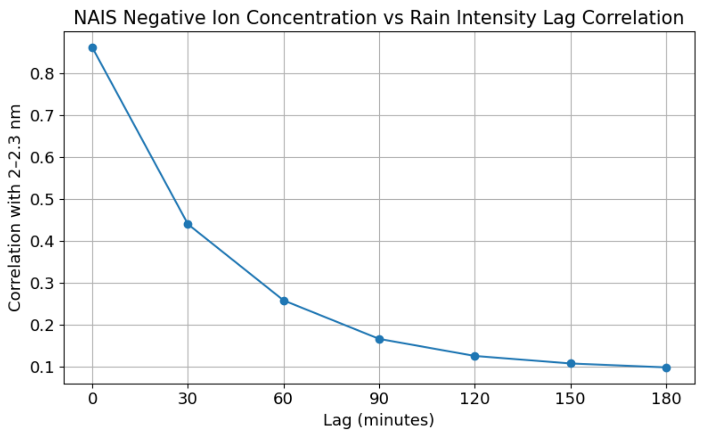


Figure 4.8: Decay of rain intensity impact to target variable in Värriö dataset.

mediately when rain intensity increased, and similarly dropped immediately when rain intensity dropped. Further, the target variable measurements were orders of magnitude higher during rain events, significantly impacting any machine learning model being applied. Further literature review showed that this same effect has been noted in other studies related to rain events [20], measurements close to waterfalls [29]. The effect has been verified in a combined laboratory and field study [42], but the exact physical reason remains unclear. Regardless of the cause, the conclusion of the study is clear [42]: water splash-related negative ion concentration peaks should not be treated as nucleation bursts.

Regardless of the reason for ionization during rain events, the rain events were filtered out from the data in further analysis. The filtering level was set at rain intensity 0.01 mm/h. It could have been possible to adjust the filtering level slightly higher. This conservative threshold was chosen because it discarded only a small amount of data, while significantly reducing the risk associated with rain-impacted target values. Figure 4.10 illustrates the threshold and impact of filtering. After removing rain events, the model CV R^2 was 0.60 and validation R^2 was 0.65, slightly higher than before using rain intensity as a feature.

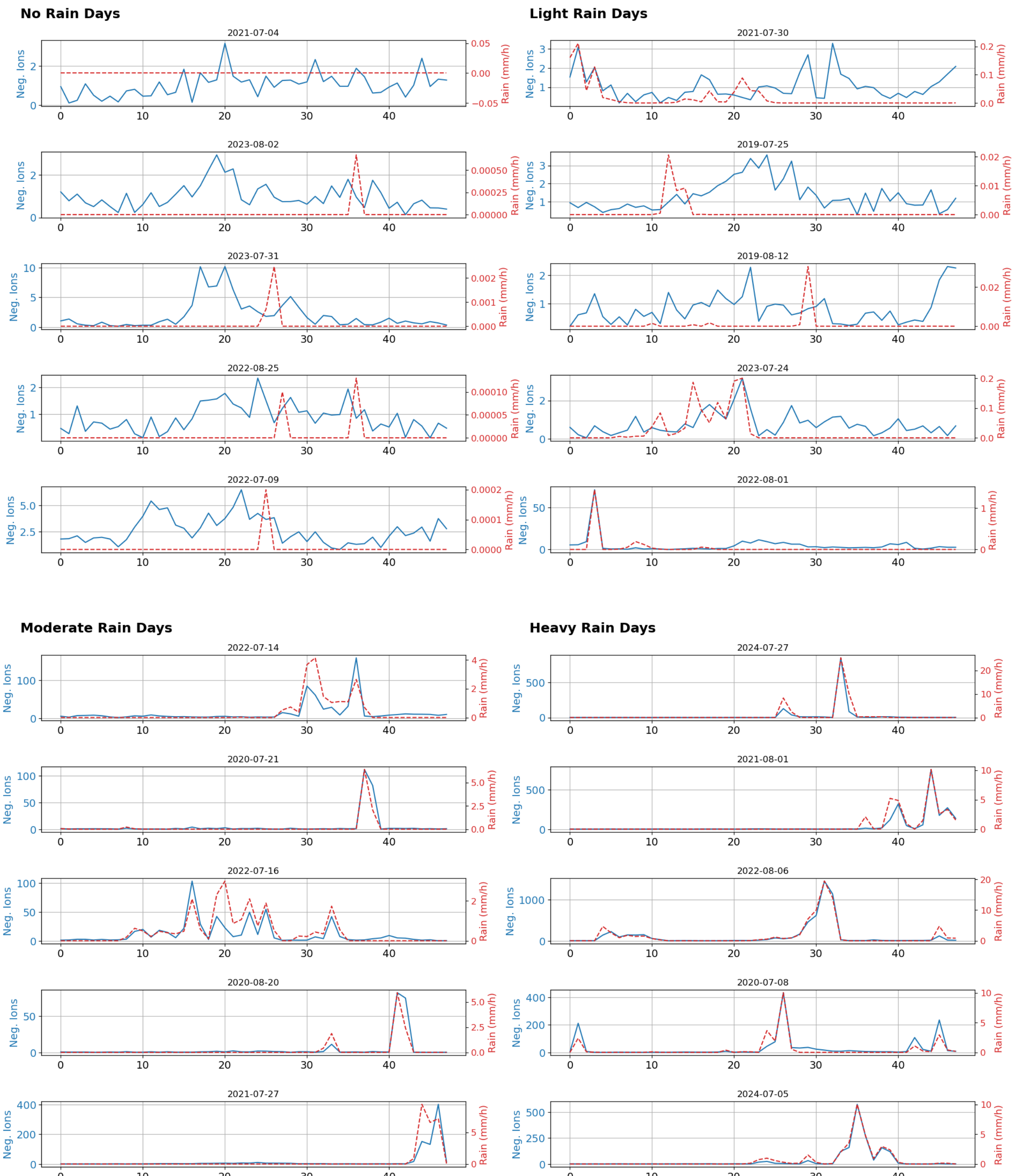


Figure 4.9: Target variable and rain intensity plots for five randomly selected days in each rain category for Värriö dataset.

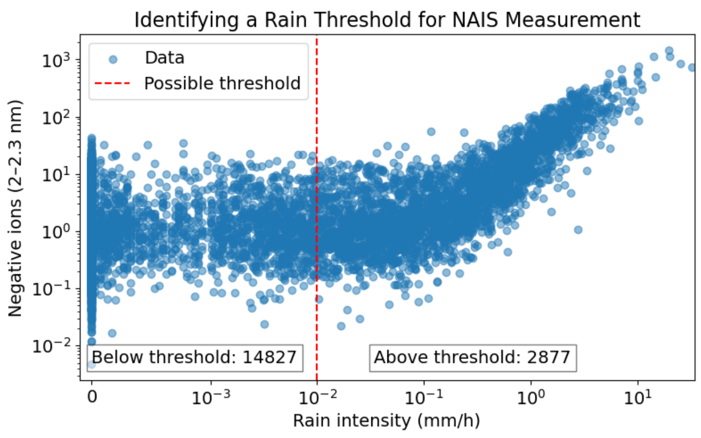


Figure 4.10: Identifying rain filtering level threshold.

4.5 Applying various transformation options

Various transformations and their combinations were tested systematically in this experiment. The transformations evaluated were:

- Water vapor concentration was added using two alternative methods: the Tetens and Goff-Gratch formulations.
- Time components were added, including time of day and days from period start; the time of day was always represented using sine and cosine transformations.
- Periods with low solar radiation were filtered out using a threshold of 20 $\frac{\mu\text{mol}}{m^2 s}$.
- Wind variables were transformed using three alternative representations, as detailed below.
  - Wind direction was transformed into sine and cosine components.
  - Wind was expressed as x- and y-components.
  - Wind was expressed as x- and y-components with wind speed included as an additional feature.

The best combination, which yielded CV R^2 of 0.60 and validation R^2 of 0.65, consisted of:

- The water vapor concentration was added using the Goff-Gratch method.
- Time features were included using the sine and cosine representation.
- Solar radiation filtering was not applied.

- Wind data were transformed into x- and y-components.

The impact of each transformation is summarized in Table 4.4. The most significant improvement was observed from adding time features, which increased validation R^2 by 0.16. Adding H_2O concentration also had a positive effect. Specific transformation methods for time features, H_2O, and wind had smaller but still measurable effects. Interestingly, filtering out solar radiation consistently hurt model performance.

Parameter	Option	Mean Validation R^2
Add H_2O Feature	Yes	0.560
Add H_2O Feature	No	0.538
Add Time Features	Yes	0.606
Add Time Features	No	0.447
H_2O Calculation Method	Goff-Gratch	0.560
H_2O Calculation Method	Tetens	0.560
PAR Filter	No	0.580
PAR Filter	Yes	0.526
Time Encoding	Cyclic	0.615
Time Encoding	Linear	0.597
Wind Representation	XY	0.558
Wind Representation	XY + Speed	0.552
Wind Representation	Cos-Sin	0.549

Table 4.4: Mean impact of transformations on validation R^2.

4.6 Time series analysis

A test using the LightGBM model was conducted to evaluate whether a generalizable empirical model could predict ion concentrations in any given summer, assuming environmental predictor data are available. The previously identified optimal transformation options were applied, but the train/test split strategy was modified: instead of a random split, one full year was selected as the test set. This process was repeated for each year from 2019 to 2024.

Although training on future data to predict the past is not valid for real-time forecasting, the goal here was not operational prediction but rather to test whether a proxy-type model could capture relationships robust enough to generalize across years. The results are shown in Table 4.5. The model's performance collapsed when evaluated

Validation set	Split by year	CV R^2	Validation R^2
2024	True	0.0646	0.1890
2023	True	0.0486	0.1178
2022	True	0.2042	-0.1896
2021	True	0.0713	0.2231
2020	True	0.0635	0.0750
2019	True	0.0695	0.1267
Random 20%	False	0.5999	0.6546

Table 4.5: Impact of splitting train and test data by year.

on a held-out year, suggesting that the trained models failed to generalize beyond their training period.

To investigate the cause of this performance degradation further, each year was modeled independently. The results of this experiment are presented in Table 4.6. When modeled separately, each year yielded good results, comparable to those obtained with a randomly split complete dataset.

Year	Test n	Train n	CV R^2	Validation R^2
2024	437	1746	0.6104	0.5968
2023	465	1858	0.5732	0.6354
2022	458	1831	0.7126	0.7516
2021	468	1868	0.6258	0.6539
2020	495	1979	0.5702	0.6132
2019	480	1918	0.6356	0.6262

Table 4.6: Model performance when each year is modeled independently.

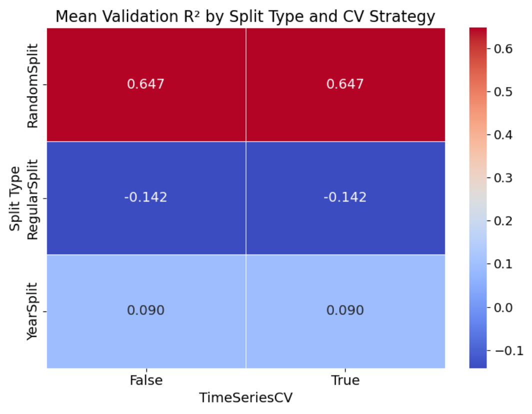
A systematic experiment set was planned to understand the temporal behavior of the data and model. The experiment tested in phases the impact of random vs. time-based train-test split, different splits within cross-validation, the impact of time features, the randomness of data, and the effect of various machine learning models. The experiment details are listed in Table 4.7.

Phase	**Test**	**Data Split**	**Model**	**CV**	**Train-Test Split**	**Time Features**	**Permutate Target**
1	Baseline Tests	All, by year	LGBM	TimeSeries, Regular	Random, Regular, Year split	On	Off
2	Full Days	All, by year	LGBM	Regular	Regular, Day split	On/off	Off
3	Lag Structure	By year	LGBM	Regular	Random, Regular	On/Off	Off
4	Permutation	All	LGBM	Regular	Year split	On	On/Off
6	Feature Importance	All, by year	LGBM	Regular	Random, Regular, Day split	On	On/off
6	Across Models	All	All	Regular	Random, Regular	On	Off

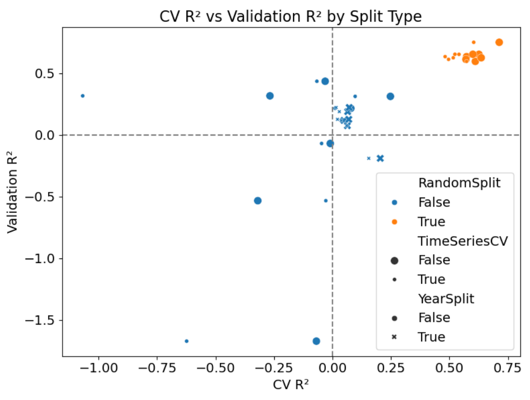
Table 4.7: Overview of testing phases and configurations. Note: In this table, CV refers to how the training set is internally split for cross-validation, such as standard or time series CV. Train-Test Split refers to how the full dataset is split into training and test sets, such as random, year-based, or day-based. For details, refer to Section 3.7.

4.6.1 Baseline tests

The first test was performed for the whole dataset and individual years. The varied parameters were randomized train-test samples on/off, using one year as a test set (for the entire dataset) and using time series CV split. The main results are summarized in Figure 4.11. The first finding was that using time series cross-validation only affects CV R^2, not validation R^2. Both regular train tests were split without randomness and used for one year as a test set, resulting in a similar catastrophic performance collapse.



(a) Impact of random train-test split, year-split, and TimeSeries CV split to validation R^2.



(b) Impact of random train-test split, year-split, and TimeSeries CV split to validation and cross-validation R^2.

Figure 4.11: Baseline tests: impact of random train-test split, year-split and time-series cross-validation split.

When individual years were modeled separately, the model performance was rel-

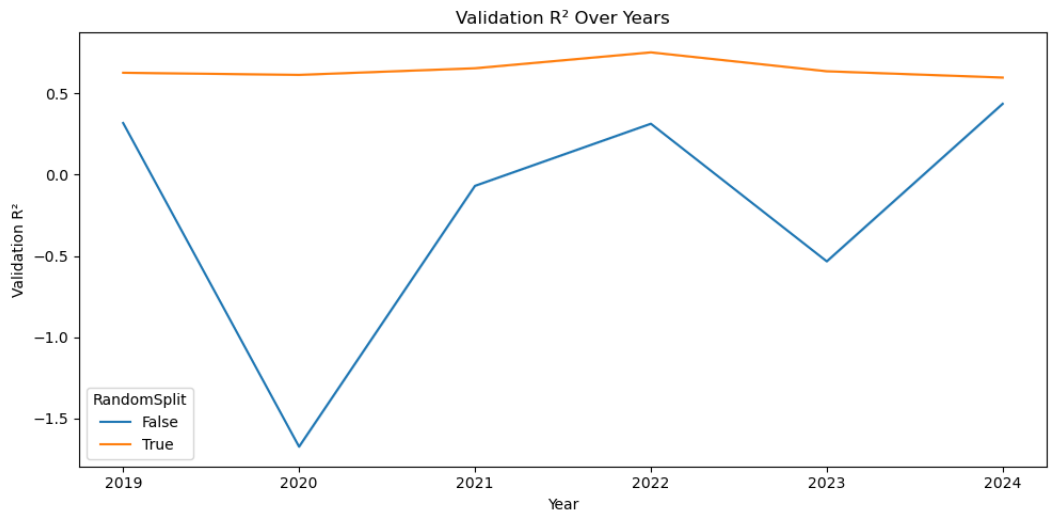


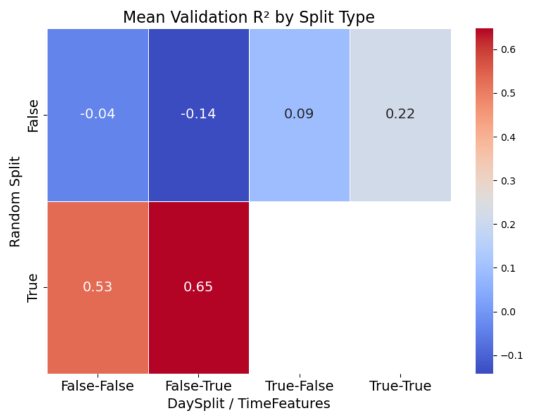
Figure 4.12: Effect of randomized vs regular train-test split when individual years are modeled separately.

atively stable and good as long as a random train-test split was used, as shown in Figure 4.12. Without a random train-test split, the performance reduced significantly. There was considerable variation in model performance between years. Validation R^2 collapsed in all cases. Nevertheless, the performance deteriorated in all cases. It should also be noted that there were only 2200–2500 samples per year, which may have contributed to instability or overfitting, especially with complex models like LightGBM.

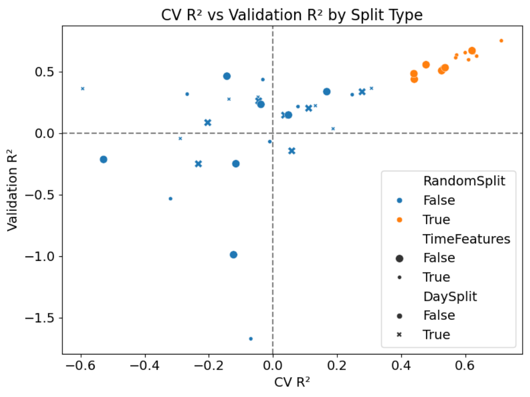
The baseline test indicates that the type of split used in cross-validation is irrelevant to model performance, so the time series CV split was not used in further tests. Still, the train-test split has a huge effect: the model performed well with a randomized train-test split, but without randomization, the performance collapsed for the whole dataset and individual years.

4.6.2 Day-based train-test split

To further test the temporal behavior of the model, a train-test split was done based on full days. Also, the impact of time features was tested in the same runs. Figure 4.13 shows the results. The day-based train-test split showed similar performance collapse as the year-based or regular split. The time features benefited the model performance, but not consistently, and the impact was far less than a collapse when moving out from the randomized train-test split. The test indicated that the model fit the proximity of data points. Adding time features somewhat improved the performance, but not consistently.



(a) Impact of random train-test split, day-split, and time features to validation R2.



(b) Impact of random train-test split, day-split, and time features to CV and validation R2.

Figure 4.13: Impact of random train-test split, day-split, and time features on model performance.

4.6.3 Impact of time lags

The impact of lagged variables was tested by adding a one-time lag step to SO_2 concentration, relative humidity, friction velocity, temperature, and solar radiation. This time lag step corresponded to 30 minutes. Adding time lags did not significantly impact performance; most runs improved slightly, which was natural because the test gave the model 5 additional degrees of freedom. Table 4.8 shows the results. Further time lag points were not tested because no significant improvement was seen. Given the rapid atmospheric processes and sub-daily measurement resolution, longer lag structures were not expected to yield improvement. This test also pointed towards the direction in which the model fitted to proximity and not to causal relationships between the target and features.

RandomSplit	TimeFeatures	TimeLags	CV R^2	Validation R^2
True	True	False	0.60	0.65
True	True	True	0.59	0.63
True	False	True	0.47	0.50
True	False	False	0.44	0.48
False	True	True	0.10	0.24
False	True	False	0.08	0.22
False	False	True	0.09	0.18
False	False	False	0.05	0.15

Table 4.8: Impact of time lags.

4.6.4 Permutation of target variable

A permutation test was performed to validate that the model's predictive power was not simply based on noise, proximity effects, or architectural bias. In this test, the model was trained on the full dataset, excluding one year, and the excluded year was used as the test set. This was repeated for each year from 2019 to 2024. For each test year, three versions of the target variable: without permutation, total random permutation, and day-based permutation

The purpose of this setup was to test whether the model was truly learning meaningful patterns from the environmental predictors or simply exploiting temporal structure in the data. The day-wise permutation, in particular, preserved coarse daily structure while breaking within-day correlations, offering an intermediate case between fully intact and completely randomized targets. The consistent performance ranking across years, even under complete target permutation, likely reflects differences in feature distributions or data quality between years, which affect model behavior regardless of the target.

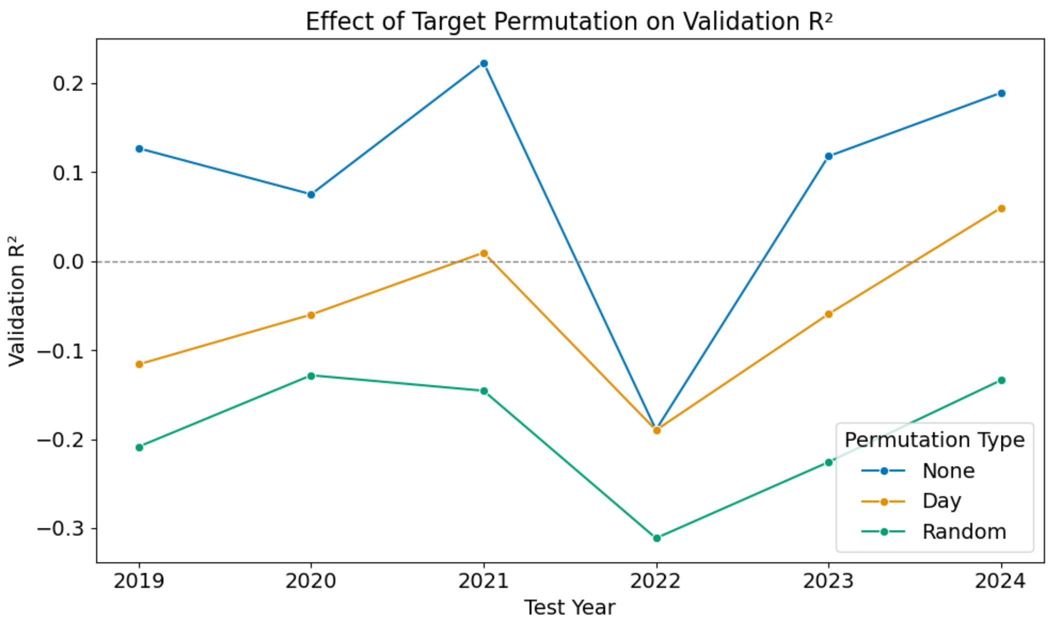
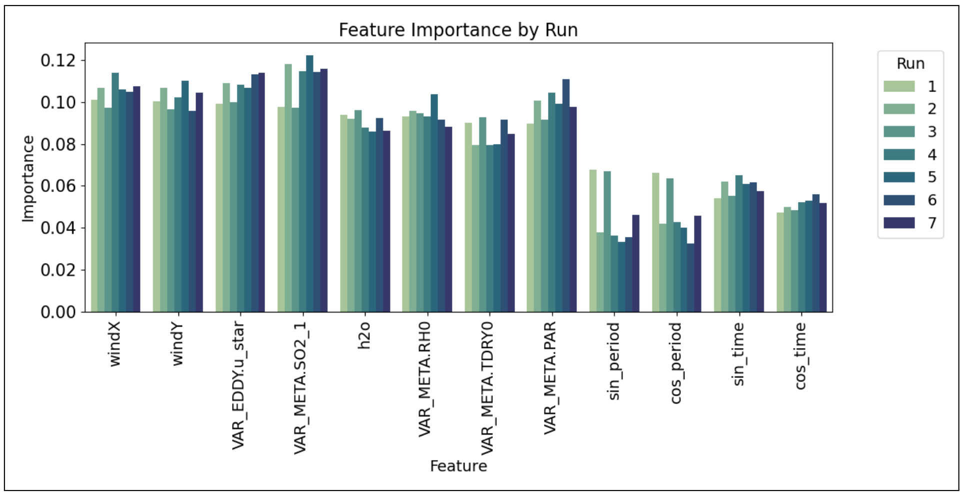


Figure 4.14: Impact of permuting target variable.

The results, summarized in Figure 4.14, show a systematic collapse in model performance when the target variable was permuted, especially under full randomization. This confirms that much of the predictive capacity observed in the LightGBM models across all test years was due to temporal or structural proximity in the data, rather than robust causal relationships with environmental drivers. However, the smaller performance drop under day-wise permutation and the consistent results ranking across years suggest that the model retained some ability to extract partial signals from the predictor variables, even if incomplete. The consistent degradation across all test years still indicates that while the model learned patterns from the training data, those patterns did not generalize meaningfully once the true feature-target relationships were disrupted.

4.6.5 Feature importance analysis

Feature importances were recorded and monitored in all experiments. Notably, the feature importances were similar in all experiments. For presentation clarity, representative examples of failing and non-failing experiments were chosen, including both individual years modeled separately and the whole dataset, with and without a day-split of train-test data, and one experiment with target variable permutation. The run specifications and feature importances are presented in Figure 4.15.



(a) Feature importance accross runs.

Run	Shuffle	DaySplit	Permutation	Data	Validation R2
1	True	False	False	All years	0.65
2	True	False	False	2022	0.75
3	False	False	False	All years	0.22
4	False	False	False	2022	0.31
5	False	True	False	2022	0.36
6	False	True	False	2023	-0.04
7	False	False	True	2022	-0.31

(b) Run configurations for feature importances.

Figure 4.15: Feature importance and corresponding run configurations.

Notably, the feature importances were similar in all cases, regardless of validation

R2. All physical parameters had close to 10 % importance, and time features had approximately 5 % importance each. Hence, features could not be classified as dominant features or weak features. The fact that each feature had similar significance indicated that the model did not fit meaningful causal relationships but rather the proximity of data points.

The slightly elevated importance of SO_2 across runs may partly reflect its broader value range and higher variance than other predictors. In tree-based models, features with greater variability often produce more distinct split points, leading to marginally higher gains in loss reduction. However, the overall feature importances remained within a narrow range (typically 8–12 %), indicating relatively uniform usage across features and no clear dominance in driving predictions.

4.6.6 Impact of model

All previous time-series analysis tests were made using LightGBM. An experiment involving various models was conducted to validate that the observed effects were not attributable to model architecture or incorrect configuration. The configuration was for the whole dataset, with a randomized train-test split on and off as the only variable parameter. Otherwise, the same normalizations and scalings were used, except for the neural network, where standard scaling was used as was customary. Results of testing various models are presented in Figure 4.16.

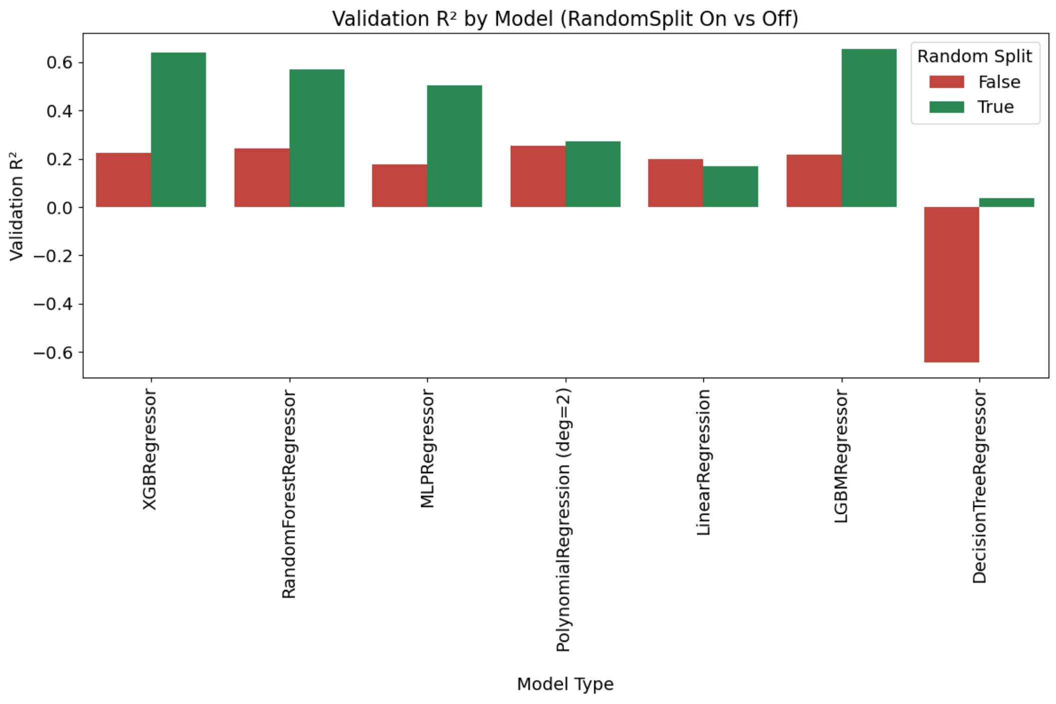


Figure 4.16: Impact of model.

Neural network, random forest, XGBRegressor, and LightGBM showed similar and significant performance decreases. For the decision tree, the starting performance, already with a randomized train-test split, was poor, and removing the randomized train-test split caused a catastrophic performance collapse. Linear regression and 2nd degree polynomial regression had relatively poor performance, but removing the randomized train-test did not decrease the performance. Performance decrease regardless of the model architecture was another strong indication that data did not contain causal relationships: while neural networks and all tree-based methods can exploit the proximity of data points, linear and polynomial regression cannot, as they are limited to only giving coefficients to actual features.

4.6.7 Test prediction

Despite the relatively poor performance of the modelling, a qualitative prediction test was made. In this test, the LightGBM model was used. Each year was modeled separately, with a regular split, meaning that the last 20 % of the period was predicted. As a number of data points, this meant training data between 1746 and 1979 and test data between 437 and 495, depending on missing target variable values and a number of filtered out rain events. The predicted period was thus approximately 10 days, though not continuous, as data points with rain are filtered out. Modelling each year separately was chosen for this test, as at least some years gave reasonable prediction performance, even though for some years the model had high negative validation R^2. The model hyperparameters were re-tuned for this test to account for the smaller dataset. Table 4.9 presents the cross-validation and validation scores.

Year	CV R^2	Validation R^2
2024	0.031	0.418
2023	-0.191	-0.030
2022	0.253	0.375
2021	0.153	0.147
2020	-0.032	-1.435
2019	-0.016	0.331

Table 4.9: Cross-validation and validation performance on separately modeled years.

The results of predictions are presented in Figure 4.17. There were no clear visual hints in the plots explaining why some years performed reasonably well in prediction, and others did not. Target variable peaks with longer duration might have been predicted slightly better than short, sharp peaks. However, this may also be a coincidence, considering the limited dataset.

Various statistical checks were made to understand why certain years modeled reasonably well, but others did not. However, no clear dependence on environmental parameters was found. For example, considering the amount of rain during summer, the number of rain events per year is listed in Table 4.10. But considering that well-performing years were 2019, 2022, and 2024, the number of rain events did not correlate with the model performance; 2019 was the summer with the most rain events, 2024 with the least rain events, and 2022 in the middle of the dataset.

Year	Dapoints	Rain events
2019	2976	534
2020	2976	444
2021	2976	514
2022	2976	505
2023	2976	510
2024	2976	368

Table 4.10: Number of data points classified as rain events vs. total data points per year.

Figure 4.18 presents additional mean comparisons for key variables. Similarly, this did not reveal any simple correlations: well-performing years 2019 and 2024 had the lowest mean negative ion concentration, but on the other hand, well-performing year 2022 had the highest negative ion concentration. Mean SO_2 concentration was highest in 2019, but lowest in 2022, while 2024 was approximately the median of the dataset. 2019 was the coolest summer, 2024 was the warmest summer, while 2022 was between these two. Of course, this was quite a rudimentary analysis, but at least this did not reveal any simple explanatory factors as to why some years modeled reasonably and others did not.

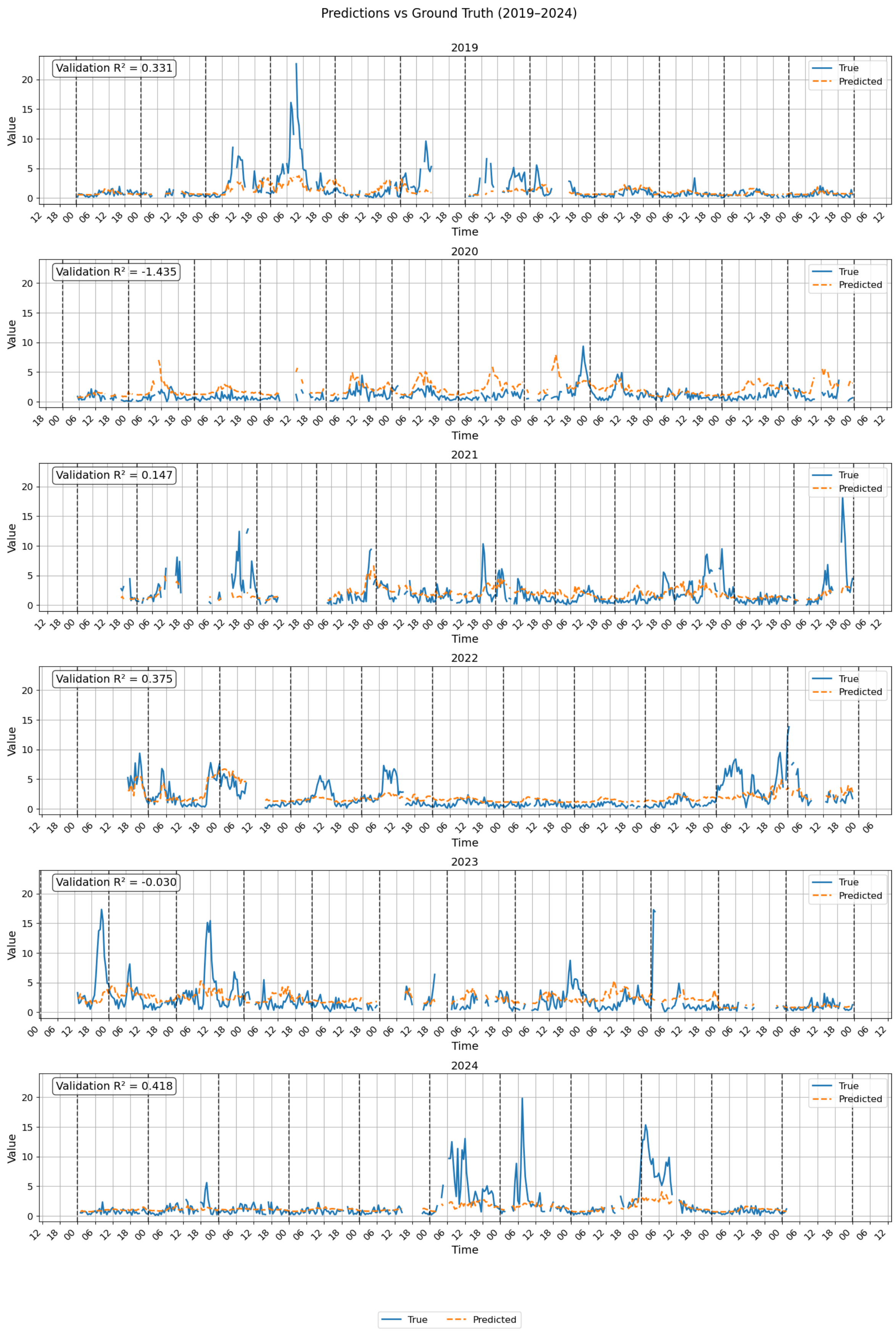


Figure 4.17: Last 20 % of each year predicted when years are modeled separately.

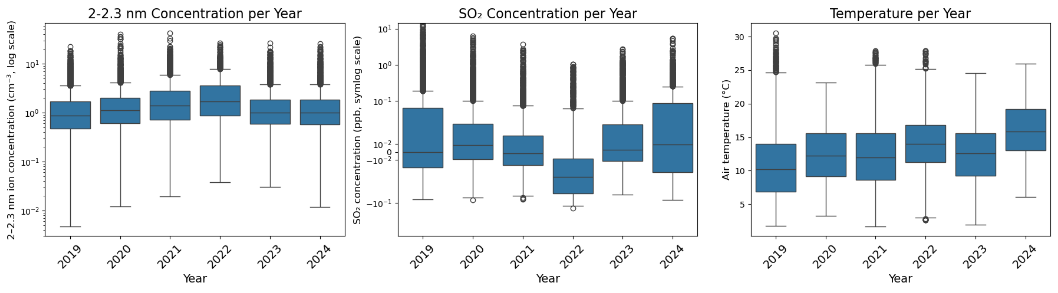


Figure 4.18: Annual boxplots of 2–2.3 nm negative ion concentration, SO_2 concentration, and temperature. The 2–2.3 nm ion concentrations are shown on a logarithmic scale, SO_2 concentrations on a symmetric logarithmic (symlog) scale with a linear threshold of 0.01, and temperature on a linear scale.

4.6.8 Conclusions on time series analysis

The experiment set performed to test the temporal nature of the data strongly indicated that the available features, as currently measured, did not allow the model to generalize in time. The seemingly good performance with random train-test splits suggests that some short-term structure was present, but this likely reflected temporal proximity rather than robust relationships. When time-based splits were used, model performance collapsed, indicating a failure to capture stable predictive patterns. This may be due to missing explanatory variables, noisy measurements, or an inherently weak signal in the available predictors. While some features may hold partial predictive value, the dataset in its current form does not support building a reliable empirical model capable of predicting ion concentrations over extended summer periods.

5. Conclusions

Even though the impact of rain on NAIS measurement is a previously known issue [20, 29, 42], this study further emphasized the effect of even minor rain on 2–2.3 nm negative ion concentration. Most existing research has been more or less NPF event-based, with various manual data processing steps, and not so much focused on continuous time series modeling; hence, the importance of filtering out rain events has not been so strongly emphasized. It was a clear finding that rain causes substantial distortion to 2–2.3 nm negative ion concentration measurement. No meaningful modeling is possible if rain events are included in the dataset.

Various modeling options were tested. It was shown that it is not possible to reliably predict target variable 2–2.3 nm with the features ambient temperature, relative humidity, wind speed and direction, photosynthetically active radiation, friction velocity, and SO_2 concentration. Ensemble models could predict the target when the train-test split was randomized, but the predictive power disappeared with time-aware train-test splits. The predictive power was lost because, with a randomized train-test split, tree-based ensemble models and neural networks, most likely learned patterns based on temporal proximity rather than genuine causal relationships. The testing was repeated with multiple model architectures: linear regression, polynomial regression, decision tree, random forest, XGBoost, LightGBM, and neural network to validate that performance limitations were not due to model architecture or configuration, but rather limitations in the data.

Feature engineering resulted in some model improvements, but the magnitude was small. A wide range of engineered features, transformations, and normalizations was tested systematically, ensuring all combinations were also tested.

The analysis suggests that the features used are insufficient predictors for 2–2.3 nm negative ion concentration. Organic vapors are an essential factor for ion formation [31], and the current dataset did not contain any organics data, which may have affected the prediction performance. Another potentially limiting factor could be the time averaging. The target variable was measured with a 30-minute averaging time. On the other hand, ion formation is known to be time-dependent and happening in bursts [13, 18, 32]. Such a long averaging time may mask temporal behavior and limit model performance. While higher-resolution data introduces more noise, machine learning approaches can be designed to manage this, and access to finer temporal resolution could improve model performance by exposing more underlying dynamics.

A computational setup with Hydra controlling experiment parameters and their combinations, MLflow logging results, parameters, and artifacts, coupled with a fast LightGBM model, proved effective for this type of work, where testing numerous combinations systematically and maintaining reproducibility are essential.

Taking into account the previously mentioned limitations, recommendations for future research for machine learning modeling 2–2.3 nm negative ion concentration are:

- Additional features are most likely needed. Potentially, any VOC data might be beneficial, as well as NO_x and ozone. Component-specific BVOC concentrations could be added in the optimal case, but even total VOC measurement would likely be beneficial. Continuous VOC data could be measured at ppb levels, for example, with A photoionization detector (PID) or flame ionization detector (FID).

- In general, it might be good to include more meteorological parameters, even if their impact is unlikely. Generally, in machine learning, spotting unnecessary features and dropping them is relatively easy. In a situation where the feature set is insufficient to predict the target, there is very little to do in modeling.

- As previously pointed out, the averaging period may be a limiting factor. Therefore, a much shorter averaging period for the target variable measurement would be recommended. The same principle applies to the number of features; if the averaging period is too short and the data is too noisy, the data can always be aggregated during modeling. Also, averaging during modeling would allow techniques such as time-weighted averaging, which could be helpful.

- Also, running the same experimentation on other SMEAR stations could reveal more about the dependence of negative ion concentration on ambient parameters. Considering the poor model performance in this study, it is advisable to apply the above recommendations to improve performance with Värriö data, before extending the analysis to other SMEAR sites.

- Careful filtering of rain events is also critical when modeling ion concentrations.

Bibliography

[1] A. I. Adler and A. Painsky. Feature Importance in Gradient Boosting Trees with Cross-Validation Feature Selection. *Entropy*, 24(5):687, 2022.

[2] J. Almeida, S. Schobesberger, A. Kürten, I. K. Ortega, O. Kupiainen-Määttä, A. P. Praplan, A. Adamov, A. Amorim, F. Bianchi, M. Breitenlechner, A. David, J. Dommen, N. M. Donahue, A. Downard, E. Dunne, J. Duplissy, S. Ehrhart, R. C. Flagan, A. Franchin, R. Guida, J. Hakala, A. Hansel, M. Heinritzi, H. Henschel, T. Jokinen, H. Junninen, M. Kajos, J. Kangasluoma, H. Keskinen, A. Kupc, T. Kurtén, A. N. Kvashin, A. Laaksonen, K. Lehtipalo, M. Leiminger, J. Leppä, V. Loukonen, V. Makhmutov, S. Mathot, M. J. McGrath, T. Nieminen, T. Olenius, A. Onnela, T. Petäjä, F. Riccobono, I. Riipinen, M. Rissanen, L. Rondo, T. Ruuskanen, F. D. Santos, N. Sarnela, S. Schallhart, R. Schnitzhofer, J. H. Seinfeld, M. Simon, M. Sipilä, Y. Stozhkov, F. Stratmann, A. Tomé, J. Tröstl, G. Tsagkogeorgas, P. Vaattovaara, Y. Viisanen, A. Virtanen, A. Vrtala, P. E. Wagner, E. Weingartner, H. Wex, C. Williamson, D. Wimmer, P. Ye, T. Yli-Juuti, K. S. Carslaw, M. Kulmala, J. Curtius, U. Baltensperger, D. R. Worsnop, H. Vehkamäki, and J. Kirkby. Molecular understanding of sulphuric acid-amine particle nucleation in the atmosphere. *Nature*, 502(7471):359–363, 2013.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[5] K. S. Carslaw, O. Boucher, D. V. Spracklen, G. W. Mann, J. G. L. Rae, S. Woodward, and M. Kulmala. A review of natural aerosol interactions and feedbacks within the Earth system. *Atmospheric Chemistry and Physics*, 10(4):1701–1737, 2010.

[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794. Association for Computing Machinery, Aug. 2016.

[7] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell. Introduction to Machine Learning, Neural Networks, and Deep Learning. *Translational Vision Science & Technology*, 9(2):14, 2020.

[8] M. Dal Maso, M. Kulmala, I. Riipinen, R. Wagner, T. Hussein, P. P. Aalto, and K. E. J. Lehtinen. Formation and growth of fresh atmospheric aerosols: eight years of aerosol size distribution data from SMEAR II, Hyytiälä, Finland. *Boreal Environment Research*, 10(5):323–336, 2005.

[9] W. Ertel. *Introduction to Artificial Intelligence*. Springer Nature, 2024.

[10] E. Ezhova, I. Ylivinkka, J. Kuusk, K. Komsaare, M. Vana, A. Krasnova, S. Noe, M. Arshinov, B. Belan, S.-B. Park, J. V. Lavrič, M. Heimann, T. Petäjä, T. Vesala, I. Mammarella, P. Kolari, J. Bäck, U. Rannik, V.-M. Kerminen, and M. Kulmala. Direct effect of aerosols on solar radiation and gross primary production in boreal and hemiboreal forests. *Atmospheric Chemistry and Physics*, 18(24):17863–17881, 2018.

[11] J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[12] A. B. Guenther, P. R. Zimmerman, P. C. Harley, R. K. Monson, and R. Fall. Isoprene and monoterpene emission rate variability: Model evaluations and sensitivity analyses. *Journal of Geophysical Research: Atmospheres*, 98(D7):12609–12617, 1993.

[13] U. Hõrrak, J. Salm, and H. Tammet. Bursts of intermediate ions in atmospheric air. *Journal of Geophysical Research: Atmospheres*, 103(D12):13909–13915, 1998.

[14] A. Hamed, H. Korhonen, S.-L. Sihto, J. Joutsensaari, H. Järvinen, T. Petäjä, F. Arnold, T. Nieminen, M. Kulmala, J. N. Smith, K. E. J. Lehtinen, and A. Laaksonen. The role of relative humidity in continental new particle formation. *Journal of Geophysical Research: Atmospheres*, 116(D3), 2011.

[15] J. Han, J. Pei, and H. Tong. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2022.

[16] Hari, P., Kulmala, M., Pohja, T., Lahti, T., Siivola, E., Palva, L., Aalto, P., Hämeri, K., Vesala, T., Luoma, S., & Pulliainen, E. Air pollution in eastern Lapland : challenge for an environmental measurement station. *Silva Fennica*, 28(1):29–39, 1994.

[17] T. Hastie, J. Friedman, and R. Tibshirani. *The Elements of Statistical Learning*. Springer, 2001.

[18] A. Hirsikko, T. Bergman, L. Laakso, M. Dal Maso, I. Riipinen, U. Hõrrak, and M. Kulmala. Identification and classification of the formation of intermediate ions measured in boreal forest. *Atmospheric Chemistry and Physics*, 7(1):201–210, 2007.

[19] A. Hirsikko, T. Nieminen, S. Gagné, K. Lehtipalo, H. E. Manninen, M. Ehn, U. Hõrrak, V.-M. Kerminen, L. Laakso, P. H. McMurry, A. Mirme, S. Mirme, T. Petäjä, H. Tammet, V. Vakkari, M. Vana, and M. Kulmala. Atmospheric ions and nucleation: a review of observations. *Atmospheric Chemistry and Physics*, 11(2):767–798, 2011.

[20] U. Hõrrak, H. Tammet, P. Aalto, M. Vana, A. Hirsikko, L. Laakso, and M. Kulmala. Formation of charged nanometer aerosol particles associated with rainfall: Atmospheric measurements and lab experiment. *Rep. Ser. Aerosol Sci.*, 80:180–185, 2006.

[21] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer Nature, 2023.

[22] H. Junninen, A. Lauri, P. I. R. Keronen, P. Aalto, V. Hiltunen, P. Hari, and M. Kulmala. Smart-SMEAR: on-line data exploration and visualization tool for SMEAR stations. *Boreal Environment Research*, 14:447–457, 2009.

[23] X. Junzeng, W. Qi, P. Shizhang, and Y. Yanmei. Error of Saturation Vapor Pressure Calculated by Different Formulas and Its Effect on Calculation of Reference Evapotranspiration in High Latitude Cold Region. *Procedia Engineering*, 28:43–48, 2012.

[24] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[25] V.-M. Kerminen, X. Chen, V. Vakkari, T. Petäjä, M. Kulmala, and F. Bianchi. Atmospheric new particle formation and growth: review of field observations. *Environmental Research Letters*, 13(10):103003, 2018.

[26] P. Kolari. Using SMEAR api. `https://wiki.helsinki.fi/xwiki/bin/view/SMEAR/The%20SMEAR%20Wikispace/SMEAR%20database/Using%20SMEAR%20API/`, 2024. Accessed February 11, 2025. University of Helsinki.

[27] M. Kulmala, E. Ezhova, T. Kalliokoski, S. Noe, T. Vesala, A. Lohila, J. Liski, R. Makkonen, J. Bäck, T. Petäjä, V.-M. Kerminen, and P. Kerminen. Carbon-Sink+ -Accounting for multiple climate feedbacks from forests. *Boreal Environment Research*, 25:145–159, 2020.

[28] M. Kulmala, J. Kontkanen, H. Junninen, K. Lehtipalo, H. E. Manninen, T. Nieminen, T. Petäjä, M. Sipilä, S. Schobesberger, P. Rantala, A. Franchin, T. Jokinen, E. Järvinen, M. Äijälä, J. Kangasluoma, J. Hakala, P. P. Aalto, P. Paasonen, J. Mikkilä, J. Vanhanen, J. Aalto, H. Hakola, U. Makkonen, T. Ruuskanen, R. L. Mauldin, J. Duplissy, H. Vehkamäki, J. Bäck, A. Kortelainen, I. Riipinen, T. Kurtén, M. V. Johnston, J. N. Smith, M. Ehn, T. F. Mentel, K. E. J. Lehtinen, A. Laaksonen, V.-M. Kerminen, and D. R. Worsnop. Direct Observations of Atmospheric Aerosol Nucleation. *Science*, 339(6122):943–946, 2013.

[29] L. Laakso, A. Hirsikko, T. Grönholm, M. Kulmala, A. Luts, and T.-E. Parts. Waterfalls as sources of small charged aerosol particles. *Atmospheric Chemistry and Physics Discussions*, 6(5):9297–9314, 2006.

[30] S.-H. Lee, H. Gordon, H. Yu, K. Lehtipalo, R. Haley, Y. Li, and R. Zhang. New Particle Formation in the Atmosphere: From Molecular Clusters to Global Climate. *Journal of Geophysical Research: Atmospheres*, 124(13):7098–7146, 2019.

[31] K. Lehtipalo, C. Yan, L. Dada, F. Bianchi, M. Xiao, R. Wagner, D. Stolzenburg, L. R. Ahonen, A. Amorim, A. Baccarini, P. S. Bauer, B. Baumgartner, A. Bergen, A.-K. Bernhammer, M. Breitenlechner, S. Brilke, A. Buchholz, S. B. Mazon, D. Chen, X. Chen, A. Dias, J. Dommen, D. C. Draper, J. Duplissy, M. Ehn, H. Finkenzeller, L. Fischer, C. Frege, C. Fuchs, O. Garmash, H. Gordon, J. Hakala, X. He, L. Heikkinen, M. Heinritzi, J. C. Helm, V. Hofbauer, C. R. Hoyle, T. Jokinen, J. Kangasluoma, V.-M. Kerminen, C. Kim, J. Kirkby, J. Kontkanen, A. Kürten, M. J. Lawler, H. Mai, S. Mathot, R. L. Mauldin, U. Molteni, L. Nichman, W. Nie, T. Nieminen, A. Ojdanic, A. Onnela, M. Passananti, T. Petäjä, F. Piel, V. Pospisilova, L. L. J. Quéléver, M. P. Rissanen, C. Rose, N. Sarnela, S. Schallhart, S. Schuchmann, K. Sengupta, M. Simon, M. Sipilä, C. Tauber, A. Tomé, J. Tröstl, O. Väisänen, A. L. Vogel, R. Volkamer, A. C. Wagner, M. Wang, L. Weitz, D. Wimmer, P. Ye, A. Ylisirniö, Q. Zha, K. S. Carslaw, J. Curtius, N. M. Donahue, R. C. Flagan, A. Hansel, I. Riipinen, A. Virtanen, P. M. Winkler, U. Baltensperger, M. Kulmala, and D. R. Worsnop. Multicomponent new particle formation from sulfuric acid, ammonia, and biogenic vapors. *Science Advances*, 4(12):eaau5363, 2018.

[32] K. Leino, T. Nieminen, H. E. Manninen, T. Petäjä, V.-M. Kerminen, and M. Kulmala. Intermediate ions as a strong indicator for new particle formation bursts in a boreal forest. *Boreal Environment Research*, 21(3-4):274–286, 2016.

[33] A. Mirme, E. Tamm, G. Mordas, M. Vana, J. Uin, S. Mirme, T. Bernotas, L. Laakso, A. Hirsikko, and M. Kulmala. A wide-range multi-channel Air Ion Spectrometer. *Boreal Environment Research*, 12(3):247–264, 2007.

[34] S. Mirme and A. Mirme. The mathematical principles and design of the NAIS - a spectrometer for the measurement of cluster ion and nanometer aerosol size distributions. *Atmospheric Measurement Techniques*, 6(4):1061–1071, 2013.

[35] R. Mitchell and E. Frank. Accelerating the XGBoost algorithm using GPU computing. *PeerJ Computer Science*, 3:e127, 2017.

[36] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18(6):275–285, 2004.

[37] A. Natekin and A. Knoll. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7, 2013.

[38] E. D. Nilsson, Ü. Rannik, M. Kulmala, G. Buzorius, and C. D. O'Dowd. Effects of continental boundary layer evolution, convection, turbulence and entrainment, on aerosol formation. *Tellus B: Chemical and Physical Meteorology*, 53(4):441–461, 2001.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] T. M. Ruuskanen, M. Kaasik, P. P. Aalto, U. Hõrrak, M. Vana, M. Mårtensson, Y. J. Yoon, P. Keronen, G. Mordas, D. Ceburnis, E. D. Nilsson, C. O'Dowd, M. Noppel, T. Alliksaar, J. Ivask, M. Sofiev, M. Prank, and M. Kulmala. Concentrations and fluxes of aerosol particles during the LAPBIAT measurement campaign at Värriö field station. *Atmospheric Chemistry and Physics*, 7(14):3683–3700, 2007.

[41] Y.-y. Song and Y. Lu. Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2):130–135, 2015.

[42] H. Tammet, U. Hõrrak, and M. Kulmala. Negatively charged nanoparticles produced by splashing of water. *Atmospheric Chemistry and Physics*, 9(2):357–367, 2009.

[43] H. Tammet, K. Komsaare, and U. Hõrrak. Intermediate ions in the atmosphere. *Atmospheric Research*, 135-136:263–273, 2014.

[44] S. Tuovinen, J. Lampilahti, V.-M. Kerminen, and M. Kulmala. Measurement report: Ion clusters as indicator for localnew particle formation. *EGUsphere*, pages 1–18, 2023. Publisher: Copernicus GmbH.

[45] S. Tuovinen, J. Lampilahti, V.-M. Kerminen, and M. Kulmala. Intermediate ions as indicator for local new particle formation. *Aerosol Research*, 2(1):93–105, 2024.

[46] L. Wasserman. *All of statistics: a concise course in statistical inference*. Springer, 2004.

[47] B. Wehner, H. Siebert, A. Ansmann, F. Ditas, P. Seifert, F. Stratmann, A. Wiedensohler, A. Apituley, R. A. Shaw, H. E. Manninen, and M. Kulmala. Observations of turbulence-induced new particle formation in the residual layer. *Atmospheric Chemistry and Physics*, 10(9):4319–4330, 2010.

[48] World Meteorological Organization. *Guide to Instruments and Methods of Observation: Volume I - Measurement of Meteorological Variables*. WMO, 2023.

[49] XGBoost Developers. XGBoost Parameters - xgboost 2.1.3 documentation. `https://xgboost.readthedocs.io/en/stable/parameter.html`. Accessed March 7, 2025.

[50] O. Yadan. Hydra - a framework for elegantly configuring complex applications. `https://github.com/facebookresearch/hydra`, 2019. Accessed February 16, 2025.

Appendix A. Use of artificial intelligence

The following artificial intelligence tools have been used for this thesis:

- Grammarly (app.grammarly.com), an AI-based writing assistance tool, has been used to check and correct grammatical issues in the text.
- ChatGPT (chatgpt.com) has been used for generating starting `tikzpicture` code for figures, which then has been edited.