MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Short-Range Wind Speed Prediction using a Kernel-based Support Vector Machine in Austria

verfasst von / submitted by

Sahebeh Dadboud

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2020 / Vienna, 2020

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von / Supervisor:	Univ. Prof. Dipl.-Inform. Univ. Dr. Claudia Plant

Abstract

Predicting wind speed is challenging due to its dependency on the topography, prevailing weather conditions, and atmospheric dynamics and its rapid changes. Recent studies have shown that support vector regression (SVR) has potentials in wind speed prediction. This thesis's overall objective is to provide reliable point predictions (i.e., tailored forecasts for a particular location) of the wind speed up to two days ahead in hourly resolution. For each location, observation data with 10 minutes update frequency and hourly numerical weather prediction (NWP) model output, such as the high-resolution NWP model for the alpine region AROME, are available. However, due to the high computational complexity, the forecasts of NWP models are delayed by several hours.

To obtain more accurate forecasts in the short-range of up to 48 hours ahead, we combine the observation with the lagged AROME forecasts with a support vector machine (SVM) based algorithm. In particular, we proceed in two steps: (I) we design a kernel-based support vector regression (SVR) identifying general relations in the selected time series data; (II) we validate its performance with the Persistence and the lagged AROME model for a case-study of 28 observation sites located in different Austrian regions with varying topography and climatology for the testing episode of one month, January 2017. Applying SVR for wind speed prediction requires optimizing a set of parameters. For the SVR model, different kernels such as linear, sigmoid, and RBF were used and compared. To optimize results, various *feature selection* methods for dimensionality reduction were applied, such as principal component analysis (PCA), mutual information (MI), Pearson correlation, and SHAP (shapley additive explanations). Due to high computational time in training SVR, we decided to select the 20 most essential features in every individual forecast hour for each station. We carried out parameter optimization via *Grid Search* for one representative station, and applied the result to all other 27 target stations. The SVR model outperforms other models in predicting wind speed. For nowcasting range (up to about 6 hours), basic models like the Persistence model, in some cases, outperform complex models like SVR. In the Persistence model, the most current wind speed measurement of the location is included and, it provides a computationally efficient forecast for all forecast hours. However, experimental results indicate that SVR is a suitable technique for performing short-range wind speed predictions.

Zusammenfassung

Aufgrund der Abhängigkeiten der Windgeschwindigkeit von der Topographie, den vorherrschenden Wetterbedingungen, der atmosphärischen Dynamik und schnellen Veränderungen des atmosphärischen Zustands ist die Vorhersage der Windgeschwindigkeit schwierig. Verwandte Arbeiten demonstrieren, dass Support Vector Regression (SVR) die Kurzfristvorhersage der Windgeschwindigkeit verbessern können. Das übergeordnete Ziel dieser Arbeit besteht darin, zuverlässige Punktvorhersagen (d.h. Vorhersagen für einen bestimmten Ort) der Windgeschwindigkeit von bis zu zwei Tagen in stündlicher Auflösung bereitzustellen. Für jeden Standort stehen Beobachtungsdaten, aktualisiert in 10-Minuten-Takt, sowie stündliche Vorhersagen von numerischen Wettervorhersagemodellen (NWP, z.B.: AROME, ein hochauflösende NWP-Modell für die Alpenregion) zur Verfügung. Aufgrund des hohen Rechenaufwands sind die Vorhersagen von NWP-Modellen jedoch erst mehrere Stunden nach ihrer Initialisierung abrufbar.

Um genauere Kurzfristvorhersagen von den nächsten 48 Stunden zu erhalten, kombinieren wir die Beobachtung mit den aktuellsten AROME-Vorhersagen mit einem SVM-basierten Algorithmus (Support Vector Machine). Dazu gehen wir in zwei Schritten vor: (I) Wir entwerfen eine kernelbasierte Support Vector Regression (SVR), die allgemeine Beziehungen in den ausgewählten Zeitreihendaten identifiziert. (II) Wir validieren die Vorhersagequalität mithilfe des Persistenz- und dem aktuell verfügbaren AROME-Modell für eine Fallstudie von 28 Beobachtungsstellen in verschiedenen österreichischen Regionen mit unterschiedlicher Topographie und Klimaregion für die Testepisode von einem Monat (Januar 2017). Die Anwendung von SVR auf die Windgeschwindigkeitsvorhersage erfordert die Optimierung einiger Parametern. Für das SVR-Modell wurden verschiedene Kernel wie Linear, Sigmoid und RBF verwendet und verglichen. Um die Ergebnisse zu optimieren, wurden verschiedene *feature selection* Methoden zur Dimensionsreduktion angewendet, wie zum Beispiel Hauptkomponentenanalyse (PCA), mutual information (MI), Pearson-Korrelation und SHAP (shapley additive explanations). Aufgrund der hohen Rechenzeit beim Training des SVR werden die 20 wichtigsten Merkmale für jede Prognosestunde und Station ausgewählt. Die Optimierung der Parameter mithilfe von *Grid Search* wurde für eine repräsentative Station durchgeführt und auf alle übrigen 27 Zielstationen angewendet. Das SVR-Modell liefert Verbesserungen im Vergleich zu den anderen Modellen. Für den Nowcasting-Bereich (bis zu ca. +6

Stunden) können Basismodelle, wie das Persistenzmodell, in einigen Fällen bessere Ergebnisse als komplexe Modelle, wie SVR, erzielen. Im Persistenzmodell wird die aktuellste Windgeschwindigkeitsmessung des Standorts einbezogen und liefert eine rechnerisch effiziente Prognose für alle Vorhersagestunden. Experimentelle Ergebnisse zeigen jedoch, dass SVR eine geeignete Technik für die Kurzfristvorhersage der Windgeschwindigkeit darstellt.

Contents

List of Figures

List of Tables

Acknowledgement

I want to express my most profound appreciation to my supervisor Prof. Dr. Claudia Plant, who extended a significant amount of assistance and patience and always provided me with encouragement throughout my project.

I want to extend my sincere thanks to Mag. Dr. Irene Schicker and Dipl.-Ing. Petrina Papazek at ZAMG (Zentralanstalt für Meteorologie und Geodynamik) for their practical suggestions, and their forbearance that cannot be underestimated.

Chapter 1

Introduction

With changing environmental conditions due to anthropogenic global warming and environmental issues related to energy sources such as coal, nuclear power, and others, wind power is a green, and inexhaustible energy source contributing to sustainable energy systems. This source of power plays an essential role in energy supply in many countries. Wind observations or measurements are required for weather monitoring and forecasting, for wind-load climatology, for the probability of wind damage and estimation of wind energy, and as part of the estimation of surface fluxes, for example, evaporation for air pollution dispersion and agricultural applications [(WMO) Updated 2010(a)]. However, the wind power industries face many technical problems, including the accurate and fast prediction of wind speed in the short-range (i.e., one to two days ahead).

There are different models for wind speed prediction, primarily physical numerical weather prediction models (NWP), statistical models, and machine learning models. Machine learning algorithms are generic models useful for decision support. In recent years, machine learning models, such as artificial neural networks (ANN) and support vector machine (SVM), were able to outperform traditional methods. [Lei et al. 2009b].

In this thesis, we implement a new forecasting method based on a machine learning model to improve the prediction of wind speed in the short-range, which is the time frame of up to 48 hours. This research is conducted in cooperation with ZAMG (Zentralanstalt für Meteorologie und Geodynamik), the national weather service of Austria.

1.1 Meteorological Background

Meteorological measurements, i.e., weather observations, have been carried out regularly for centuries. Around 350 BC, Aristotle, a Greek philosopher, wrote the first weather book, *Meteorologica*, which was describing weather terms like clouds, rain, snow, and wind. Traditional weather forecasting methods usually relied on observations. However, the modern age of weather forecasting began in 1835 when the invention of the electric telegraph made an exchange of weather data from various stations. It was in the 20th century that the advancement of atmospheric physics led to the foundation of numerical weather prediction (NWP).

Some of the main meteorological parameters used in weather predictions are wind speed, wind direction, temperature, pressure, humidity, and precipitation at the Earth's surface [Rösemann 2011]. Wind systems are commonly categorized by their speed, the types of forces that cause them, the regions where they occur, topographic impacts, and their effects on climate.

Wind velocity is a three-dimensional vector (two horizontal and one vertical) quantity with small-scale random fluctuations in space and time superimposed upon a larger-scale organized flow. Three-dimensional vector wind is considered, for example, for airborne pollution and the landing of aircraft. However, surface wind is mainly considered as a two-dimensional vector. We neglect the vertical component and only consider the horizontal ones, specified by two numbers representing direction and speed [(WMO) Updated 2010(a)].

Wind is driven by forces such as gravitation, the gradient of the pressure, friction, and the Coriolis effect, which can be described formally by motion equations. Wind consists of bulk flow of air and is generated by differences in temperature within the atmosphere due to differential solar heating [Manwell, McGowan and Rogers 2009]. Wind speed should be reported to a resolution of 0.5 ms^{-1} or in knots (0.515 ms^{-1}) to the nearest unit and should represent, for synoptic reports, an average over 10 minutes. Averages over a shorter period are necessary for specific aeronautical purposes [(WMO) Updated 2010(b)]. Additionally, wind speed has different behaviors in the atmospheric boundary layer (ABL), also known as the planetary boundary layer (PBL). ABL is the lowest part of the atmosphere that is directly influenced by the Earth's surface and has a very complex behavior in the Alpine area, due to its high spatial and temporal variability [Ketterer et al. 2014]. Fast changing local wind systems in ABL associated with mountainous regions, make wind speed predictions much harder.

Wind speed predictions can be divided into four scales: very short-range or nowcasting (up to

6 hours), short-range (beyond 12 hours and up to 72 hours), mid-range (beyond 72 hours and up to 240 hours), and long-range (from 30 days up to two years) [(WMO) n.d.].

Long-range wind speed prediction is vital for the location and dimension decisions of wind power applications [G, C and M 2009; A and F 2009]. In contrast, short-range forecasting of wind speed is essential for improving wind power generation systems [Monfared, Rastegar and Kojabadi 2009; P.Pinson et al. 2009] and are usually of great interest for system operators, electricity companies, and wind farm promoters. Commonly, the temporal resolution can vary between 5 minutes and hours for short-range wind speed forecasts. Short-range forecastings are employed in the daily and intraday spot market, system management, and maintenance schedules.

1.2 Numerical Weather Prediction Models

Numerical weather prediction models (NWP) are physical-dynamical models of the atmosphere to forecast the weather based on current weather conditions. The first attempt was made in 1920, but by 1950, NWP started producing realistic results using computer simulation [Coiffier 2011].

An NWP model is a computer program that produces meteorological information for future times at given locations and altitudes. By using a set of equations, this program computes, among others, the density, the pressure, and the wind vector in different vertical layers, from oceans to the top of the atmosphere. Due to their non-linearity, analytical solutions are computationally expensive to find, and therefore, numerical approximations are applied.

NWP models divide the atmosphere into 3D cubes, and grid points are put in the middle of the cube. NWP calculates different weather parameter equations for each atmospheric variable at each grid point. The result of this calculation represents the grid box area average. The least distance between the adjacent grid points represents the horizontal model resolution.

Higher-resolution models are more accurate than lower-resolution ones and have more computational needs, which often run on supercomputers in a high-performance computing (HPC) environment. NWP models divide the atmosphere also vertically into layers to illustrate the weather phenomena. The more the number of vertical layers, the higher the chance to demonstrate the weather phenomena, and the more computational power is needed. Besides, higher-resolution models require shorter time intervals and hence, more computational power [Al-Yahyai, Charabi and Gastli 2010].

1.3 Related Work in Wind Speed Forecasting

The stochastic characteristic of wind speed makes the wind speed prediction challenging. Wind speed forecast results can be distinguished by time horizons, which include short-range predictions (timescales of minutes, hours, or days) and long-range predictions (timescales of months or years). Many methods have been developed in the field of wind speed prediction. The state-of-the-art methods can mostly be classified into physical, statistical, and artificial intelligence methods [Lei et al. 2009a; Alexandre Costaa and et al. 2007; Carro-Calvoa et al. 2011; D.A.Fadare 2010].

Physical methods use numeric, parameterization, observation, and physical data such as temperature, pressure, and topography to predict wind speed. Physical methods use the advantages of physical algorithms to integrate data observed by wind turbines into numerical weather forecast (NWF) or mesoscale models [Watson, Landberg and Halliday 1994] to improve the prediction accuracy of wind speed. Statistical methods based on statistical analysis are the main methods associated with wind energy production forecasting, such as wind speed and temperature. These methods have a lower horizon validation between 2 and 6 hours. In contrast, conventional statistical methods use a mathematical model of the problem. Some of these models are auto-regressive (AR) models [Poggi et al. 2003], auto-regressive moving average (ARMA) models [Erdem and Shi 2011], auto-regressive integrated moving average (ARIMA) models [J.A. 2011; A.Sfetsos 2000], Markov chain model [A.Shamshad et al. 2005] and Kalman filtering [Bossanyi 1985; O.Carranzaa et al. 2011].

Artificial intelligence (AI) methods have been the most widely adopted methods for wind speed prediction. AI methods typically attempt to minimize training errors through the use of historical wind speed data. Due to their excellent nonlinear mapping, generalization, and self-learning, AI-based methods prove to be of widespread utility in the wind speed forecasting field. Artificial neural networks (ANNs) [G. Li and Shi 2010; A.Kalogirou 2001] including (radial basis functions (RBFs) [Silva, Fonte and Quadrado 2006], recurrent neural networks (RNNs) [Kariniotakis, Stavrakakis and Nogaret 1996; More and M.C.Deo 2003], Bayesian learning model [G. Li, Shi and Zhou 2011], fuzzy logic model [K. G., E. and S. G. 1996; Wang et al. 2004], adaptive neuro-Fuzzy inference model [S.Rehman and S.M.Rahmand 2011; R.Ataa and Y.Kocyigit 2010], K-means clustering model [Kusiak and W. Li 2010] and KNN (K nearest neighbor) [Yesilbudak, Sagiroglu and Colak 2013] were implemented in the literature.

In recent years, support vector machines (SVMs) has been successfully used for time-series prediction with satisfactory prediction results in various fields [T.B., I. and M.B. 2007; Hong 2008; Osowski and Garanty 2007; Zhou and Shi 2009; Widodo et al. 2009; G.-F. Lin et al. 2009]. SVM has been used in different applications as well. M.A.Mohandes [M.A.Mohandes et al. 2004] applied SVM for wind speed prediction and compared its performance with the multi-layer perceptron (MLP) of neural networks. The results indicate that the SVM method outperforms the MLP model in terms of root mean squared error (RMSE). [Salcedo-Sanz et al. 2011] studied wind speed predictions by SVM in a Spanish wind farm. Two new hybrid methods (EP-SVM and PSO-SVM) were proposed to improve the performance of the classical SVM. Their results showed that both hybrid models had satisfactory results. Jing Shi, Jinmei Guo and Songtao Zheng compared the performance of the ARIMA, ANN, or SVM models in wind speed short-range predictions [Shi, Guo and S. Zheng 2012]. Their results showed that both performances of the hybrid ARIMA-ANN and the ARIMA-SVM were better than those of the single ARIMA, ANN, and SVM models. Furthermore, Sreelakshmi and Kumar [K and PR 2008] indicated that SVM models could give better accuracy than the neural network models in short-range wind forecasts. Kusiak [Kusiak, H. Zheng and Song 2009] studied the prediction of both wind speed and wind power at forecast intervals of 10-minutes up to 1-hour. They observed that the SVM method is a better choice compared to the MLP algorithm.

This thesis aims to present a machine learning methodology based on short-range wind speed prediction to predict 10-minute frequencies of wind speed by using both Observation and NWP data as input for 28 observation sites in Austria in different complex/semi-complex terrains. The aim is to discuss the performance of three types of kernel functions of the SVR regression method namely radial, linear and sigmoid, and select the best kernel for comparing the results before and after applying different feature selection methods such as principal component analysis (PCA), Pearson correlation, mutual information (MI), and SHAP method. We intend to optimize the well-performing kernel by tuning its parameters to investigate their performance on the prediction.

1.4 Overview

The remainder of this thesis is organized as follows: The machine learning principals for SVM is introduced in Chapter 2. Chapter 3 describes the data-sets used for this study and explains the data cleaning and data engineering process. The modeling workflow and optimization of the model are discussed in Chapter 4. The wind speed prediction results are presented and discussed in Chapter 5. In Chapter 6, we draw the conclusions.

Chapter 2

Machine Learning Principals

Data mining is the process of discovering potentially useful information from large amounts of data [Rushing et al. 2005]. Data mining techniques search for new information without demanding any a priori hypotheses, and they are added to discover pattern-specific data mining tasks. There are two types of data mining tasks: a descriptive data mining task describes the general properties of the existing data, and a predictive data mining task attempts to make predictions based on inference on available data.

Methods and algorithms used for wind speed prediction in the field of data mining are supervised and unsupervised machine learning algorithms. The term machine learning is the system of automatic data processing and decision-making algorithms to improve their operation according to their work results.

The goal of this thesis is to design an SVM based algorithm as a prediction method. This section provides an introduction to the SVM and describes the operation of the proposed model.

2.1 Support Vector Machine

SVM is based on statistical learning theory and was proposed by Vapnik in 1995 [Cortes and V. Vapnik 1995]. SVM is a supervised learning model that analyzes data for classification, regression analysis, and outlier detection. Supervised learning algorithms are learning algorithms that learn to associate some input x with some output y. A supervised learning algorithm can infer a function from labeled training data consisting of a set of training examples [Russell and Norvig 1995].

SVMs can build a hyper-plane to classify the classes. The best hyper-plane is the one that provides an optimal separation, i.e., maximizes the distance to the nearest data points of any class. These data points which influence our hyper-plane are known as support vectors.

SVM has two different classifiers: linear and non-linear. The main task of a classifier is to find patterns and study the type of associations for different types of data.

SVM Linear Classifier:

The original maximum-margin hyper-plane algorithm proposed by Vapnik in 1963 constructed a linear classifier.

$$w^T x + b = b + \sum_{i=1}^{m} \alpha_i x^T x^i \tag{2.1}$$

In equation 2.1, x^i is a training sample, and α is a vector of coeffcients. Linear classifier finds linear boundaries in the input feature space. If a data point considers a p-dimensional vector, then the linear classifier separates the points using a (p-1) dimensional hyper-plane. The distance between the points and the dividing plain is known as margin. The larger the margin, the less misclassification [Hastie, Tibshirani and Friedman 2008]. Figure 2.1 shows the linear classifier with the optimal margin.

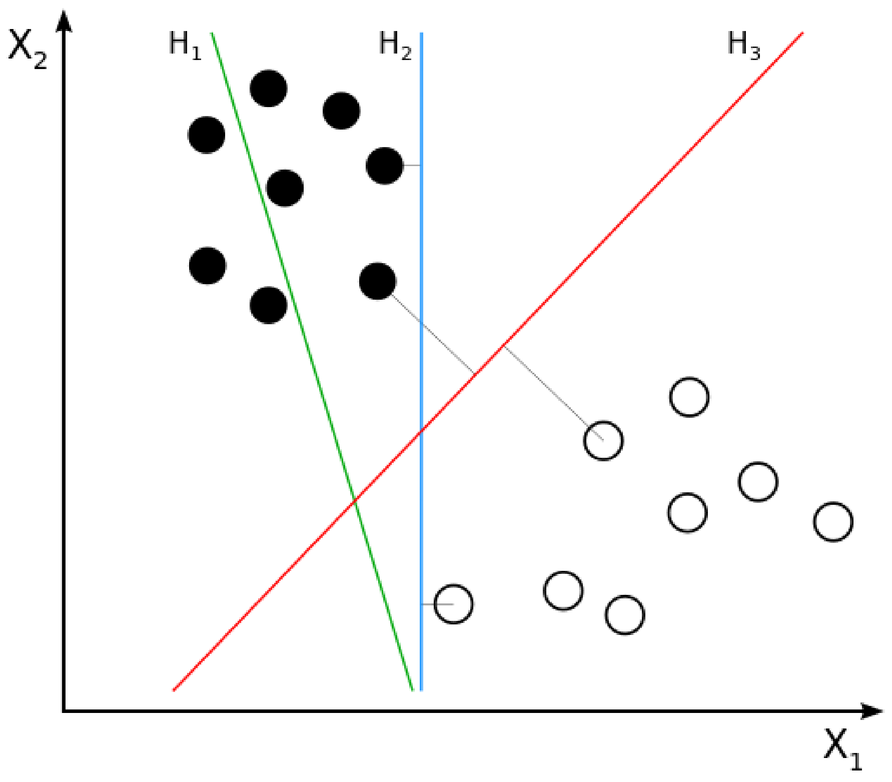


Figure 2.1: Separates two classes linearly by different hyper-planes with a support vector machine. H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximum margin. Adopted from [wikipedia 2012].

SVM Non-Linear Classifier:

In the real world, data-sets can be dispersed up to some extent, and are not linearly separable in a finite space. In 1992 [Boser, Guyon and V. N. Vapnik 1992] proposed the kernel trick, a way to create non-linear classifiers by maximum-margin hyper-planes. Kernel tricks implicitly project the data from a lower dimension into a higher feature space and can be written exclusively in terms of dot products between examples. If we replace x from Equation 2.1 with the output of a feature function $\Phi(x)$, then the dot product with a function $k(x, x^i)$ $=\Phi(x)^T.\Phi(x^i)$ can create a kernel function. Kernels allow the algorithm to fit the maximum-margin hyper-plane in a transformed feature space. By replacing dot products with kernel functions, we can make predictions using the equation 2.2.

$$f(x) = b + \sum_i \alpha_i k(x, x^i) \tag{2.2}$$

This function is linear in the space that Φ maps to, but non-linear as a function of x.

Besides a linear classifier, we compare two widely-used kernels, namely the sigmoid and radial basis function (RBF). They are defined as follows.

- Sigmoid kernel:
$$k(x, x^i) = tanh(\gamma||x - x^i||^2) \tag{2.3}$$
- Gaussian Radial Basis Function kernel: The most commonly used kernel in wind speed prediction
$$k(x, x^i) = exp(-\gamma x^T - x^i + r) \tag{2.4}$$
where $||.||$ denotes the 2-norm, and γ must be greater than 0.

Figure 2.2 shows how kernel functions are used to compute a non-linearly separable function into a higher dimension linearly separable function.

2.2 Support Vector Regression

SVM was initially employed to solve the classification problem, and later on, in 1997, it was applied to the regression field and referred to as support vector regression (SVR) by Vapnik,

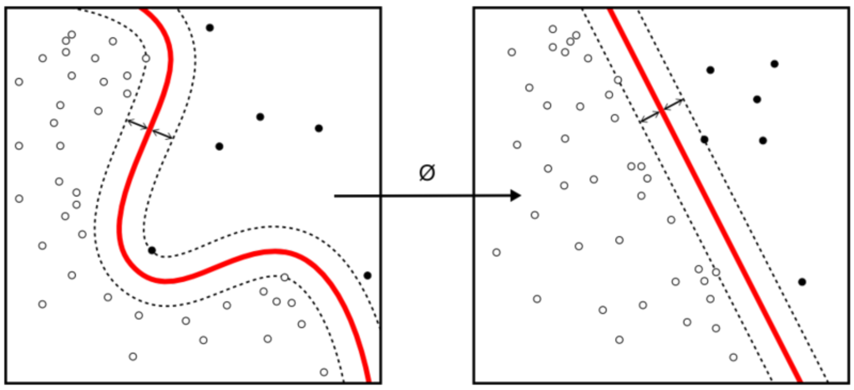


Figure 2.2: Non-linear SVM classifier. Adopted from [wikipedia 2011].

Steven Golowich, and Alex Smola [Vladimir Vapnik, Golowich and Smola 1997]. In this type, SVR predicts a numerical value.

SVR uses the same principles as the SVM for regression, with only a few minor differences. The main idea is to minimize error by individualizing the hyper-plane, which maximizes the margin. Nevertheless, SVR also supports linear and non-linear regression. As the SVR output is a real number, it becomes challenging to predict the information at hand, which has infinite possibilities. In this case, a margin of tolerance (ϵ) is set in approximation to the SVR. The goal of SVR is to find a function $f(x)$ that has at most a ϵ deviation from the targets for all the training data [J.Smola and Schölkopf 1998]. In other words, errors must remain within the ϵ, and therefore, should not accept deviations more significant than this. Using the ϵ intensive loss function, we ensure the global minimum and, at the same time, optimization of reliable generalization boundaries. As we discussed in SVM, the linear function is in the form of

$$f(x) = w^T x + b \tag{2.5}$$

Where with a small w, flatness occurs, i.e., a smaller slope. One way to ensure this is to minimize the euclidean $norm,^2$, i.e $||w||^2$. Formally we can write this problem as a convex optimization problem by requiring

$$minimize : \frac{1}{2}||w||^2 \tag{2.6}$$

$$Constraints : \begin{cases} y_i - wx_i - b \leq \varepsilon \\ wx_i + b - y_i \leq \varepsilon \end{cases}$$

The assumption in 2.6, function f, actually exists that approximates all pairs (x_i, y_i) with ε precision, or in other words, that the convex optimization problem is feasible.

In order to cope with the infeasible constraints of the optimization problem in equation 2.6, Vapnik introduced slack variables ξ_i, ξ_i^*.

$$minimize: \frac{1}{2}||w||^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \tag{2.7}$$

$$Constraints: \begin{cases} y_i - wx_i - b \leq \varepsilon + \xi_i \\ wx_i + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

w is the learned weight vector, x_i is the i-th training instance, y_i is the training label, and ξ_i is the distance between the boundaries and the predicted values outside the boundaries. The constant C>0 determines the trade-off between the flatness of f and the extend of derivation larger than ϵ being tolerated, thereby reducing over-fitting. Figure 2.3 depicts the defined optimization problem where the data points represent the predicted values $(\hat{y})$, and the line represents the label (y) data. The two dashed lines are the bounds that are in an ϵ distance of the reference data, where ϵ is an arbitrarily chosen parameter. The SVR algorithm uses only values outside the dashed lines to optimize the model.

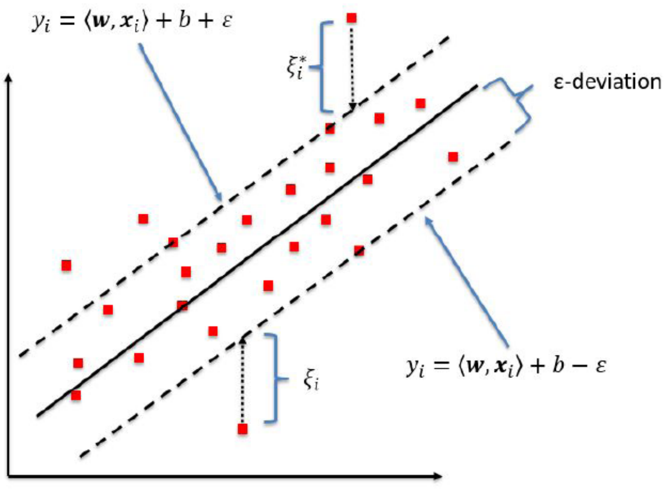

$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle \mathbf{w}, \mathbf{x}_i \rangle + b - \varepsilon$

Figure 2.3: Schematic of the one-dimensional support vector regression (SVR) model. Adopted from [Kleynhans et al. 2017].

Chapter 3

Data Analyse

3.1 Data Characteristic

The two data-sets used in this study are Observation and AROME. The data-sets were selected from 28 representative stations in Austria and include different topography characteristics.

Fig 3.1 shows the 273 observation site locations in Austria. Blue points show all of the stations, and the red points are the 28 selected stations.

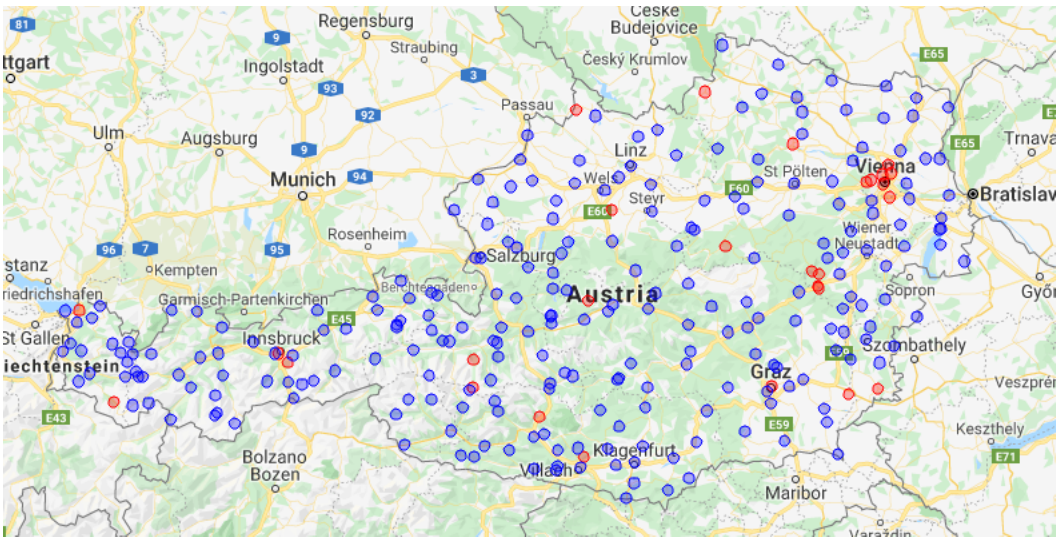


Figure 3.1: Location of Austrian weather stations (TAWES network).

3.1.1 Observation Data

In various regions, the meteorological data are obtained by insite-observation or remote-sensing. In 1980 the observing weather stations became semi-automatic weather stations (TAWES) and climate stations. The semi-automatic weather stations (TAWES) operated by ZAMG, take measurements of air temperature, wind speed, wind direction, pressure, and relative humidity and transmit the data with a frequency of 10 minutes. Table 3.1 shows the parameters in Observation data, their data types, and units.

Table 3.1: Observation data description. In the unit column, **m/s** refers to meter/second, **mm** refers to millimeter, **hPa** refers to hectopascal (1 hPa = 100 Pa).

Attributes	**Data type**	**Unit**	**Description**
Statnr	int		station number
Datum + Stdmin	int		observed timestamp
Tl	int	°C	observed air temperature
Rf	int	percent	observed relative humidity
Td	int	°C	observed dew temperature
Dd	int	1-360 °	observed wind direction
Ff	int	m/s	observed wind speed
rr	int	mm	observed precipitation
P	int	hPa	observed pressure
Pred	int	hPa	observed reduced pressure
So	int	seconds	observed sunshine duration
Ffx	int	m/s	observed gust
Ddx	int	1-360 °	observed gust direction
Ts	int	°C	observed soil temperature

3.1.2 AROME Data

In Austria, the current operational NWP model is AROME (Application of Research to Operations at Mesoscale). AROME is a mesoscale NWP model, developed at Meteo-France since December 2008. It was designed to predict small-, or mesoscale (10 to 100 km) weather phenomenon (e.g., thunder-storms, squall-lines, mesoscale convective systems). Thus, this model has a spatial resolution of at least 2.4 km and hourly temporal resolution, which executes eight times a day (00, 03, 06, 09, 12, 15, 18, and 21 UTC) and the output is, at the earliest, available after three hours of computation. AROME produces a 48-hour forecast to support the very short-range forecasting in Austria [Seity and Brousseau 2011]. Table 3.2 shows the description of AROME data.

3.2 Data Preprocessing

Data cleaning is the process of correcting or deleting data or manually processing them as needed to prevent the error from happening. Also, ensuring that our data is correct, consistent, and

Table 3.2: AROME data description. In the unit column, **m/s** refers to meter/second, **mm** refers to millimeter, **hPa** refers to hectopascal (1 hPa = 100 Pa).

Attributes	**Data type**	**Unit**	**Description**
Statnr	int		station number
Datumagl + Stdmagl	int		initialized timestamp
Stdragl	int		lead time in hours
Datum + Stdmin	date		forecasted timestamp
Rrr	int	mm	predicted precipitation
Tl	int	°C	predicted air temperature
Ff	int	m/s	predicted wind speed
Dd	int	1-360 °	predicted wind direction
Rf	int	percent	predicted humidity
P	int	hPa	predicted pressure
Pred	int	hPa	predicted reduced pressure

usable by identifying any errors in the data. Old and inaccurate data can have an impact on results. In many cases, more accurate data beats fancier algorithms. Different types of data will require different types of cleaning. For this thesis, some data cleaning steps, like quality control of data-sets, dealing with missing values, outliers detection, and scaling the features, were made.

3.2.1 Quality Control

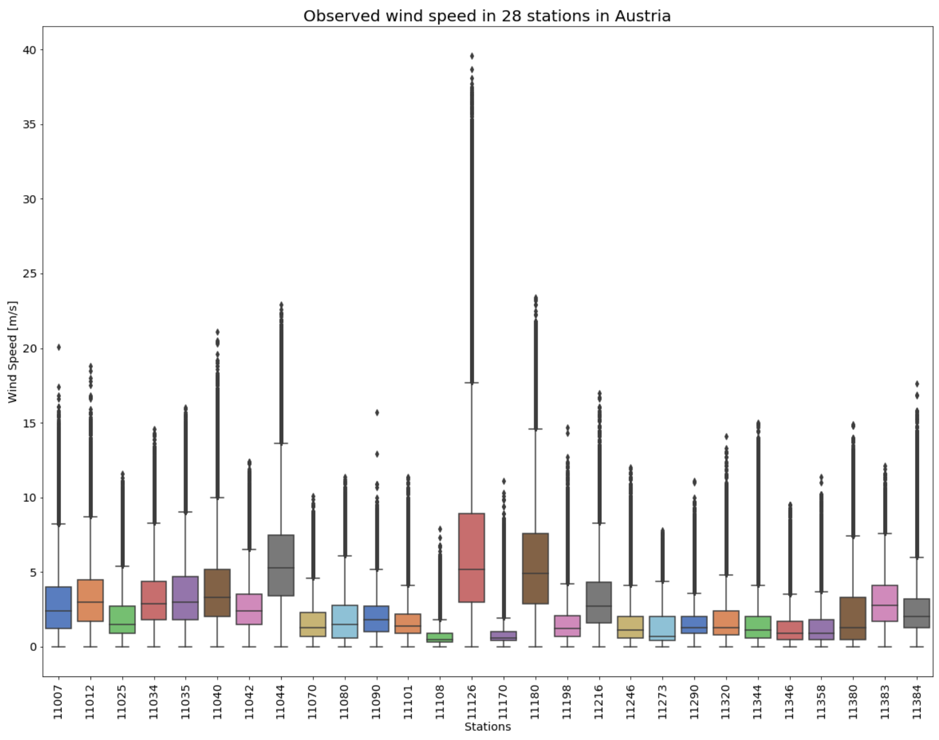
In the first step, we examined and then cleaned the data from errors. For instance, errors in the data arise during measurement or data transfer. Generally, the SVR method can use this imperfect record for predictions but can skew a model to discover the wrong patterns in the data-sets. Both AROME and Observation data had significant errors. The names of the features were based on ZAMG format (the first column of Tables 3.1 and 3.2), which we changed into a more compact and problem specific data format. Both data-sets, provided by ZAMG, were in *.asc* format which we convert to *.csv* files.

Observation data:

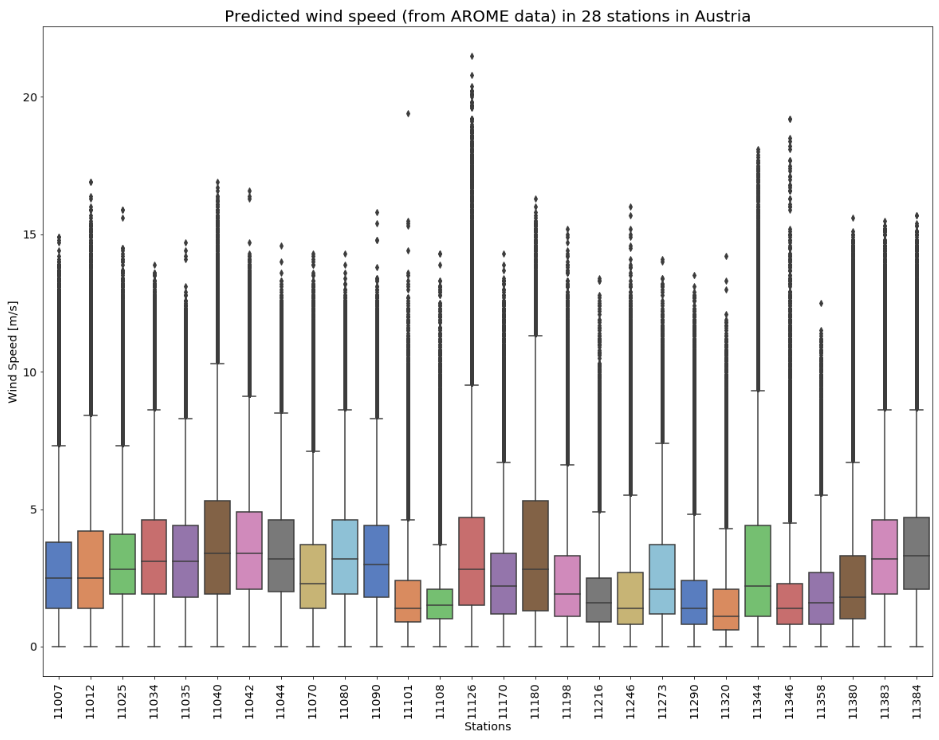
Total instances for Observation data were $4,988,639$. High dimensionality and a skewed distribution of wind speed between stations in Fig 3.4 indicated that we needed to divide the Observation data before data cleaning. Fig 3.2a shows the boxplot of observed wind speed by 28 stations. The plot shows that the distribution of wind speed can be very different due to the region (e.g., mountains vs. urban). Station Wien-Jubiläumswarte (station number 11044) shows a higher variability for wind speed area_mean in the Vienna urban area; it is because this station is located at a higher altitude (450 meters) than the other stations in Vienna. We can find the description of each station in Table 3.3. Station Gaschurn in Vorarlberg (station number 11108) shows less wind speed area_mean. Also, the mountain station Patscherkofel (station number 11126) contains too many outliers and can be challenging for predictions.

AROME data:

The total data-set resulted in $13,704,288$ instances, including performed outlier detection. An outlier may occur due to variability in the measurement or by experimental error. Outliers can cause severe problems in statistical analyses, so their elimination is necessary. After removing the outliers, the maximum predicted wind speed in AROME data showed 21 m/s, which is reasonable (before it was up to 1400 m/s). The AROME data is also divided into stations due to high dimensionality and the differences between single stations in wind speed behavior. The number of instances for each station in Observation and AROME data is presented in Table 3.3. Fig 3.5b shows the boxplot of AROME wind speed by 28 stations. The distribution of wind speed indicates a similar behavior as in the Observation data. For example, station 11126 shows a high number of outliers compared to other stations. Stations in the equivalent region, like Vienna (stations 11034, 11035, 11040, 11042, 11044, 11080, and 11090), show the same distribution as in the Observation data-set.



(a) Observed wind speed [m/s] of 28 stations in Austria.



(b) Predicted wind speed [m/s] of 28 stations in Austria (AROME data).

Figure 3.2: Observed and predicted wind speed of 28 stations in Austria

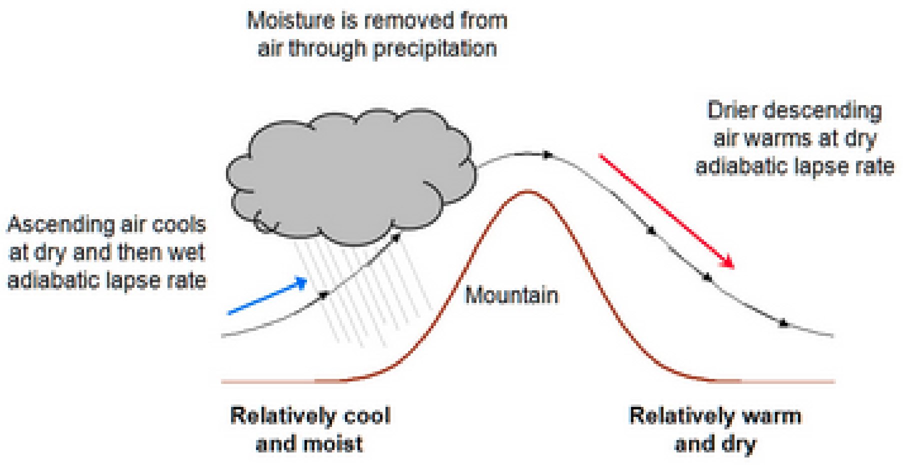

Moisture is removed from air through precipitation
Drier descending air warms at dry adiabatic lapse rate
Ascending air cools at dry and then wet adiabatic lapse rate
Mountain
Relatively cool and moist
Relatively warm and dry

Figure 3.3: Schematic illustration of Föhn.

The histograms in Fig 3.4 show the different distribution of observed wind speed (Observation data) and predicted wind speed (AROME data) for four stations. Stations 11034 and 11044 are in Vienna; station 11108 is in a village area in Gaschurn, Vorarlberg, and station 11126 is in the region of Tirol on a mountain top named Patscherkofel with an altitude of 2251 meters. Fig 3.4 shows that wind is not normally distributed but skewed to the right, i.e., low wind speeds are dominant. In all of the stations, the AROME model predicted a higher density of wind speed. We see higher density for lower wind speed in station 11108.

Station 11126 is located in the Alpine region, where Föhn conditions prevail. Föhn (foehn or foehn wind in English) frequently occurs in the Alps and is a warm, dry downslope wind on the lee side of mountain ranges (see Fig 3.3). When it blows, the leeward mountain slope's temperature rises; the topography of mountainous regions forces the air to flow uphill, which leads to the formation of clouds and often leads to precipitation on the windward side of the mountain. Thereby, the air loses its moisture and gains heat. Turbulence over the mountain removes moisture from the air and increases its temperature. Hence, the now dry air flows downslope on the lee side and is heated by solar radiation (adiabatic warming). Föhn can be very turbulent and gusty and, thus, remixes the air of valleys if it breaks through the inversion layer [Heimann et al. 2007; AFAC Research Dissemination Pilot Study 2012]. This weather phenomenon can cause faster wind speed. As we see in the Fig 3.4, observation wind speed shows wind speed even more than 35 m/s, which is much faster than other stations. Both data-sets, AROME, and Observation were from 2015-01-01 until 2017-01-31.

Table 3.3: Selection of 28 stations: number of attributes for AROME and Observation data (OBS). In the *State* column, the words represent: **OOE**: Oberösterreich , **NOE**: Niederösterreich, **WIE**: Wien, **VBG**: Vorarlberg, **TIR**: Tirol, **BGL**: Berchtesgadener Land, **KNT**: Kärnten, **STMK**: Steiermark, **SAL**: Salzburg.

Station	**Station Name**	**State**	**Altitude (m)**	**AROME**	**OBS**
11007	KOLLERSCHLAG	OOE	714	428,254	155,933
11012	KREMSMÜNSTER	OOE	382	428,254	155,909
11025	WEITRA	NOE	572	428,254	155,942
11034	WIEN-INNERE STADT	WIE	177	428,254	155,930
11035	WIEN-HOHE WARTE	WIE	198	428,254	155,883
11040	WIEN-UNTERLAA	WIE	200	428,254	155,919
11042	WIEN-STAMMERSDORF	WIE	191	428,254	155,923
11044	WIEN-JUBILÄUMSWARTE	WIE	450	428,254	155,921
11070	KREMS	NOE	203	428,254	155,805
11080	WIEN-MARIABRUNN	WIE	225	428,254	155,934
11090	WIEN-DONAUFELD	WIE	160	428,254	155,645
11101	BREGENZ	VBG	424	428,254	155,951
11108	GASCHURN	VBG	976	428,254	155,936
11126	PATSCHERKOFEL	TIR	2251	428,254	155,918
11170	LINZ	NOE	612	428,254	155,929
11180	RAX/SEILBAHNBERGSTATION	NOE	1547	428,254	155,915
11198	GÜSSING	BGL	215	428,254	155,941
11216	KANZELHÖHE	KNT	1520	428,254	155,898
11246	FÜRSTENFELD	STMK	271	428,254	155,911
11273	GMÜND	KNT	738	428,254	155,929
11290	GRAZ-UNIVERSITÄT	STMK	367	428,254	155,884
11320	INNSBRUCK-UNIV.	TIR	578	428,254	155,897
11344	KOLM SAIGURN	SAL	1626	428,254	155,775
11346	RAURIS	SAL	934	428,254	155,928
11358	BAD MITTERNDORF	STMK	814	428,254	155,925
11380	REICHENAU/RAX	NOE	488	428,254	155,929
11383	SEMMERING	NOE	988	428,254	155,924
11384	HIRSCHENKOGEL	NOE	1318	428,254	155,924

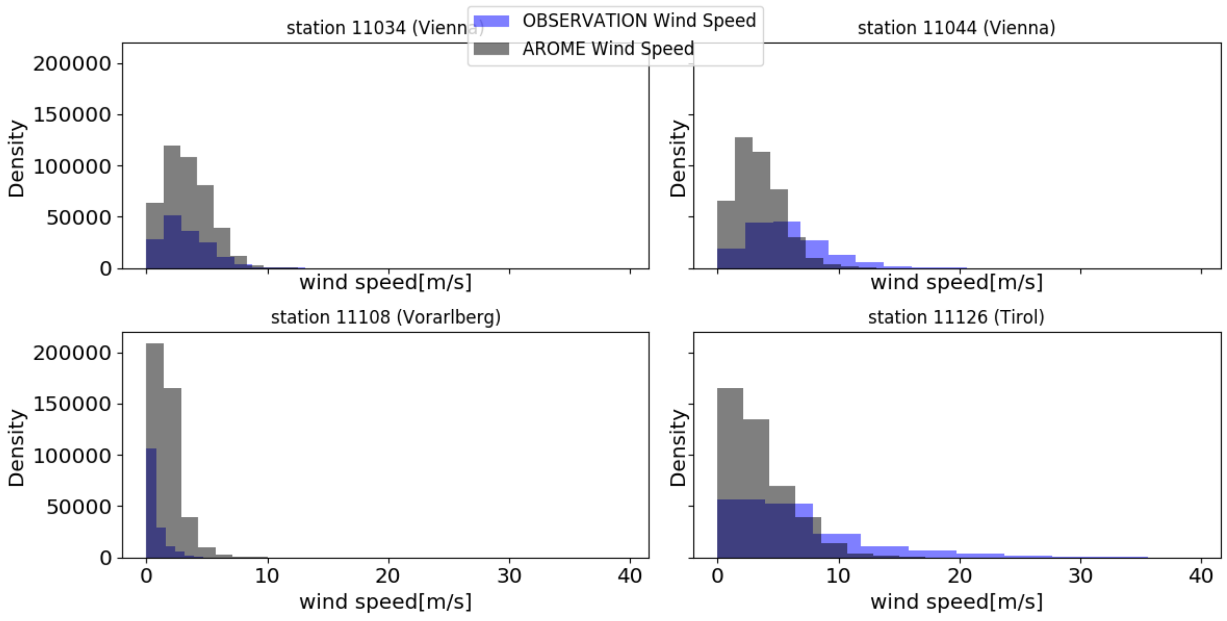


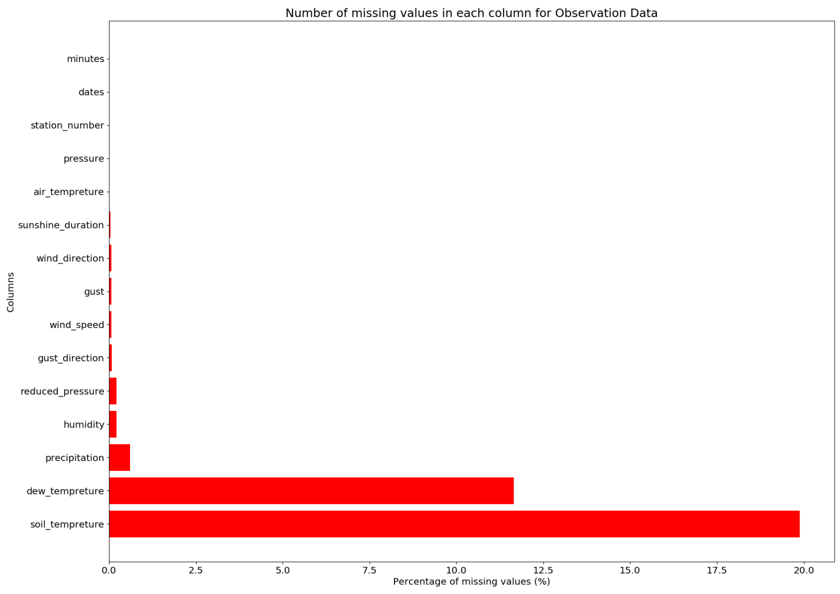
Figure 3.4: Histograms of Observation and AROME wind speed for stations 11034, 11044, 11108, 11126.

3.2.2 Dealing with Missing Values

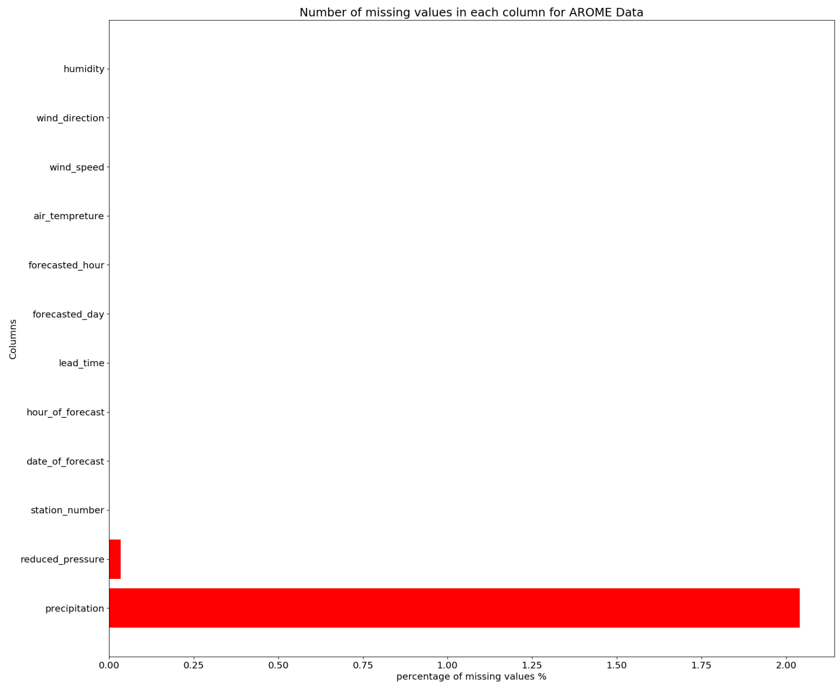
SVR cannot control missing values in the data and need to be handled. Data containing missing values is usually due to machine failure, routine maintenance, and human errors. Both AROME and Observation data had missing values filled with -999 or -998 values. These values impact the wind speed predictions and should be converted to an appropriate value acceptable by SVR. Fig 3.5 shows the percentage of missing values for each column. The *pressure* column from AROME data had no entries, and we removed this column. Other columns with missing values were transformed into no value first in order to apply different missing values methods, like dropping and imputation.

Dropping:

Drop the rows that have missing values. Dropping missing values is sub-optimal because by dropping observations, we lose information. The most significant advantage of this method is its simplicity. It is always reasonable to use it when the number of dropped observations is relatively small compared to the total [C.M. et al. 2016].



(a) Missing values of each column for Observation data



(b) Missing values of each column for AROME data

Figure 3.5: Percentage of missing values of each column for AROME and Observation data.

Imputation:

Impute the missing values based on other observations. Imputing missing values is also sub-optimal because it reduces variability, which can leads to a loss in information, no matter how sophisticated our imputation method is. However, the imputation method, i.e., the *Median*, is more robust in the presence of outliers in the observed data.

Missing values imputation is performed through the random forest (RF) algorithm, then the performance of the RF algorithm using different techniques, *Constant*, *Mean*, *Median* and *Most Frequent* were compared. The imputation performance of RF is computationally efficient and, therefore, not computationally time-consuming due to the use of high dimensional data-set.

- **Constant**: Replace missing values with a value. It can be with strings or numeric data. In this project, *zero* used for the fill-value.
- **Mean**: Replace missing values using the mean along each column.
- **Median**: Replace missing values using the median along each column.
- **Most Frequent**: Replace missing values using the most common value along each column.

Wind speed is the only feature from AROME data that we consider for further analysis, and because there are no missing values in the wind speed feature, handling missing value was not needed. For Observation data, the result of handling missing values shows in Fig 3.6 that we get better forecast results, i.e., less mean square error (MSE), by dropping the missing values. Here is how we proceeded with the imputation and the evaluation in Observation data:

1. Select all of the data-set including missing values.
2. Reconstruct the missing data using the mentioned imputation methods.
3. Apply RF algorithm on the altered data-sets with the artificially imputed missing values using different methods (with five-fold cross-validation) and compute the forecasting performance of each method (MSE).
4. Plot the MSE for each method, which is shown in Fig 3.6.

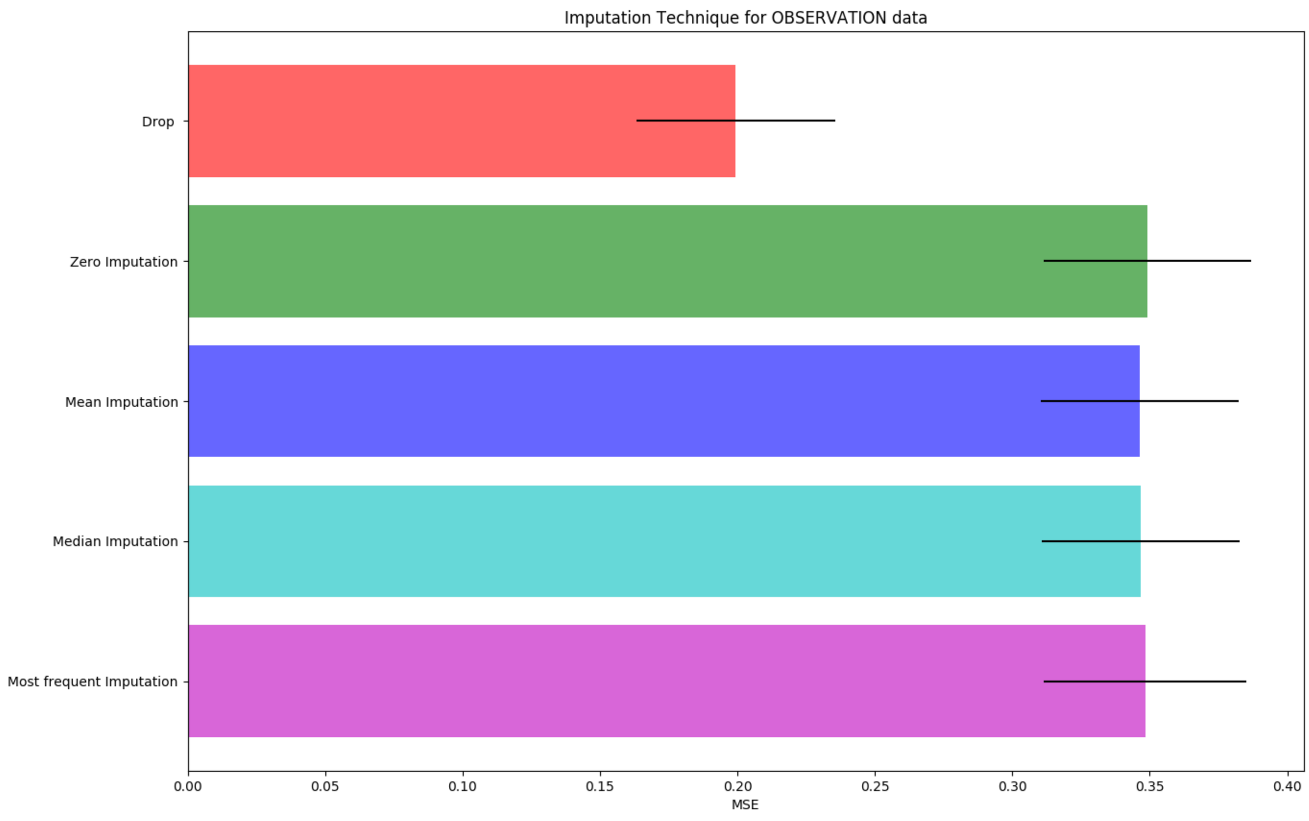


Figure 3.6: Comparing missing values methods for Observation data.

Fig 3.6 indicates that dropping missing values yields sufficiently reasonable results, and additionally it is very fast. However, by dropping missing values 8 stations (particularly 11036, 11126, 11180, 11343, 11344, 11358, 11380, 11383) were excluded from data-set. In Table 3.4 we can see station numbers 11383, 11380, 11358, 11343, 11344, 11126, 11180 are completely empty in *dew temperature* and *soil temperature* columns. The correlation between features was checked in order to preserve these stations. *Dew temperature* and *soil temperature* have a high correlation (more than 80%), with *air temperature*. The heat-map plot for stations 11383 and 11343 is shown in Fig 3.7 and Fig 3.8 respectively. The color bar next to the figures represents the correlation between features on the color scale. High correlation tends to dark blue, and low correlation tends to light yellow color. Fig 3.7 shows the correlation of features in station 11343. We see *air temperature* is almost 80% correlated with *dew temperature*, and in Fig 3.8, station 11383 has a more than 90% correlation between *air temperature* and *soil temperature*. The same behavior has been seen in the rest of the stations. These results indicate that by eliminating *soil temperature* and *dew temperature*, simplify the data-set, and are able to keep the aforementioned

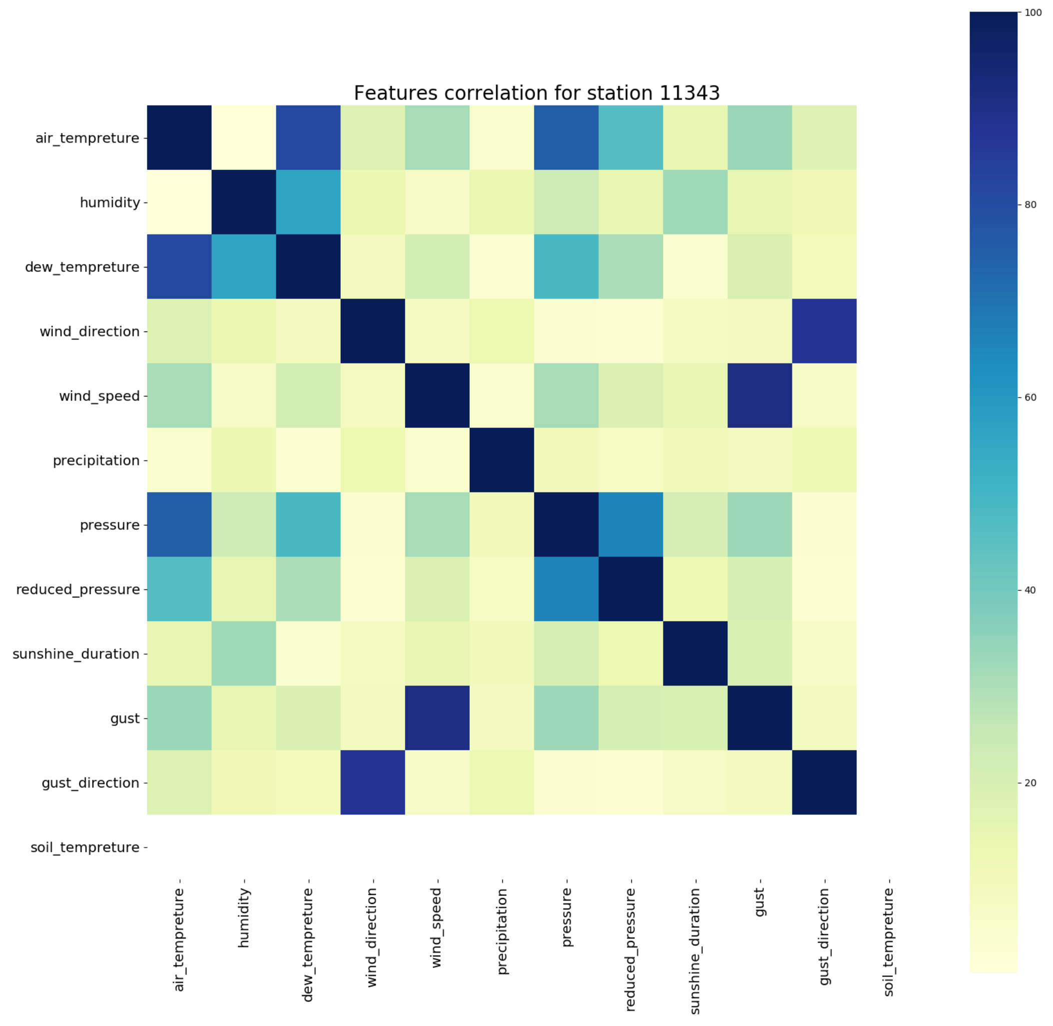


Figure 3.7: Correlation in station 11343.

target sites. On the other hand, station number 11036 has 100% empty *gust direction*, and station number 11343 has 63.05% missing values in *precipitation*.

After excluding stations 11036 and 11343, and removing *soil temperature* and *dew temperature* from data-sets, the highest missing values reduced to 0.6% and 0.2% which is for *precipitation* in Observation and AROME data respectively. Dropping the missing values is sufficient, because there is yet enough data with various dates and times for modeling the SVR method, and doing so much is beyond the scope of the thesis.

Both AROME and Observation data had 30 stations at first, but due to missing-values issues, we neglect station 11343, 11036 in our further analysis, and continued with 28 stations.

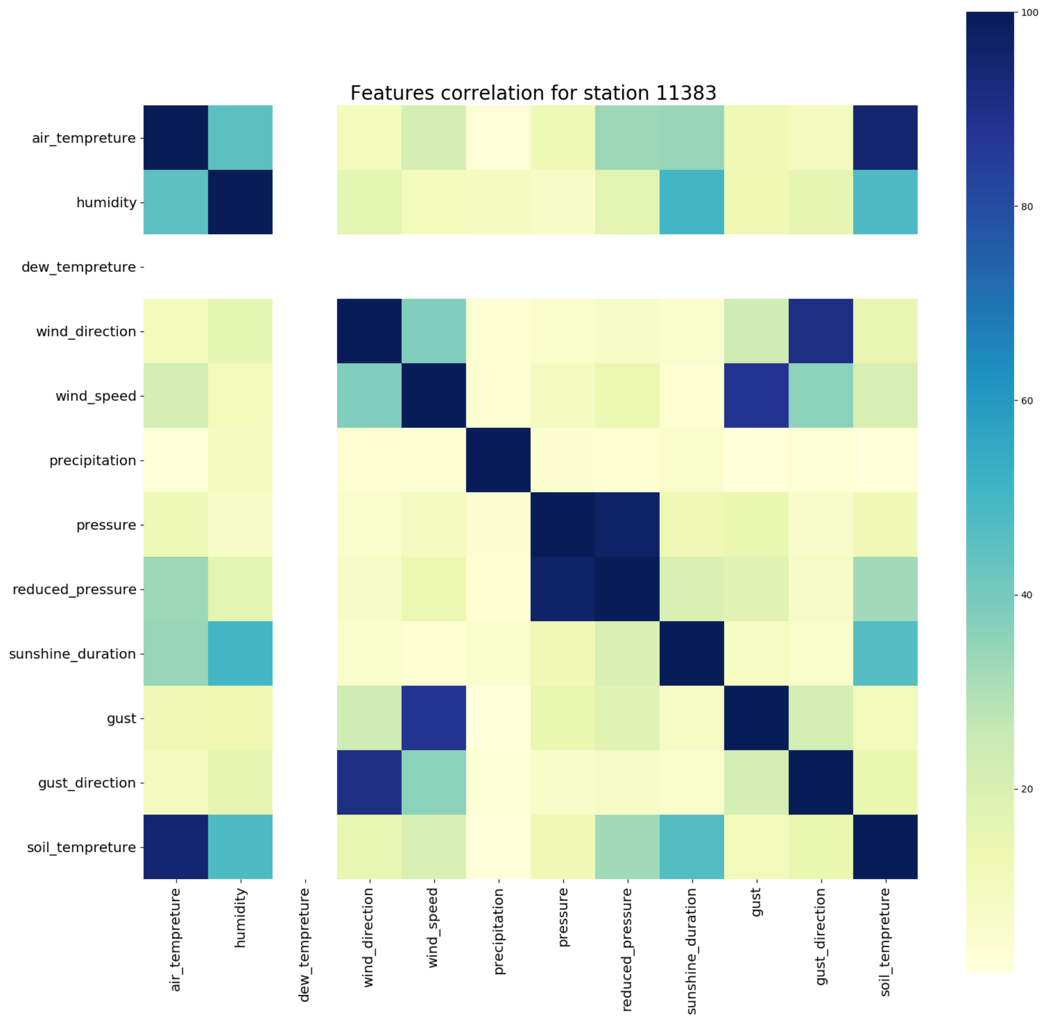


Figure 3.8: Correlation in station 11383.

3.2.3 Feature Scaling

Before the model training stage, the data processed by the SVR algorithm should be normalized to the [0,1] interval. The main advantage of scaling is that it avoids attributes in higher numeric ranges, dominating those in smaller numeric ranges. Feature scaling avoids numerical difficulties during the calculation because kernel functions for SVR usually depend on the inner products of feature vectors [Hsu, Chang and C. J. Lin 2003]. It enables the algorithm to converge faster.

Experiments using SVR with and without scaling methods were conducted. The scaling methods are standard-scaling, min-max-scaling, robust-scaling, and normalizing.

Table 3.4: Percentage of missing values in Observation data.

Station	dew temperature	gust direction	precipitation	soil temperature
11383	100%	–	–	–
11380	100%	–	–	–
11358	100%	–	–	100%
11090	25.69%	–	–	–
11036	–	100%	–	–
11343	–	–	63.05%	100%
11384	–	–	–	46.71%
11344	–	–	–	100%
11126	–	–	–	100%
11044	–	–	–	96.78%
11180	–	–	–	100%

Standard-scaling:

One common and widely used scaling method is standard-scaling [A. Y. Lin, Zhang and Selpi 2017]. The idea is to standardize features by removing the mean and scaling to unit variance. For a given data-set X, a standard-scaling data-set (X_{SS}) is typically done via the following equation:

$$X_{SS} = \frac{X - mean(X)}{StandardDeviation(X)} \tag{3.1}$$

Min-Max-scaling:

It rescales the data-set X to a fixed range, usually [0,1] or [-1,1], according to

$$X_{MMS} = lb + \frac{X - min(X)}{max(X) - min(X)}(ub - lb) \tag{3.2}$$

Where lb is a lower bound of the range, ub is an upper bound, and X_{MMS} is the normalized data-set.

Robust-scaling:

It is based on the median and the interquartile range (IQR). IQR is the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile). If the data-set X contains many outliers, robust-scaling often gives better results. Robust-scaling is defined as

$$X_{RS} = \frac{X - median(X)}{IQR} \tag{3.3}$$

Where X_{RS} is the normalized data-set.

Normalizing:

It rescales the vector for each sample to have a unit norm, independently of the distribution of the samples. Based on the l2 (or Euclidean) norm with data-set X of length n, we obtain the normalized data-set $||X||_2$ by:

$$||X||_2 = \sqrt{\sum_{i=1}^{n} X_i^2} \tag{3.4}$$

Fig 3.9 shows that Observation data with the scaled feature has better results with min-max-scaling and normalized-scaling. Furthermore, AROME data with min-max-scaling shows a better outcome for scaling the wind speed feature in Fig 3.10. Thus, the normalizing method was used as a scaling method for all further experiments performed in this thesis.

3.3 Feature Engineering

Choosing the right features to train a model can boost the forecasting performance. Creating new features by combining or dividing existing ones is a basic but powerful feature engineering technique. In time-series data, we must choose the best variables as input to predict future time series. In this thesis, lag features were created for both AROME and Observation data. Creating lag features is a classical way for time series forecasting. We use a random number of previous time steps (t-1, t-2, ...) to predict the next time steps (t+1, t+2, ...). To create the shift or lag features, we used PostgreSQL, for faster computations. For Observation data, new features were generated based on current Observation features: *wind speed, wind direction, gust, gust direction, pressure, reduced pressure, air temperature, precipitation, humidity,* and *sunshine duration*. The new features were shifted until the previous 2 hours with 10 minutes frequency, e.g., past 10 minutes, past 20 minutes, past 30 minutes, past 40 minutes, past 50 minutes, past 60 minutes, past 70 minutes, past 80 minutes, past 90 minutes, past 100 minutes, past 110 minutes and past 120 minutes. Therefore, we generate 120 new features. Table 3.5 shows an example of the lag

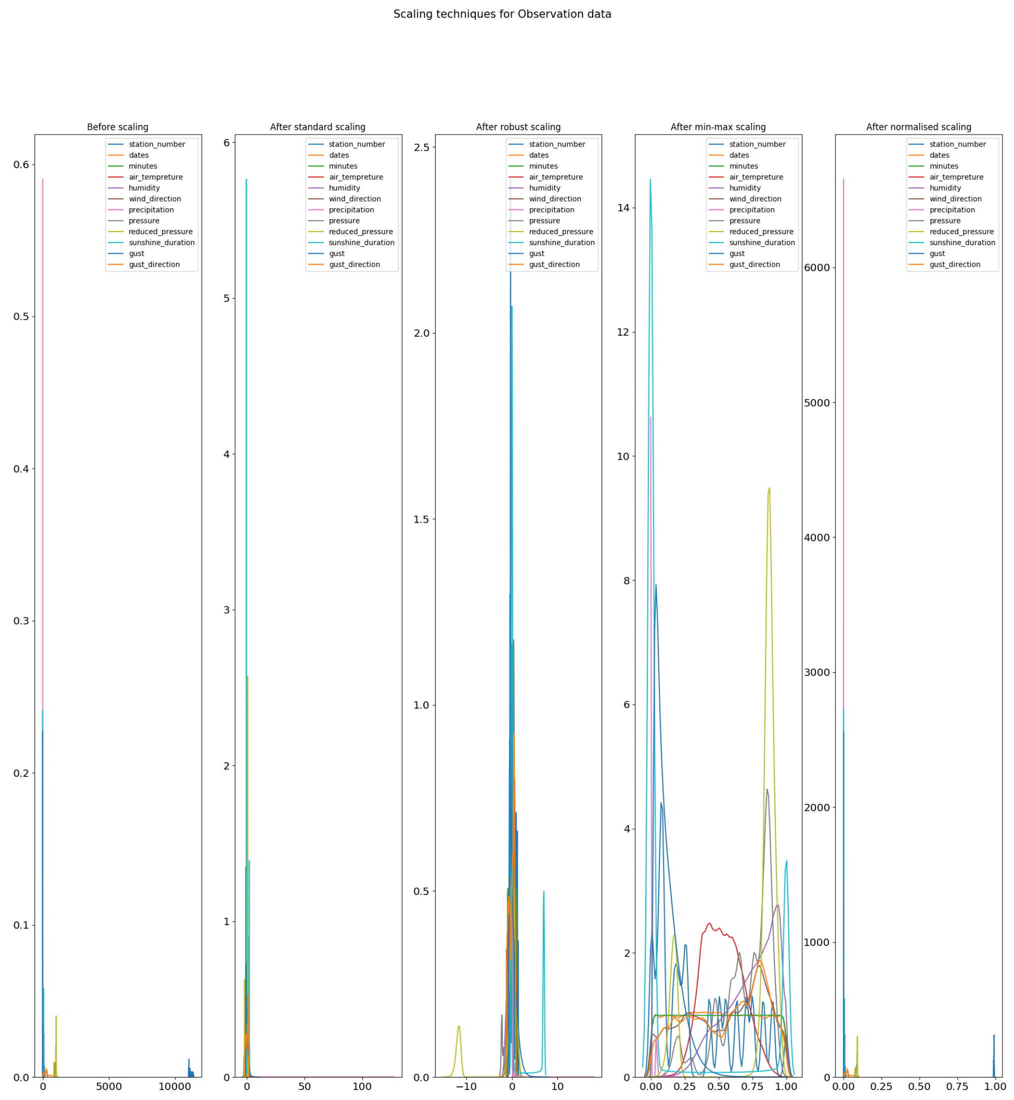


Figure 3.9: Comparing different scaling methods for Observation data.

features for wind speed. Shifting the values creates NaN (unknown) in some entries.

Table 3.5: An example of the new features with lags for Observation data. The column names refer to: **ObsTimeStamp**: observed time-stamp, which is the combination of observation date and observation minute columns, **ObsWindSpeed**: observed wind speed, **Past10mWindSpeed**: past 10 minutes observed wind speed, **Past20mWindSpeed**: past 20 minutes observed wind speed.

ObsTimeStamp	ObsWindSpeed	Past10mWindSpeed	Past20mWindSpeed
2015-01-01 00:10:00	2.6	NaN	NaN
2015-01-01 00:20:00	2.8	2.6	NaN
2015-01-01 00:30:00	3.4	2.8	2.6

The first 11 rows were discarded because of NaN entries (i.e., assigned to not a numeric

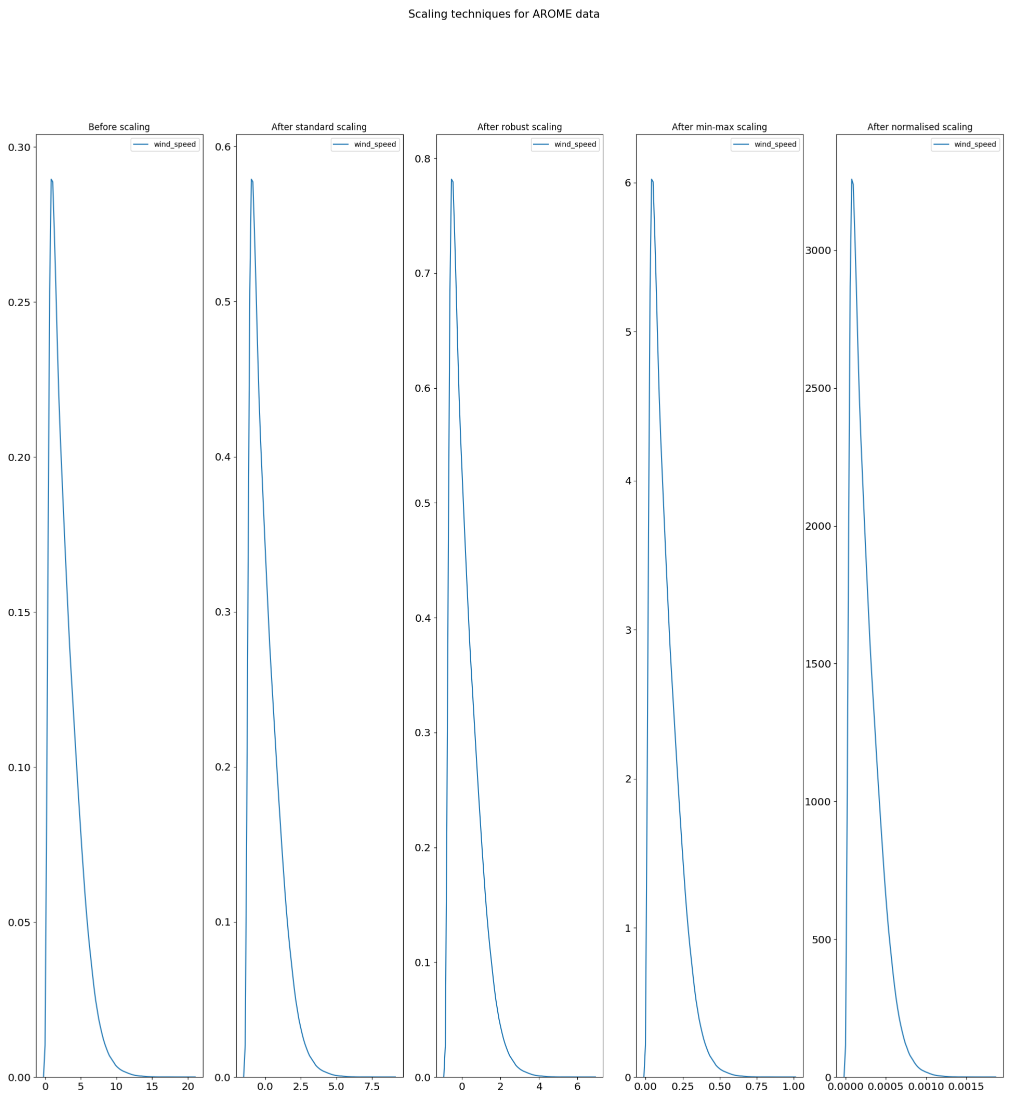


Figure 3.10: Comparing different scaling methods for AROME data.

value) while creating lag features, because we do not have enough wind speed data for observed timestamp from *2015-01-01 00:10:00* until *2015-01-01 02:00:00* in our data-set, for the past 10 minutes wind speed *Past10mWindSpeed* until past 2 hours wind speed *past2hWindSpeed*. After removing NaN values, the first time-stamp in the new data starts from *2015-01-01 02:10:00*. The wind speed feature for 48 hours in the future was shifted (t+1, t+2, ..., t+48) and added to the data-set.

For AROME data, merely wind speed is used for creating lag features. Forty-eight columns were added to the data-set, based on the calculation of wind speed with 4 hours delay. AROME model needs 4 hours after model initialization before it is available for further use. It means when

AROME starts a forecast run at 00 UTC, it takes $\sim$ 3.XX hours to finish, and also 1 hour extra for data reduction and transferring. As we know, AROME initial forecast runs are issued every three hours at 00, 03, 06, 09, 12, 15, 18, 21 UTC, but they are only available four hours after their initial time, at 04, 07, 10, 13, 16, 19, 01 UTC. The finishing time of lagged AROME features is shown in Table 3.6. The number xx UTC in AROME specifies the run, which is eight times per day [Papazek n.d.].

Table 3.6: An example of finishing and initialization times for AROME model run.

time of availability	time of initialization
04:15 - 04:45 UTC	00:00 UTC
07:15 - 07:45 UTC	03:00 UTC
10:15 - 10:45 UTC	06:00 UTC
13:15 - 13:45 UTC	09:00 UTC
16:15 - 16:45 UTC	12:00 UTC
19:15 - 19:45 UTC	15:00 UTC
22:15 - 22:45 UTC	18:00 UTC
01:15 - 01:45 UTC	21:00 UTC

Chapter 4

Modeling

4.1 Machine Learning Workflow

The diagram in Fig 4.1 shows the machine learning workflow of this thesis for one station. The same was carried out for all stations. The machine learning steps used in this thesis are:

1. Gathering data
2. Data pre-processing
3. Select the best model
4. Training and testing the model
5. Evaluation

Gathering data and data pre-processing steps were already discussed in Chapter 3. In this chapter, we discuss the process of selecting the best model.

4.2 Software Details and Environment

For this thesis, python version 3.6 was used as the main programming language. PostgreSQL version 10.6 was used to store the data-sets in order to archive lagged features in Observation and AROME data, joining the two data-sets, and calculating the Persistence and AROME models. Observation and AROME original data were consisted of separated *.asc* format files for each time series and each day of each month from 02015-01-01 until 2017-01-31. To merge the time series files

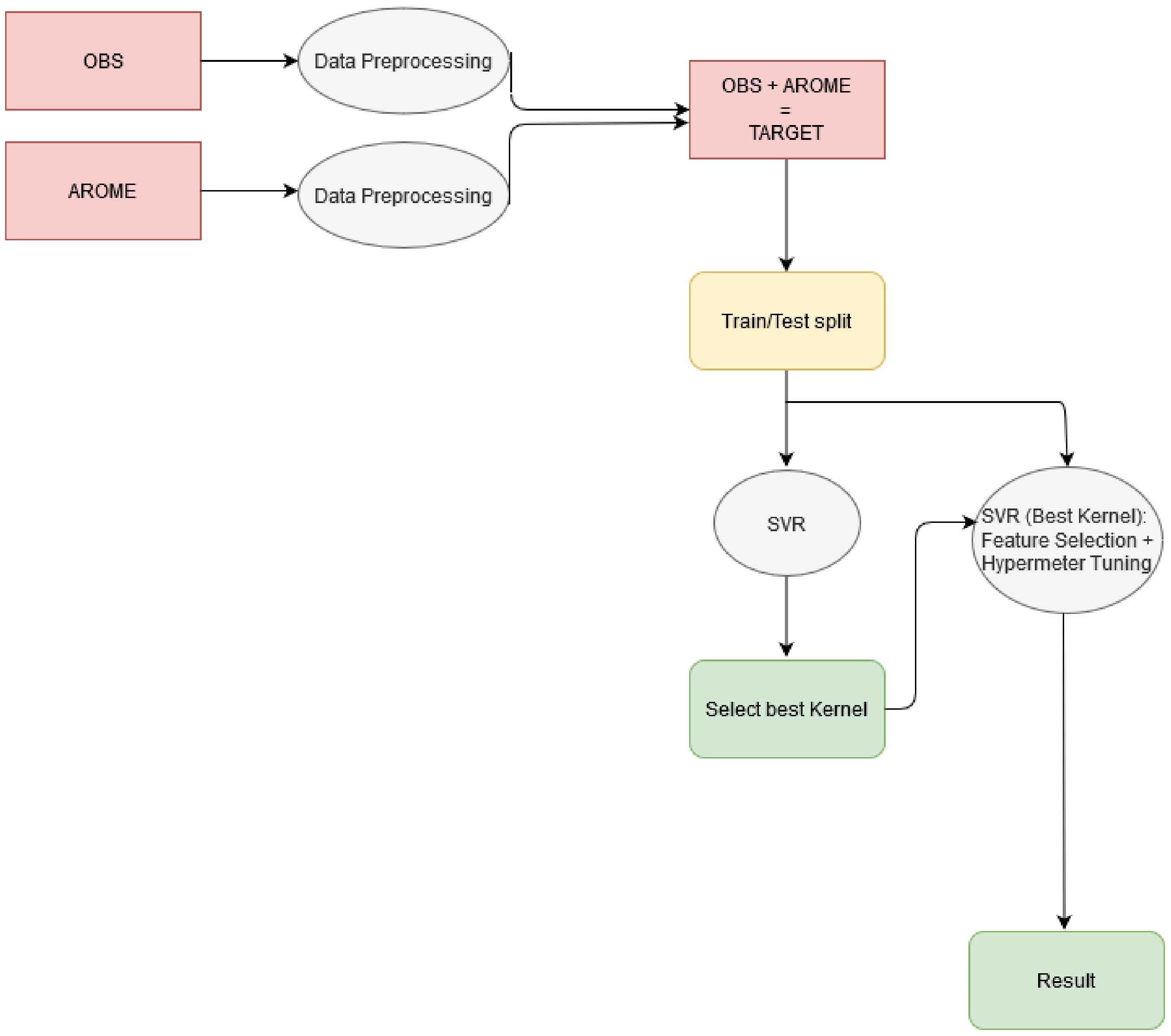


Figure 4.1: Machine learning workflow.

into one file and convert to a *.csv* format file, scripting in Bash was used. As an operating system, Linux - Ubuntu-based version 18.4 - was used, with 12 CPUs and 96 Gigabyte RAM. In addition, we employed Microsoft Azure and the Google cloud to run the SVR method for each lead time for all stations, due to the high dimensional data and lack of high-performance systems. For using cloud systems, pre-processed data were used due to data privacy. The python modules that were used are:

pandas to perform data manipulation

numpy is the fundamental package for scientific computing with python

sklearn for featuring regression algorithms including support vector machines, scaling the data for pre-processing, impute the missing values, feature selection methods, calculating evaluation metrics and hyper-parameter optimization techniques

statistics to use the $mean$ function

matplotlib, **seaborn** and **plotly** for plotting and figures

time to record the time during training and testing

xgboost for the SHAP method. Before one can apply SHAP for feature selection, it is necessary to convert the data to a data matrix format, because of memory efficiency and training speed.

SHAP for feature selection

scipy to calculate the Spearman's correlation metric

psycopg2 is the PostgreSQL database adapter for the python

sqlalchemy creates a connection to the database and reads the data from the database with SQL queries.

4.3 Training and Testing Data-set

It was necessary to join the two AROME and Observation data for each station. The final number of features is 133 in each station, with a combination of shifted and original features.

For faster computation, it is helpful to use cross-validation, which divides the training-set into two sets and validates the model in one set and builds the model in another one. To keep the model unbiased, we used cross-validation just for hyper-meter tuning and trained the models with the full training-set.

The test data contains a time-range from 2017-01-01 until 2017-01-31 (January) for all of the selected stations.

4.4 Evaluation Metrics

In weather forecasting, the forecast performance of different models is evaluated by multiple metrics. For feature selection and hyper-meter tuning, accuracy, MSE (mean square error), and bias were used as evaluation metrics. We used RMSE (root mean square error) for final testing among previous metrics and Spearman's correlation to measure regression performances.

Accuracy:

It indicates the goodness of fit and is usually denoted as R^2, which computes the determination coefficient. It represents the proportion of variance (of y), which has been explained by the model's independent variables. The best score is 1.0. The result can be negative because the model can be arbitrarily worse. If $\hat{y}_i$ is the predicted value of the i sample and y_i is the corresponding real value for total n as the total number of samples, the estimated R^2 is defined as:

$$R^2(y, \hat{y}) = 1 - \frac{\sum\limits_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum\limits_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{4.1}$$

where $\bar{y} = \frac{1}{n}\sum\limits_{i=1}^{n} y_i$.

MSE (Mean Square Error):

This function computes a metric corresponding to the expected value of the squared (quadratic) error or loss. MSE of zero indicates a perfect model with no error. If $\hat{y}_i$ is the predicted value of the i sample, and y_i is the corresponding real value, then the MSE estimated over n as the total number of samples is defined as:

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.2}$$

RMSE (Root Mean Square Error):

It is the square root of the MSE. It has a stronger influence on higher errors than smaller errors, which may be useful if higher errors are especially undesirable. The perfect score is 0.

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.3}$$

Bias:

Bias measures the mean of the difference between the forecasts and the observations. An estimator or a model with zero bias is called unbiased. An unbiased estimator is preferable to a biased estimator, but in practice, biased estimators are frequently used, generally with small bias [Hardy 2002]. A bias value other than zero suggests the model's tendency to be over-fitted (negative bias) or under-fitted (positive bias). If $\hat{y}_i$ is the predicted value of the i sample, y_i is the corresponding real value, and n the total number of samples, then the bias is defined as:

$$BIAS(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{4.4}$$

Spearman's Correlation:

It assesses how well the relationship between two variables, the real and the predicted values, can be. Spearman's correlation assesses monotonic relationships (whether linear or not) [Fieller, Hartley and Pearson 1957]. A perfect spearman's correlation (or identical for a correlation of $+1$) means a similar rank between the two variables, and a dissimilar (or fully opposed for a correlation of -1) rank between the two variables indicates a perfect negative association. Zero correlation indicates no association between the true and the predicted variables. Spearman's correlation is often denoted as r_s. If $\hat{y}_i$ is the predicted value of the i sample, y_i is the corresponding true value, and n the total number of samples, then r_s can be computed using the formula

$$r_s(y, \hat{y}) = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \tag{4.5}$$

where $d_i = y_i - \hat{y}_i$.

4.5 SVR Configuration

Kernel Function:

Among the three standard kernels mentioned in Chapter 2, we must select one kernel, then the penalty parameter C and other kernel parameters. We carry out the following steps:

1. Run the three kernels for station 11007 for 14 lead times (1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 18, 24, 36, 48), then select the best two kernels.
2. Run the best two kernels for the rest of the stations and select that one kernel that performs best in all stations.

The $gamma$ parameter is the kernel coefficient for radial basis function (RBF) and sigmoid, in which the default value is $scale$. The $scale$ value can be calculated like in Equation 4.6, where $n\_features$ is the number of features, X is the data-set and $.var()$ is a *numpy* variance function, which computes the variances for each individual feature. However, $gamma$ can take the value of any float number, and for this part of the thesis, $gamma = 100$ was used. For linear kernel and other parameters of RBF, and sigmoid, the default parameters were used.

$$scale = 1/(n\_features * X.var()) \tag{4.6}$$

In general, in most short-range wind speed forecasting, RBF kernel is a reasonable first choice, because it can handle the non-linearly relations between the labels and the attributes. On the other hand, with a penalty parameter C, the linear kernel can have the same performance as the RBF kernel with parameters (C and $gamma$) since the linear kernel is the special case of RBF [Keerthi and C.-J. Lin 2003]. Also, the sigmoid kernel can have the same behavior as RBF for specific parameters [H.-T. Lin and C.-J. Lin 2003]. From the first run, we noticed that th polynomial kernel does not converge with our data-set, and we neglected it from further experiments. Table 4.1 shows that the sigmoid kernel surpassed by linear and RBF kernels.

One of the biggest challenges was to reduce the computational complexity and run-time. Due to high dimensional data, and lack of a high-performance environment, we focused on 14 selected lead times with more time steps in nowcasting and fewer in short-range. The average of R^2, MSE, bias, and the run-time for training the SVR model are shown in Fig 4.2. Our results show that for the second run, the linear kernel had the best performance in 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 18, 24, 36, 48 lead times. As we aimed to select the best kernel, the linear kernel was chosen for further analysis. The reason for selecting a linear kernel is when the number of features increases, the problem is more likely to be linearly separable, and one may not need to map the data. However, linear kernel function corresponds to a dot product in the feature space, where the number of features can be much larger than the number of dimensions of the input space (in principle even

Table 4.1: Comparison of three kernels, linear, RBF, and sigmoid for 14 lead times - station 11007. Best results of each score are printed bold, worst are underlined.

Lead-Time	Linear			RBF			Sigmoid		
	R^2	MSE	Bias	R^2	MSE	Bias	R^2	MSE	Bias
1-h	-30.02	8.37	-2.14	**-14.95**	**7.40**	**0.98**	-17.63	7.57	1.06
2-h	**-9.76**	**7.06**	-1.86	-14.99	7.39	**0.98**	-17.69	7.57	1.06
3-h	**-0.41**	**6.45**	-1.72	-15.01	7.39	**0.98**	-17.74	7.57	1.06
4-h	**16.63**	**5.35**	-1.47	-15.00	7.39	**0.98**	-17.77	7.56	1.06
5-h	**25.32**	**4.79**	-1.32	-15.03	7.38	**0.98**	-17.81	7.56	1.06
6-h	**35.86**	**4.11**	-1.09	-15.01	7.38	**0.98**	-17.80	7.56	1.06
8-h	**48.72**	**3.29**	**-0.71**	-15.00	7.38	0.98	-17.84	7.56	1.07
10-h	**43.59**	**3.61**	-15.07	-0.84	7.37	**0.98**	-17.93	7.55	1.07
12-h	**41.22**	**3.76**	**-0.79**	-15.12	7.37	0.98	-18.01	7.55	1.07
15-h	**27.5**	**4.64**	**-0.95**	-14.95	7.36	0.97	-17.93	7.55	1.07
18-h	**6.95**	**5.95**	-1.31	-14.99	7.35	**0.98**	-18.05	7.54	1.07
24-h	**37.48**	**3.99**	**-0.72**	-15.04	7.34	0.98	-18.11	7.54	1.07
36-h	**48.65**	**3.29**	**-0.49**	-14.30	7.34	0.95	-17.23	7.52	1.05
48-h	**23.20**	**5.00**	-1.17	-12.87	7.35	**0.91**	-15.81	7.55	1.01

infinite). Using the linear kernel is good enough, and one only searches for the parameter C [Hsu, Chang and C. J. Lin 2003]. In the next section, we optimize the parameter of C by applying an optimization algorithm to the model training stage to obtain higher forecasting accuracy.

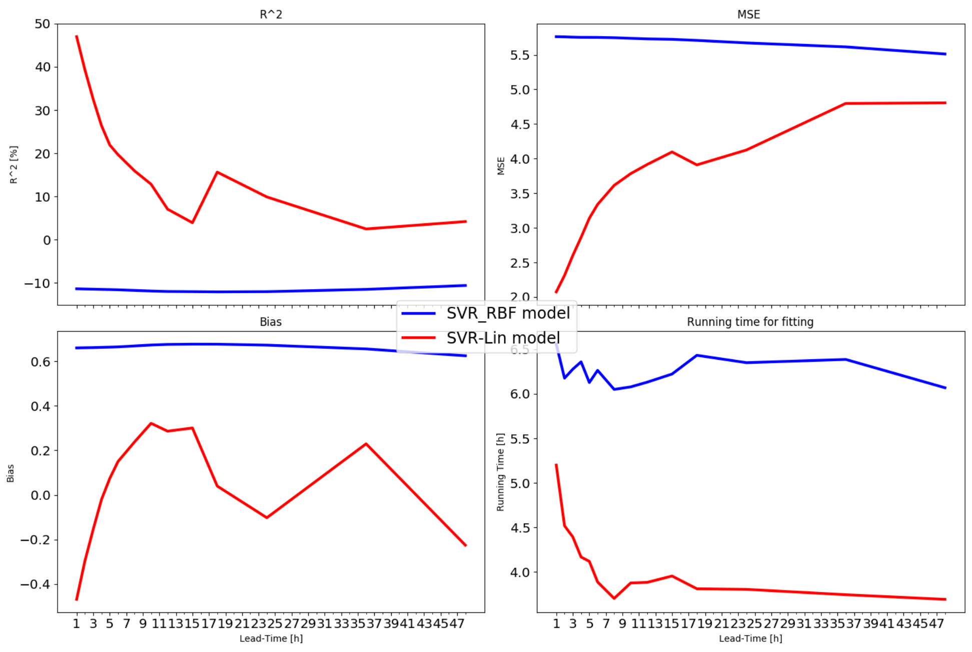


Figure 4.2: Comparison of different metrics for linear and RBF kernels for all stations.

Hyper-parameter Tuning:

Hyper-meters are settings that we can use to control the algorithm's behavior. The values of hyper-parameters are not directly adopted within the learning algorithm itself.

In SVR, we can tune six hyper-parameters: kernels, C (for all kernels), ϵ (for all kernels), sigma (for radial kernel), scale, and degree (only for polynomial kernel). Indeed, the selection of the optimal parameter for a given data-set is an important step to configure SVR since it can cause significant differences in forecast performance.

Scikit-learn library offers two popular methods for fine-tuning the hyper-parameters, *Grid Search* and *Random Search*. One approach for sampling the search candidates is *Grid Search*, and it can be used for tuning all hyper-meters mentioned above. *Grid Search* exhaustively considers all the parameter combinations specified in a grid of parameter values. Hence, a high computational effort is required for large grids. Although such optimization strategy explored quite a large set of parameter combinations, it proceeded by force with predefined steps and did not try to exploit the information collected at best while performing. Moreover, it is computationally expensive [Corazza et al. 2010].

By contrast, *Random Search* sets up a grid of hyper-parameter values and uses random combinations of hyper-parameters to find the best solution for the model. The chances of finding the optimal parameter are higher in *Random Search* because of the random search pattern, and also for lower dimensional data since the time taken to find the right set is less. Although *Grid Search* is computationally too expensive, it is more accurate than *Random Search*. Thus, we decided to continue with *Grid Search* as the hyper-parameter tuning technique for this thesis.

To decrease the length of CPU time for training the linear kernel, we reduced the number of features by using feature selection techniques (See details in section 4.6). We should also note that the cross-validation procedure's random splits can have small differences in score results. Different variations of the training-set can be done by setting the number of cross-validation (CV) iterations in *Grid Search*. The cross-validation procedure can prevent the over-fitting problem [Hsu, Chang and C. J. Lin 2003] and give us a less biased estimate of the error on the testing-set. Five-fold cross-validation was used for hyper-meter tuning. It first divides the training-set into five subsets of equal size. Sequentially one subset is for testing, and the SVR trained on the remaining four subsets. *Grid Search* picks the best cross-validation accuracy for each C value.

The linear kernel takes two parameters, C and ϵ. C is the regularization parameter which controls the *softness* of the boundary margin. Lower values for C draws smoother decision boundaries (less flexible), whereas higher values give more rugged boundaries that can fit the training data better (more flexible) [Academy n.d.] and it must be strictly positive. The parameter ϵ defines a margin of tolerance. It affects the smoothness of the SVR's response, and the number of support vectors. Both the complexity and the generalization capability of the network depend on its value [Horváth 2003]. Intuitively, the value of ϵ can affect the number of support vectors used to construct the regression function. The bigger ϵ is, the fewer support vectors are selected. On the other hand, bigger ϵ-values result in more *flat* estimates. Hence, both C and ϵ-values affect model complexity (but in a different way) [Cherkassky and Ma 2004]. The default value for each parameter is $C = 1$ and $\epsilon = 0.1$. For the coarse grid, different values for C and ϵ were chosen.

C = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
ϵ = [0, 0.1, 0.2, 0.3, 0.5, 0.8, 1]

Due to computational performance issues, we were not able to complete the *Grid Search* (i.e., to optimize the parameters for 1-hour forecasting, it took 146 hours). The run-time analysis was mandatory in this phase. Run-time analysis estimates and anticipates the increase in running time (or run-time) of an algorithm as its input size increases [Wikipedia n.d.]. Each training-set contains more than $145,000$ instances. One idea was to reduce the number of rows by dividing each training-set into years, tune each year on a coarse grid, and then based on the performance scores, tune the full data set again on a finer grid.

The years 2015 and 2016 have almost the same size for training ($\approx 52,600$ rows), but the year 2017 has less training size ($\approx 46,300$ rows) because we use January 2017 as a testing-set. Now each year takes 5 - 6 hours for *Grid Search* to find the optimized parameters.

Fig 4.3 shows the optimized parameters for each year, and for 12 (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) lead times. The x-axis shows the ϵ parameters, and the y-axis shows the C parameters in 2015, 2016, and 2017 plots. In 2015 and 2016, most of the lead times are optimized within $C = 1, \epsilon = 10$ (blue point), and in 2017 optimized parameters are $C = 1, \epsilon = 1$ (orange point). The bottom right plot shows the tuned parameters' scores for the 12 lead times. From the plot, we can see the scores decrease when lead times increase. It is possible to have more optimized results in the first lead times than the others.

To limit the overall computational time, we decided to select a small number of grid values. For the final grid search on the full data-set, the chosen grid is:

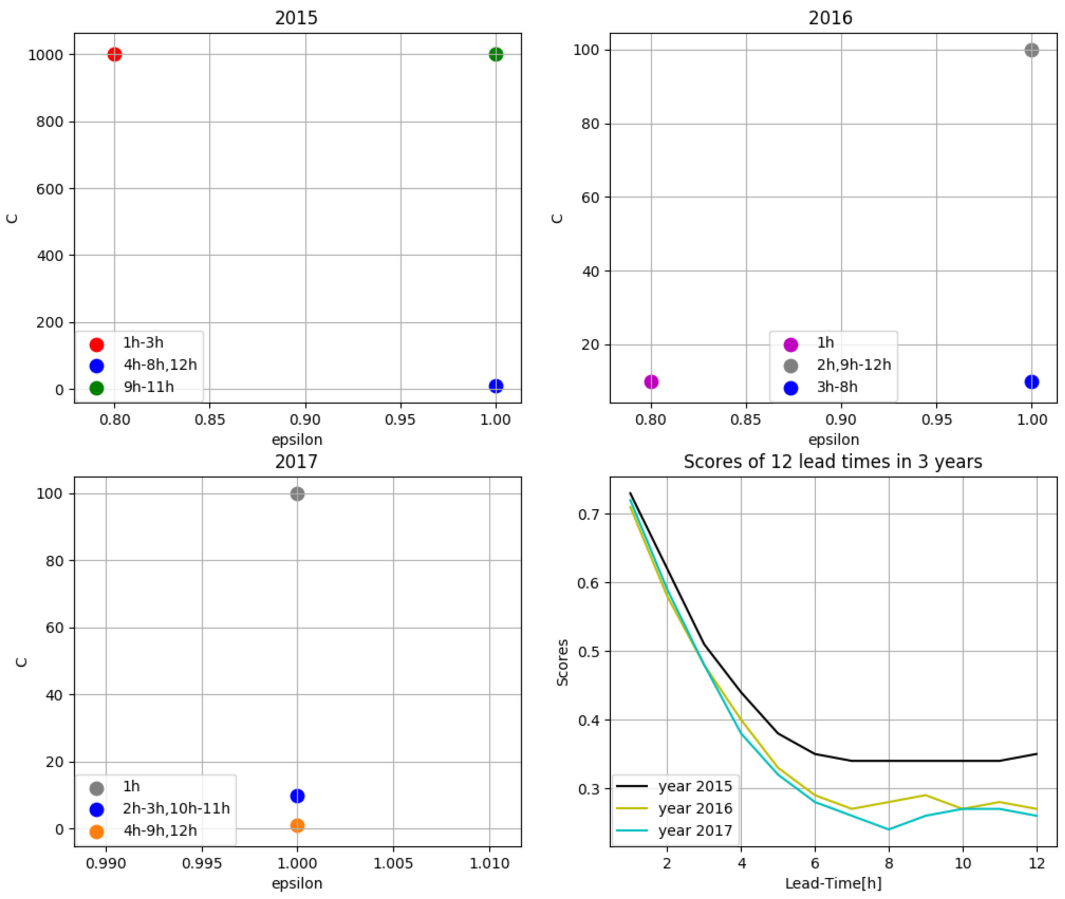


Figure 4.3: Optimized C and ϵ for years 2015, 2016 and 2017 in 12 lead times in station 11007, and scores of hyper-meters for three years.

$$C = [1,\ 10,\ 100,\ 1000]$$
$$\epsilon = [1]$$

Generally, the best performing hyper-parameter combinations are limited to these grid choices. *Grid Search* workflow to select the best parameters and retrain the model is shown in Fig 4.4.

4.6 Feature Selection

Feature selection is the process of selecting the most useful features in a data-set. It is a crucial step in machine learning. The top reasons to use feature selection are to reduce the complexity of a model, which makes it easier for interpretation, and to improve the accuracy of the model. Feature selection reduces the dimensionality of the data and enables machine learning algorithms to be

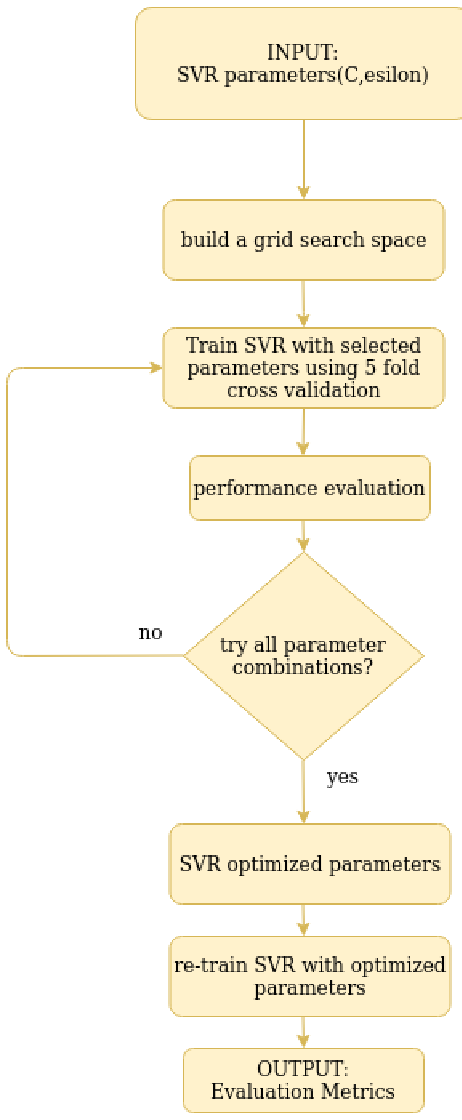

INPUT:
SVR parameters(C,esilon)
build a grid search space
Train SVR with selected parameters using 5 fold cross validation
performance evaluation
try all parameter combinations?
no
yes
SVR optimized parameters
re-train SVR with optimized parameters
OUTPUT:
Evaluation Metrics

Figure 4.4: *Grid Search* workflow to find the optimized parameters of SVR.

trained faster. Another reason to use feature selection is the over-fitting problem. Over-fitting problems or the *curse of dimensionality* occurs when the dimensionality of the feature space increases. In this case, the number of configurations can grow exponentially, and thus the number of configurations covered by an observation decrease. To avoid the *curse of dimensionality*, we select only useful features. There are various methodologies to subset the feature space and improve model performances, for example:

- **Filter Method:**

  In the Filter method, the selection of features is based on their scores in various statistical metrics. Various techniques, such as mutual information (MI), Pearson correlation, and principal component analysis (PCA), were selected for applying the filter method. Fig 4.5 show the process of using the filter method.

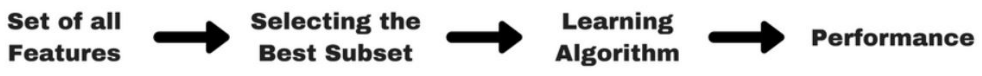


Figure 4.5: Overview of the filter method, adopted from [Kaushik n.d.].

- **Wrapper Method:**

  As we see in Fig 4.6, wrapper methods can always select the best subset of features, but because these methods must train the model, they are computationally expensive. It follows a greedy search approach by evaluating all the possible combinations of features against the performance measure.

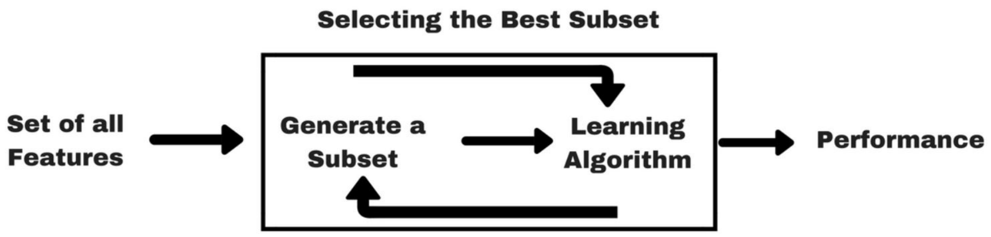


Figure 4.6: Overview of the wrapper method, adopted from [Kaushik n.d.].

  The most common examples of wrapper methods are recursive feature elimination and forward feature selection. Due to the lack of access to a high-performance system, we did not use the wrapper methods for feature selection.

- **Embedded Method:**

  The embedded method is a combination of filter and wrapper methods. We use algorithms that have their built-in feature selection methods like tree-based models. The shapely additive explanation (SHAP) method was used as a feature selection technique from the embedded method. Fig 4.7 shows how the embedded method works.

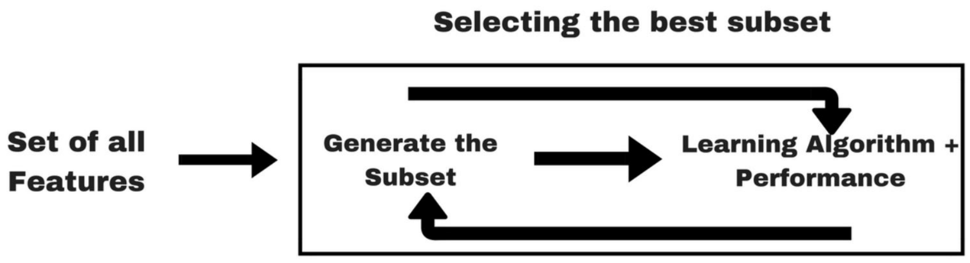


Figure 4.7: Overview of the embedded method, adopted from [Kaushik n.d.].

Mutual Information:

Mutual information (MI) estimates mutual information for a continuous target variable. MI between two random variables is a non-negative value, which measures the statistical dependency between the variables [Cover and Thomas n.d.]. It is equal to zero if and only if two random variables are independent, where higher values mean higher dependency [Kraskov, Stögbauer, Andrzejak et al. 2003].

If we consider X as a feature matrix and Y as the target vector, the function $M(X,Y)$ returns the mutual information of each feature with the target variable. The function relies on non-parametric methods based on entropy estimation from k-nearest neighbors' distances. It means that data is efficient (with k=1 we resolve structures down to the smallest possible scales), adaptive (the resolution is higher where data are more numerous), and have minimal bias [Kraskov, Stögbauer and Grassberger 2004; BC 2014].

The selected number of features are 5, 20, 50, 100. The number of neighbors to use for MI estimation for continuous variables is three. The higher number of neighbors reduces the variance of the estimation but could introduce a bias.

Table 4.2: Result of mutual information on 1-12 lead times of station 11007.

Number of features	Accuracy(%)	MSE	Bias
5 features	24.52	4.84	-1.06
20 features	24.34	4.86	-1.07
50 features	**28.22**	**4.56**	**-1.02**
100 features	28.22	4.61	-1.12
All features (132)	23.26	4.92	-1.26

As we can see from Table 4.2, selecting 50 features with high MI has better results compared to others.

Pearson Correlation:

Hereabouts, we use a correlation matrix based on the Pearson correlation. The correlation coefficient has a range of values between -1 to 1. A value closer to 0 implies a weaker correlation (exact 0 implying no correlation), a value closer to 1 implies a stronger positive correlation, and a value closer to -1 implies a stronger negative correlation. We were selecting features with different correlation thresholds. We only selected n top features with correlate above 0.2, 0.5, and 0.8 with the target variable, and then we compared the results.

Table 4.3: Result of the Pearson correlation on 1-12 lead times of station 11007.

Number of features	Accuracy(%)	MSE	Bias
>20	16.01	5.39	-1.06
>50	16.35	5.37	-1.08
>80	**26.23**	**4.73**	**-1.10**
All features (132)	23.26	4.92	-1.26

Table 4.3 shows that the remaining features after dropping the features which had more than 80% correlation between each other, performs better.

Principal Component Analysis:

Another feature selection method that we used is PCA. PCA is a linear dimensionality reduction method using singular value decomposition of the data and projects it to a lower-dimensional space [Scikit-Learn n.d.]. PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the highest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second-highest variance on the second coordinate, and so on [I.T 2002].

The principal component transformation associated with the singular value decomposition (SVD) of X can be written as

$$X = P\Delta Q^T \tag{4.7}$$

Here Δ is an n-by-p diagonal matrix of singular values of X; P is an n-by-n matrix, the columns of which are orthogonal unit vectors of length n called the left singular vectors of X; and Q is a p-by-p whose columns are orthogonal unit vectors of length p and called the right singular vectors of X [Abdi and Williams 2010].

We employed PCA with 20, 50, 80, and 100 components. From Table 4.4, we can see if we keep 50 components from the data-set, the linear kernel performs better.

Shapely Additive Explanations:

Shapley additive explanations (SHAP) is a game-theoretic approach to explain the output

Table 4.4: Result of PCA on 1-12 lead times of station 11007.

Number of Features	Accuracy(%)	MSE	Bias
20 components	16.66	5.35	-1.15
50 components	**27.81**	**4.63**	**-1.07**
80 components	25.57	4.78	-1.18
100 components	26.03	4.75	-1.16
All features (132)	23.26	4.92	-1.26

of any machine learning model, and it is a subset of the embedded method. SHAP values are defined as a unified measure of feature importance.

Let f be the original prediction model to be explained and g the explanation model. Here, we focus on local methods designed to explain a prediction $f(x)$ based on a single input x, as proposed in [Ribeiro, Singh and Guestrin 2016]. Explanation models often use simplified inputs x' that map to the original inputs through a mapping function $x = h_x(x')$. Local methods try to ensure $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$.

Definition 1: Additive feature attribution methods

Have an explanation model that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{4.8}$$

where $z' \in \{0,1\}^M$ is the simplified input, M is the number of simplified input, and $\phi_i \in R$ is the feature attribution for a feature i, the Shapley values.

Methods with explanation models according to Definition 1, attribute an effect ϕ_i to each feature, and summing the effects of all feature attributions approximates the output $f(x)$ of the original model. SHAP is a model that follows the Definition 1.

Shapley regression values are feature importances for linear models in the presence of multicollinearity. This method requires retraining the model on all feature subsets $S \subseteq F$, where F is the set of all features. It assigns an importance value to each feature that represents the effect on the model prediction of including that feature. To compute this effect, a machine learning model $f_{S\cup\{i\}}$ is trained with that feature present, and another model f_S is trained with the feature withheld. Predictions from the two models then compared to the current input $f_{S\cup\{i\}}(x_{S\cup\{i\}})$-$f_S(x_S)$, where x_S represents the values of the input features in the set S. Since the effect of withholding a feature depends on other features in the model, the other differences are computed for all possible subsets $S \subseteq F \setminus \{i\}$. The Shapley values are then computed and used as feature attributions [Lundberg and Lee 2017]. They are a weighted average of all possible differences:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] \tag{4.9}$$

For Shapley regression values, h_x maps 1 or 0 to the original input space, where one indicates the input is included in the model, and 0 indicates exclusion from the model. If we let $\phi_0 = f_\phi(\phi)$, then the Shapley regression values match Equation 4.8 and are hence an additive feature attribution method. The output of Equation 4.8, is the Shapley value of the features which are included in the model [Molnar n.d.]. Historical accuracy, missingness, and consistency will be satisfied if and only if ϕ_i are Shapley values defined in Equation 4.9.

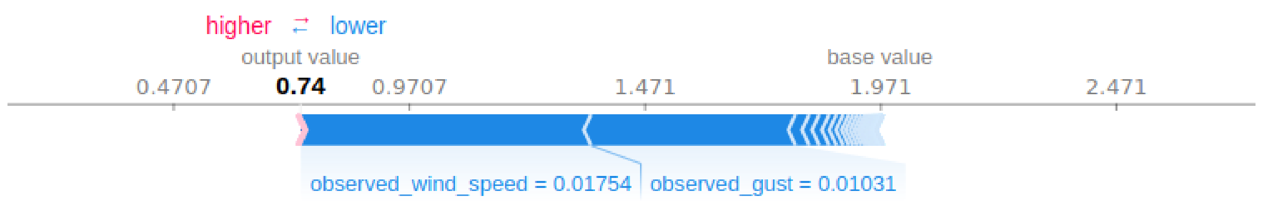
Table 4.5: Average results of SHAP for 1-12 lead times in station 11007.

Number of Features	Accuracy(%)	MSE	Bias
5 features	28.62	4.58	-0.99
20 features	**31.41**	**4.40**	**-1.04**
50 features	29.21	4.54	-1.09
80 features	26.67	4.71	-1.16
100 features	24.93	4.82	-1.21
All features (132)	23.26	4.92	-1.26

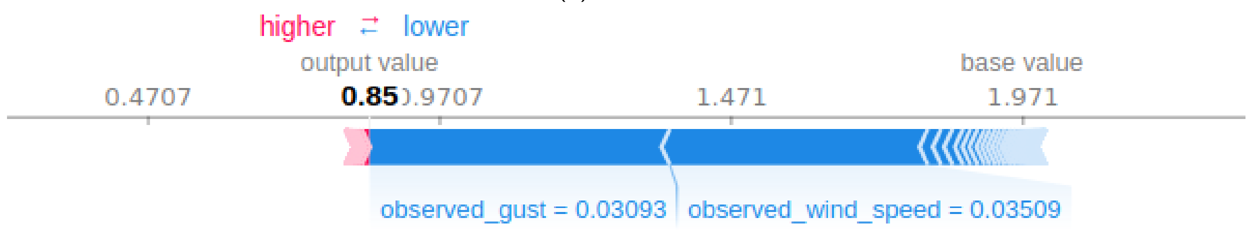
After applying the SHAP algorithm, 5, 20, 50, 80, and 100 important features were selected and compared. After selecting a different number of important features for modeling the linear kernel from SVR, Table 4.5 shows that keeping 20 most important features performs better than the others.

In the above results, it is evident that an optimized feature selection improves the predictions. To select the 20 most important features, we have to look at the importance of each lead time individually. Selecting important features in predicting wind speed for the next 1, 2, 6, and 12 hours will be discussed below.

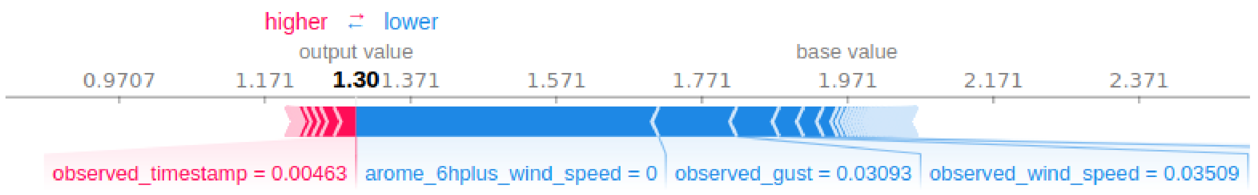
The wind speed forecasting in the next 1, 2, 6, and 12 hours shown in Fig 4.8 is driven by a set of features. The SHAP summary plot shows how far features contribute to pushing the output value from the original value (the average of the model output over the training-set) to the predicted value. Features pushing the wind speed prediction higher are shown in red, those pushing the wind speed prediction lower are in blue. The magnitude of impact sorts each feature, and the features with the most significant impact are labeled. We can visualize features importance



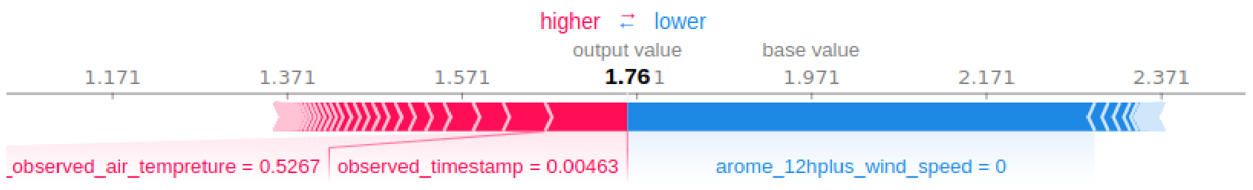
(a) + 1-hour



(b) + 2-hours



(c) + 6-hours



(d) + 12-hours

Figure 4.8: SHAP summary plots for 1, 2, 6 and 12 lead times.

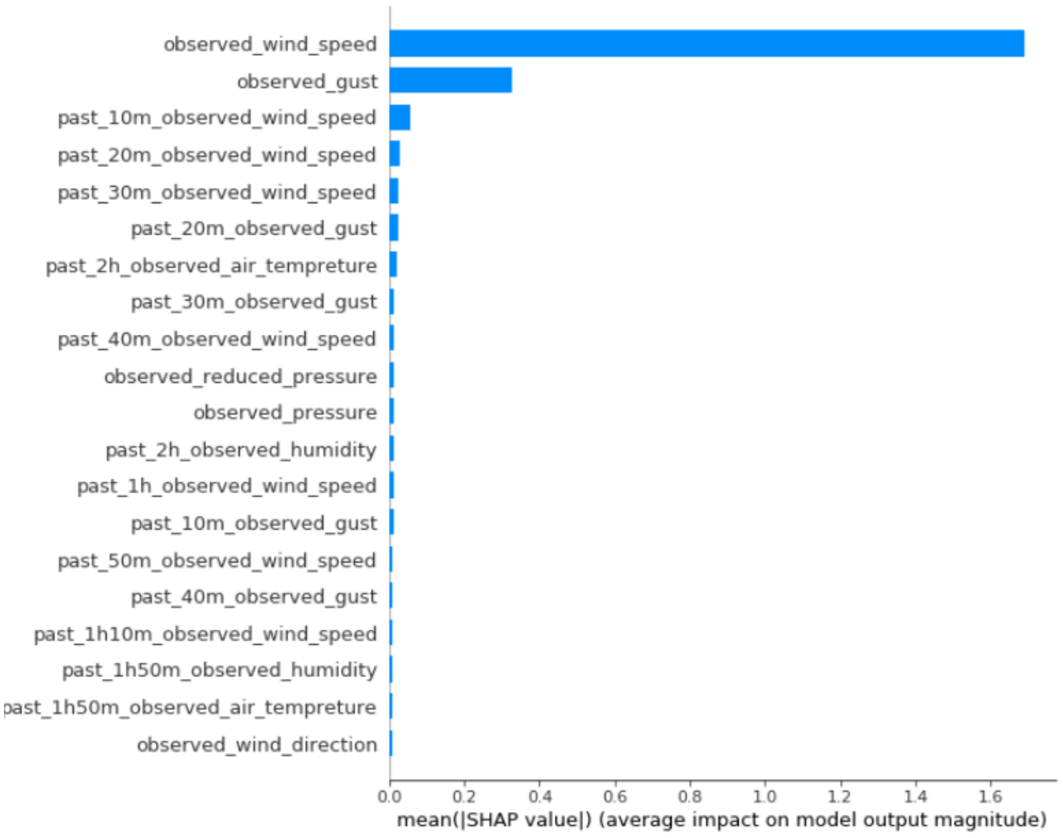
such as Shapley values as "forces". In the plot, each Shapley value is an arrow that pushes the prediction to increase (positive value) or decrease (negative value). These forces balance each other out at the actual prediction of the data instance [Molnar n.d.].

Fig 4.8a shows *observed_wind_speed* and *observed_gust* (See the description in Table 3.1) have the highest impact in our prediction, but both of them push the prediction to the lower value. Hence, if our base value wind speed is 1.971, both *observed_wind_speed* and *observed_gust* try to lower the wind speed prediction to 0.74. Nevertheless, this is the best we can get for a 1-hour ahead prediction. The same can be observed in Fig 4.8b, which is for a 2-hours ahead prediction. However, results in Fig 4.8c and Fig 4.8d indicate that *observed_timestamp* feature tries to push the output value to the base value, which explains the effect of specific times in our prediction. In addition *arome_6hplus_wind_speed* (lag feature for AROME data, predicted wind speed for the next 6 hours) and *arome_12hplus_wind_speed* (lag feature for AROME data, predicted wind speed for the next 12 hours) have more importance compared to *observed_wind_speed* in 1-hour and 2-hours forecasting. It seems when lead times increase, the impact of AROME data is more than Observation data. Also, we see in Fig 4.8d, for a 12-hours ahead prediction, the output value is 1.76. The wind speed prediction increasing effects such as *observed_air_temperature* is offset by decreasing effects such as *arome_12hplus_wind_speed*.

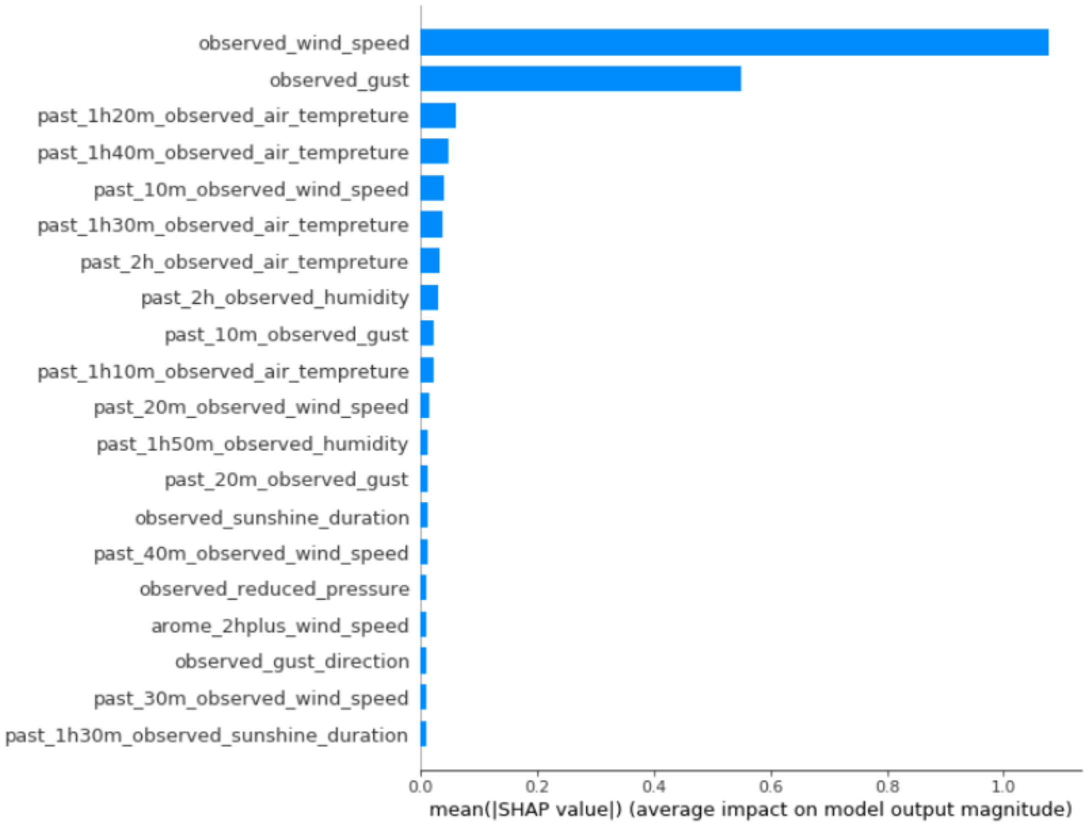
The 20 most important features for 1 and 2 hours lead time are shown in Fig 4.9 and for 6 and 12 hours lead time in Fig 4.10. The idea behind the SHAP feature importance plot is simple: features with large absolute Shapley values are important.

In this plot, we see the sorting features by decreasing importance. The x-axis shows the mean absolute Shapley values. In Fig 4.9a, for 1-hour ahead wind speed prediction, *observed_wind_speed* is the most important feature, changing the prediction on average by 160 percentage points (1.6 on the x-axis). On the other hand, *arome_12hplus_wind_speed* in Fig 4.10b has the effect of almost 1.04 for 12-hours ahead prediction. From the importance plots, we can see it is better to consider the effect of *arome_XXhplus_wind_speed* feature in the wind speed prediction for higher lead times.

As we can see from Fig 4.9 and Fig 4.10, each lead time shows different optimized features, and all 28 stations may have the same behavior. However, it is too time-consuming to select the best features for 48 lead times in 28 stations for the final run. The main idea followed in this thesis was to choose a unique list of optimized features for each lead time to decrease the computational running time and also for better interpretability. By preliminary experiments, Table 4.6 shows the defined list of the 20 most important features for the target stations.

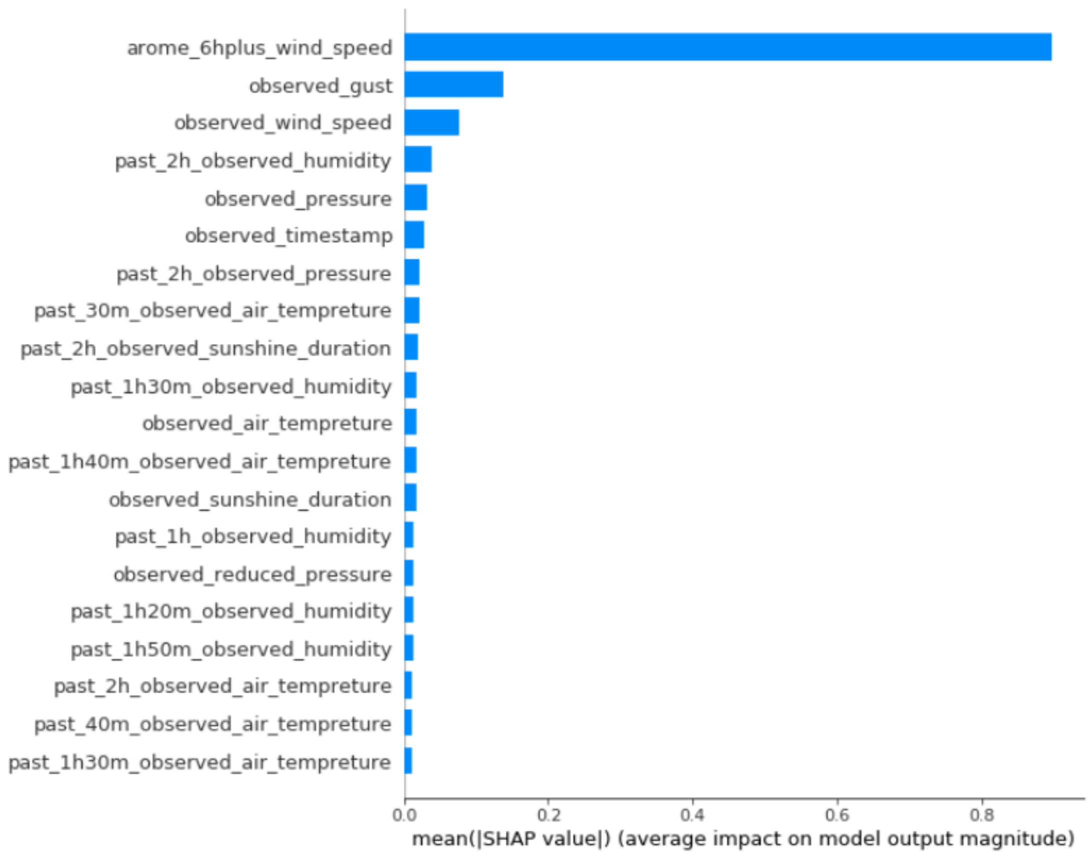


(a) + 1-hour

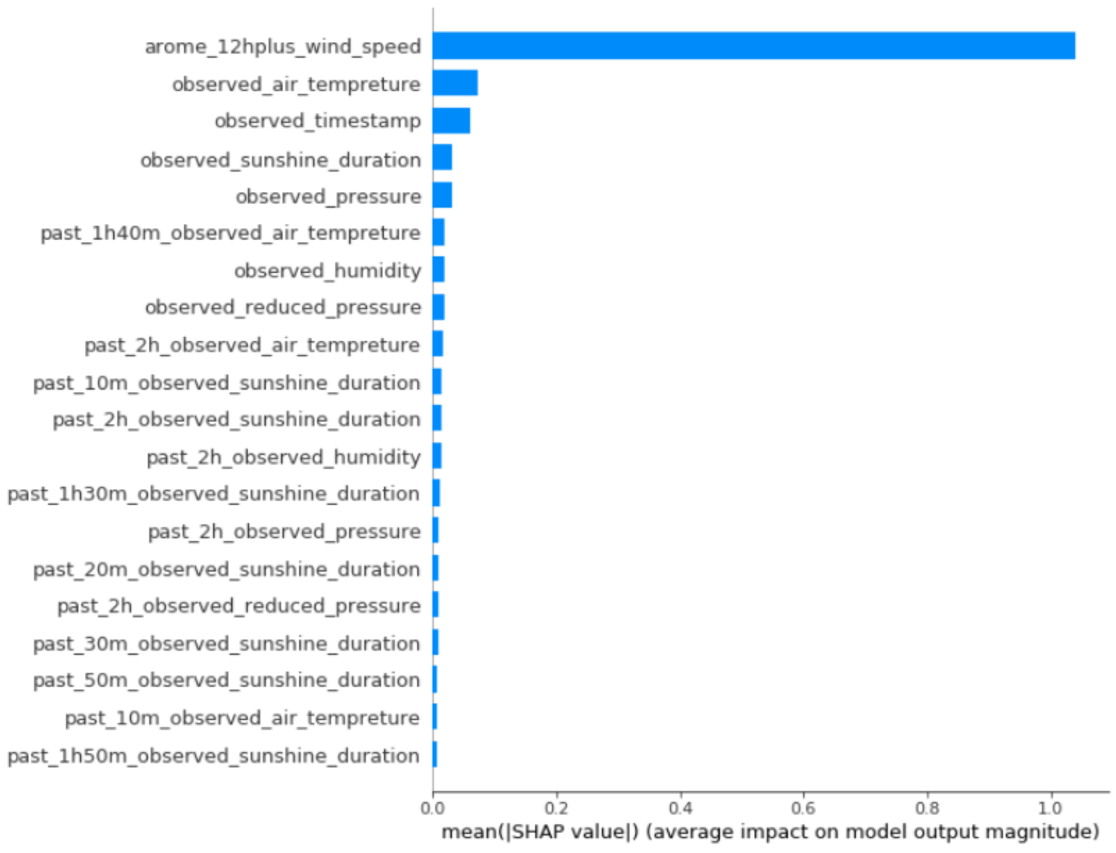


(b) + 2-hours

Figure 4.9: SHAP feature importance plots for 1 and 2 hours lead times



(a) + 6-hours



(b) + 12-hours

Figure 4.10: SHAP feature importance plots for 6 and 12 hours lead times

Table 4.6: List of 20 most important features. * All wind parameters are observed at 10 meters above the surface, and all temperature parameters are observed at 2 meters above the surface.

	Features	Description
1	observed_timestamp	observation day + observation minutes
2	observed_wind_speed	observed wind speed*
3	observed_gust	observed gust
4	past_2h_observed_air_temperature	past 2 hours observed air temperature*
5	observed_reduced_pressure	observed reduced pressure
6	observed_pressure	observed pressure
7	past_2h_observed_humidity	past 2 hours observed humidity
8	observed_air_temperature	observed air temperature*
9	observed_sunshine_duration	observed sunshine duration
10	past_2h_observed_pressure	past 2 hours observed pressure
11	past_2h_observed_sunshine_duration	past 2 hours observed sunshine duration
12	observed_humidity	observed humidity
13	past_1h50m_observed_humidity	past 1 hour and 50 minutes observed humidity
14	observed_wind_direction	observed wind direction*
15	past_10m_observed_air_temperature	past 10 minutes observed air temperature*
16	past_2h_observed_wind_direction	past 2 hours observed wind direction*
17	past_2h_observed_reduced_pressure	past 2 hours observed reduced pressure
18	past_10m_observed_gust	past 10 minutes observed gust
19	past_1h50m_observed_air_temperature	past 1 hour and 50 minutes observed air temperature*
20	past_1h40m_observed_air_temperature	past 1 hour and 40 minutes observed air temperature*

By 20 features and *arome_XXhplus_wind_speed* being added in each lead time, we obtained a total of 21 features for the hyper-parameter tuning and our final SVR model. The choice of features (feature selection) may have an impact on which hyper-parameters for our algorithm are optimal, and which hyper-parameters we select for our algorithm may have an impact on which choice of features would be optimal. The best solution is to make the feature selection simultaneously with hyper-parameter tuning to achieve high performance and reduce complexity.

1. Feature selection with a decently configured machine learning algorithm significantly has a high impact on our performance. Finding the best kernel in section 4.5, improves having a more configured model to find the best selection of features.
2. Tuning hyper-parameters with the features selected in the step above. A list of good feature set increases the chance of having more optimized hyper-meters.

Feature selection is more time consuming than hyper-parameter tuning. With an optimized feature selection, it is sufficient to apply a coarse grid for a hyper-parameter search.

To evaluate the SVR performance, we compared the forecasting results with two baseline models, the Persistence model, and the AROME model. Both models have evolving wind speed values, which is regressed on its own lagged prior values. The purpose of creating new features (lagged wind speed) is to make the model fit the data as well as possible.

4.7 Baseline Models

4.7.1 Persistence Model

One of the simplest methods for predicting time series is the so-called Persistence model. Persistence implies that future values of the time series are calculated on the assumption that conditions remain unchanged between current observation time and the later time [F.M.Coimbra and T.C.Pedro 2013]. A straightforward implementation of the Persistence model for multi-step ahead prediction is:

$$\hat{y}(t_i) = y(t_0) \tag{4.10}$$

for i = 1, 2, ,3 , ... , 48 and t_0 as initial time.

Multi-step ahead prediction by means is to predict observed wind speed for 48-hours ahead (in this case, 43-hours). The Persistence forecast may be the best possible model for nowcasting. An example of creating the Persistence model is provided in Table 4.7. The Persistence model can be easily generated. We assume the wind speed to remain constant for all forecasting hours of the run.

Table 4.7: Persistence model

Persistence model time	forecast hour	Observed time
2015-01-01 03:00:00	+1	2015-01-01 03:00:00
2015-01-01 04:00:00	+1	2015-01-01 04:00:00
2015-01-01 03:00:00	+2	2015-01-01 03:00:00
2015-01-01 04:00:00	+2	2015-01-01 04:00:00
2015-01-01 03:00:00	+43	2015-01-01 03:00:00
2015-01-01 04:00:00	+43	2015-01-01 04:00:00

4.7.2 AROME Model

Table 4.8: AROME model

AROME model time	forecast hour	AROME computational time
2015-01-01 03:00:00	+1	2015-01-01 00:00:00 (+4h)
2015-01-01 04:00:00	+1	2015-01-01 00:00:00 (+5h)
2015-01-01 05:00:00	+1	2015-01-01 00:00:00 (+6h)
2015-01-01 06:00:00	+1	2015-01-01 03:00:00 (+4h)
2015-01-01 03:00:00	+2	2015-01-01 00:00:00 (+5h)
2015-01-01 04:00:00	+2	2015-01-01 00:00:00 (+6h)
2015-01-01 05:00:00	+2	2015-01-01 00:00:00 (+7h)
2015-01-01 06:00:00	+2	2015-01-01 03:00:00 (+5h)
2015-01-01 03:00:00	+43	2015-01-01 00:00:00 (+46h)
2015-01-01 04:00:00	+43	2015-01-01 00:00:00 (+47h)
2015-01-01 05:00:00	+43	2015-01-01 00:00:00 (+48h)
2015-01-01 06:00:00	+43	2015-01-01 03:00:00 (+46h)
2015-01-01 03:00:00	+44	2015-01-01 00:00:00 (+47h)
2015-01-01 04:00:00	+44	2015-01-01 00:00:00 (+48h)
2015-01-01 05:00:00	+44	2015-01-01 00:00:00 (+49h)

AROME model is also fitted to time series data for a better understanding of the data or forecasting future points in the series. For the AROME model, shifting the features to the future is necessary, as we know from section 3.1.2. Table 4.8 shows a small example of AROME model.

The AROME computational times are at 00, 03, 06, 09, 12,15, 18, 21 UTC (eight times per day). For the initial time + 1-hour at the initial time *2015-01-01 03:00:00*, we retrieved the most recent AROME model output, which is *2015-01-01 00:00:00* with the lead time +4. Choosing this model output originates in the delayed model availability of NWP, which is assumed to be four hours for the AROME model. Also, for *2015-01-01 04:00:00* and *2015-01-01 05:00:00*, we should take computed wind speed in AROME data at *2015-01-01 00:00:00* for 5-hours and 6-hours ahead respectively, until the next available AROME computational time, which is *2015-01-01 03:00:00*. The same applies for 2, 3, up to 42 forecast hours. Because our last AROME forecasting hour is 48, we have the wind speed shifted feature for AROME data up to 43 hours. Thus, the last possible forecasting hour in the AROME model is 43-hours, and we will predict the wind speed in the Persistence and the final SVR model until 43-hours.

Chapter 5

Results

Our objective was to evaluate the forecast skill of different setups of a SVR model for 1-hour to 43-hours ahead forecasts of wind speed based on the data-sets from 28 observation sites and 28 NWP AROME data for each station in Austria. The data was taken from the period 2015-01-01 until 2017-12-31 with a resolution of 10 minutes. We divided the data into a section for the SVR training and a test part to evaluate the generalization ability of the prediction models, which was January 2017. The time period for the test part was entirely outside the time period for training. The evaluation results are based on test data.

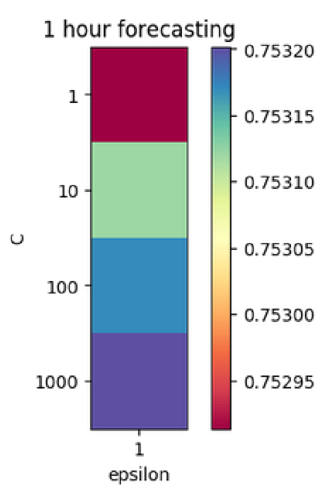
To test the performance of the models, different metrics proposed in section 4.4 were used. Therefore, we needed to evaluate the performance based on multiple metrics, like accuracy, MSE, RMSE, bias, and Spearman's correlation.

A linear SVR, which is equivalent to an SVR without kernel transformation, was used considering the SVM regression. For all of the linear SVR models, it was necessary to drive the value of the regularization parameters C and ϵ. In the next section, we discuss the final evaluation scores and the results.

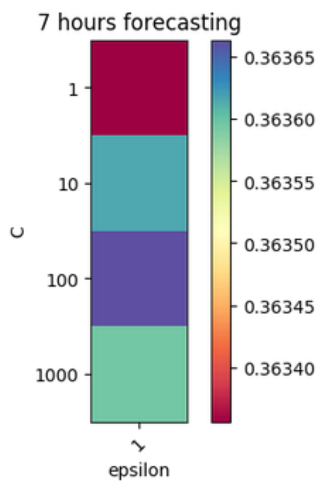
5.1 SVR Configuration

The training-data was used to optimize the parameters C and ϵ described in Section 4.5. Several trials were used to find reasonably good values of these parameters for the wind speed prediction for 28 different stations in Austria. As expected, the results showed that for each lead time, the SVR model performed better with different values of C and one value for the ϵ parameter.

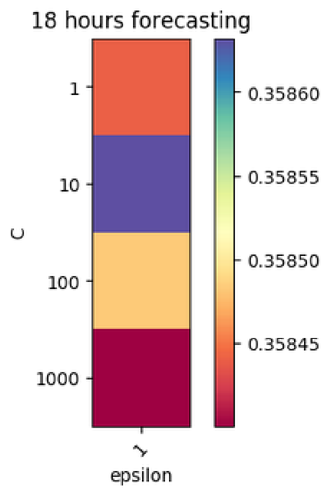
Fig 5.1 shows the scores of *Grid Search* for three different lead times in station 11007. The



(a) + 1-hour



(b) + 7-hours

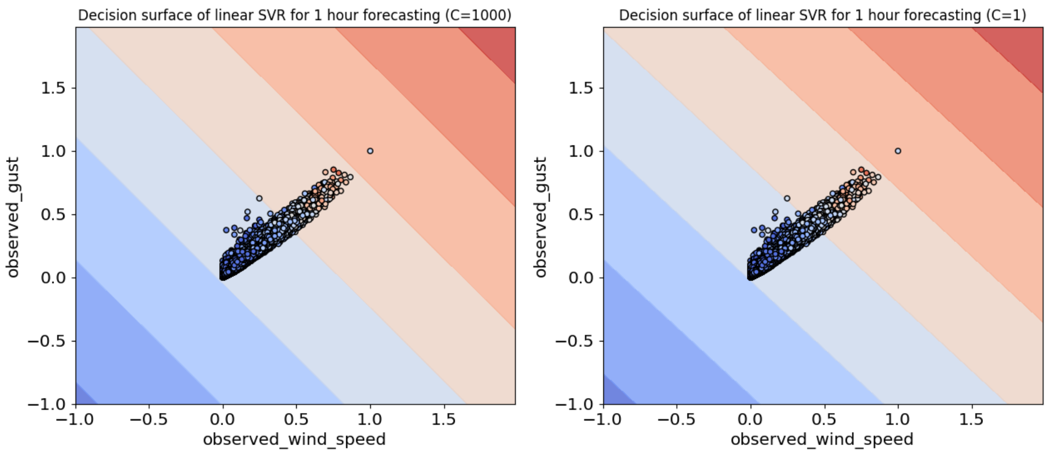


(c) + 18-hours

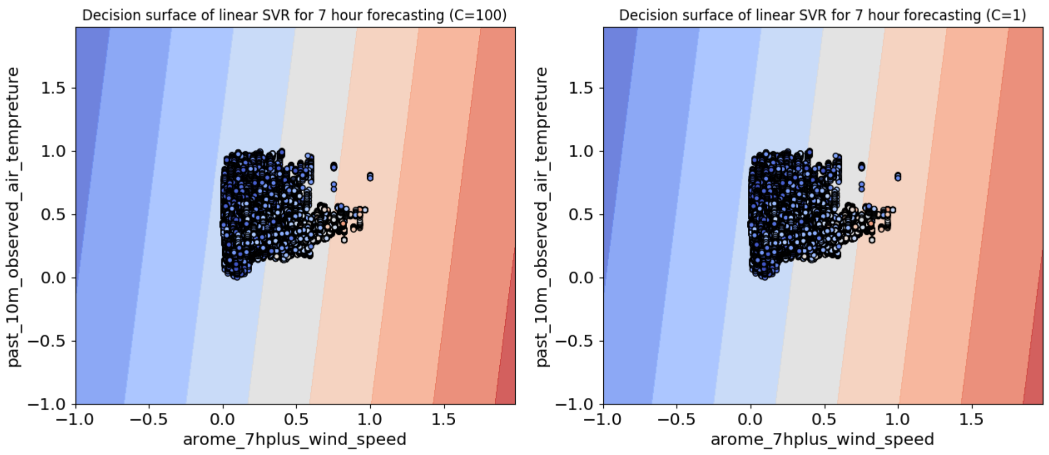
Figure 5.1: R^2 score for three lead times in station 11007.

purple color shows the highest score for each lead time. Fig 5.1a shows when $C = 1000$ and $\epsilon = 1$, we get the highest $R^2 \simeq 0.7533$. And Fig 5.1b, 5.1c show the highest score of $R^2 \simeq 0.3624$ when $C = 100$, $\epsilon = 1$ and $R^2 \simeq 0.35863209$, when $C = 10$, $\epsilon = 1$ respectively.

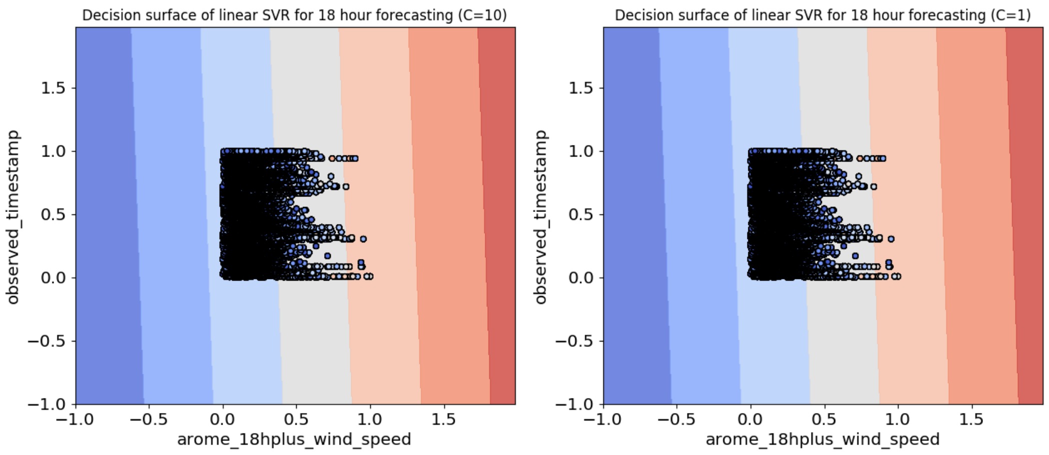
Results for other lead times showed for 1-3 hours forecasting $C = 1000$, $\epsilon = 1$; for 4-17 hours forecasting $C = 100$, $\epsilon = 1$ and for 18-43 hours forecasting $C = 10$, $\epsilon = 1$ have the highest scores.



(a) + 1-hour



(b) + 7-hours



(c) + 18-hours

Figure 5.2: Different value of C for three lead times in station 11007.

Fig 5.2 shows how SVR, with its hyper-planes, separates the testing points for the three previous lead times, 1-hour, 7-hours, and 18-hours, when the C parameter changes. For a better understanding, only two features for each lead time were considered, the highest two important features. A linear model has linear decision boundaries (intersecting hyper-planes). We have to consider that these plots only give an intuitive understanding of their respective expressive power, and we cannot generalize them to more realistic high-dimensional problems. The difference of the regularization parameter C might be not significant, specially in Fig 5.2b and 5.2c, but in Fig 5.2a, we see a significant difference between two plots. The left plot shows the decision boundaries when $C = 1000$, and the right shows the decision boundaries when $C = 1$. We can see the slope of the decision boundaries in $C = 1000$ has slightly turned anticlockwise. All plots on the left side show the C parameter with its tuned value, and all the plots on the right side show the C parameter with its default value.

In SVR, the fitting time complexity is more than quadratic with the number of samples, and even using SVR in *Grid Search*, it gets extremely slow, and may lead to a very long execution time. Especially with the high values of C, the computational time increases significantly. Fig 5.3 shows that when the C value decreases, the fitting time also decreases.

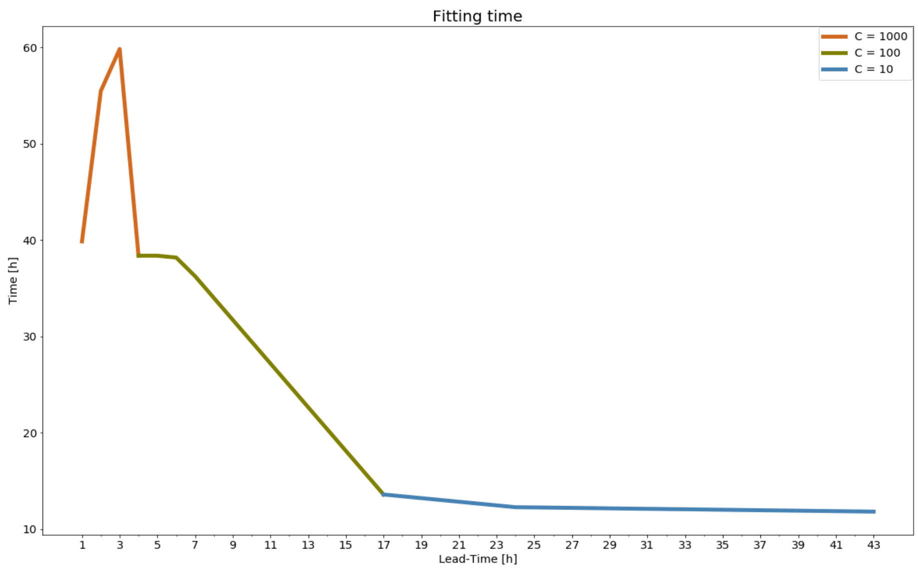


Figure 5.3: Fitting time for different parameter C for each lead time in station 11007.

Due to the high time complexity of the *Grid Search* algorithm, and the lack of access to high-performance systems, we used the selected parameters from station 11007 for other stations as well.

5.2 Feature Selection

Two different approaches were used for feature selection. The first approach is to use the 21 unique lists (see page 50) for each lead time in each station. The second approach is to employ the SHAP method (See Section 4.6) to select the 20 most important features for each lead time in each station. The second approach was more time-consuming because the SHAP algorithm needs to be applied 43 times to each of the 28 stations.

In Fig 3.4, the distribution of wind speed for four stations was presented (11034, 11044, 11108 and 11126). They represent different topological and climatological regions in Austria. The comparison of the models for stations 11034, 11044, 11108, and 11126 are shown in Fig 5.4, Fig 5.5, Fig 5.6 and Fig 5.7 respectively. The three models used are: SVR-Linear model with default parameters and all features, SVR-Linear (fs(unique) + tune) model with a selection of a different list of 21 features plus the tuned regularization parameter C and SVR-Linear(shap+tune) model which shows SVR with a linear kernel with the 20 most important features with tuned regularization parameter C for each lead time.

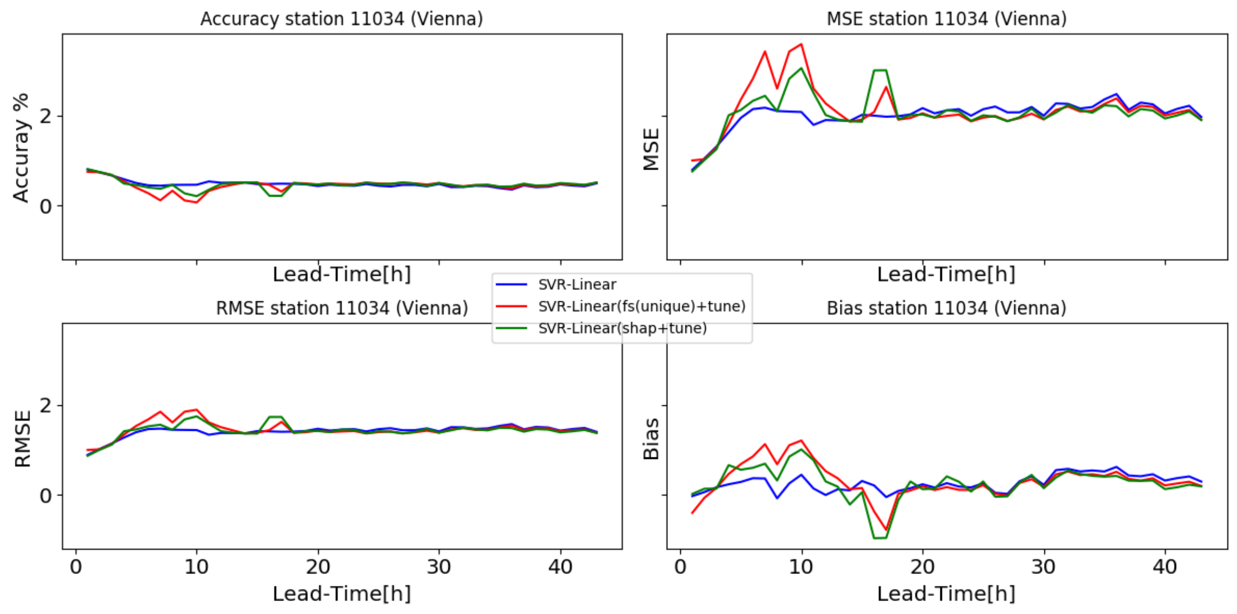


Figure 5.4: Comparison of three SVR models for station 11034.

The blue line shows the SVR-Linear model with default parameters, and using all the features (132 features); the red line is the SVR-Linear model with 21 selected features and tuned C parameter; and the green line is the SVR-Linear model with 20 most important features and tuned C parameter. Four different plots are shown in each figure: the accuracy plot, MSE plot, RMSE plot, and bias plot.

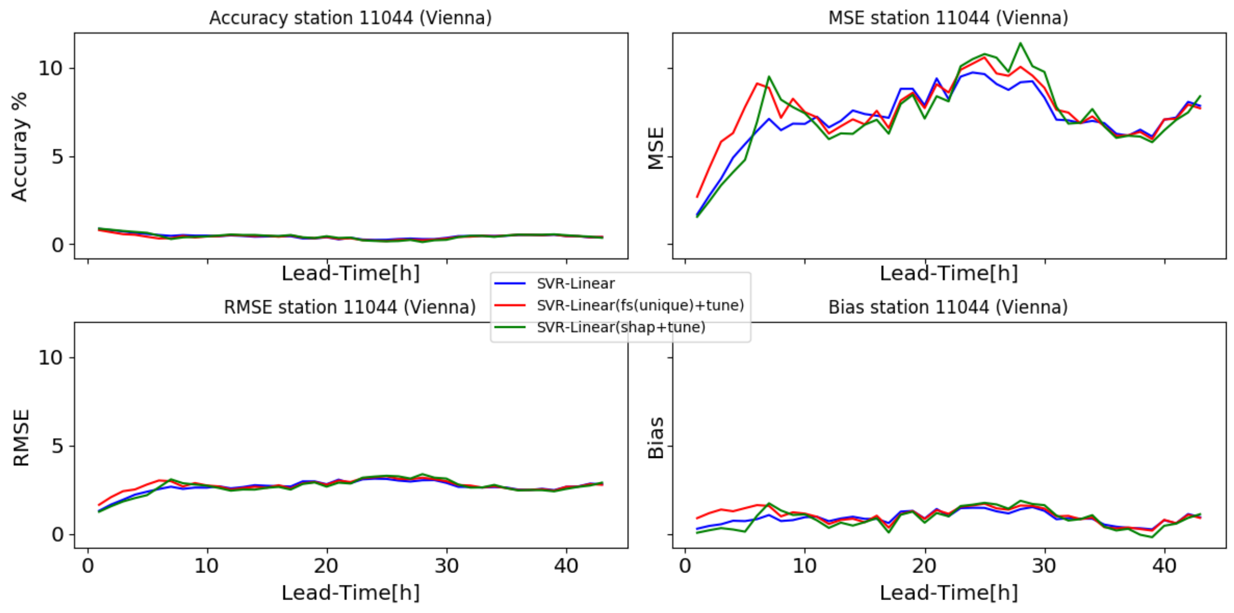


Figure 5.5: Comparison of three SVR models for station 11044.

Fig 5.4 and Fig 5.5 shows the evaluation metrics in the Vienna region. Unexpectedly, SVR-Linear (shap+tune) model does not a show better performance; however, close to the original model, SVR-Linear, which indicates that we can use fewer features but get the same results.

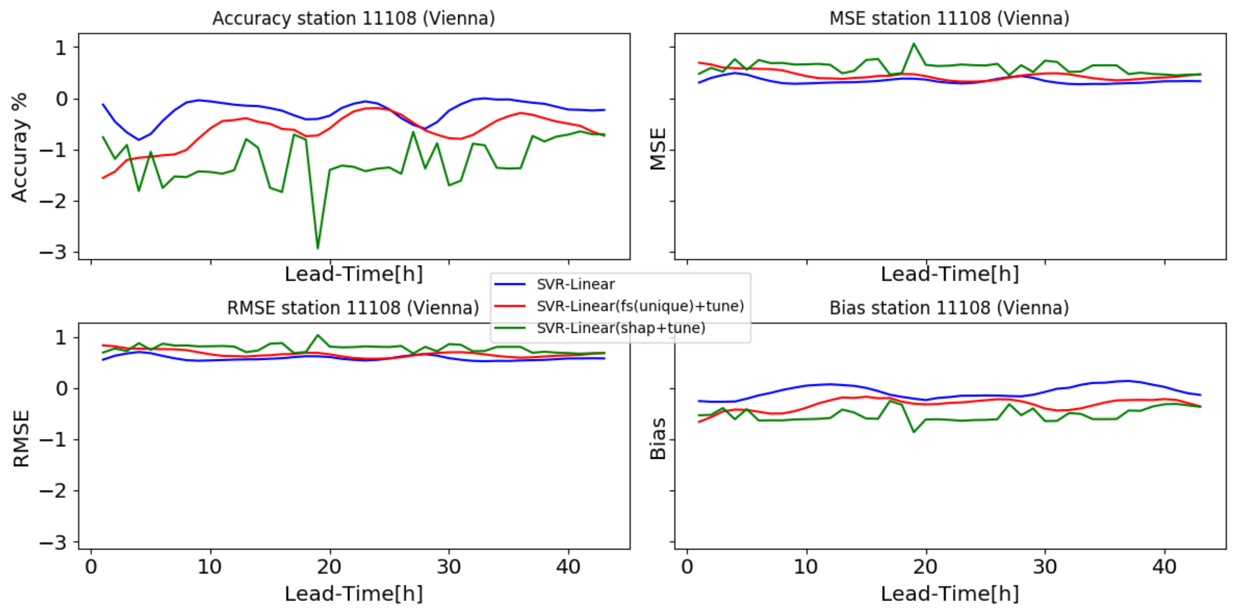


Figure 5.6: Comparison of three SVR models for station 11108.

Fig 5.6 shows station 11108 in Vorarlberg and Fig 5.7 shows station 11126 in Tirol. Fig 5.6 shows really poor results regarding both SVR-Linear (fs(unique)+tune) and SVR-Linear (shape+tune). Both feature selection and hyper-meter tuning did not improve the results in this station. Low bias (over-fitting problem) causes high error (MSE) and it is a possibility that SVR is not a suitable model for this station. Fig 5.7 shows performance issues with SVR model.

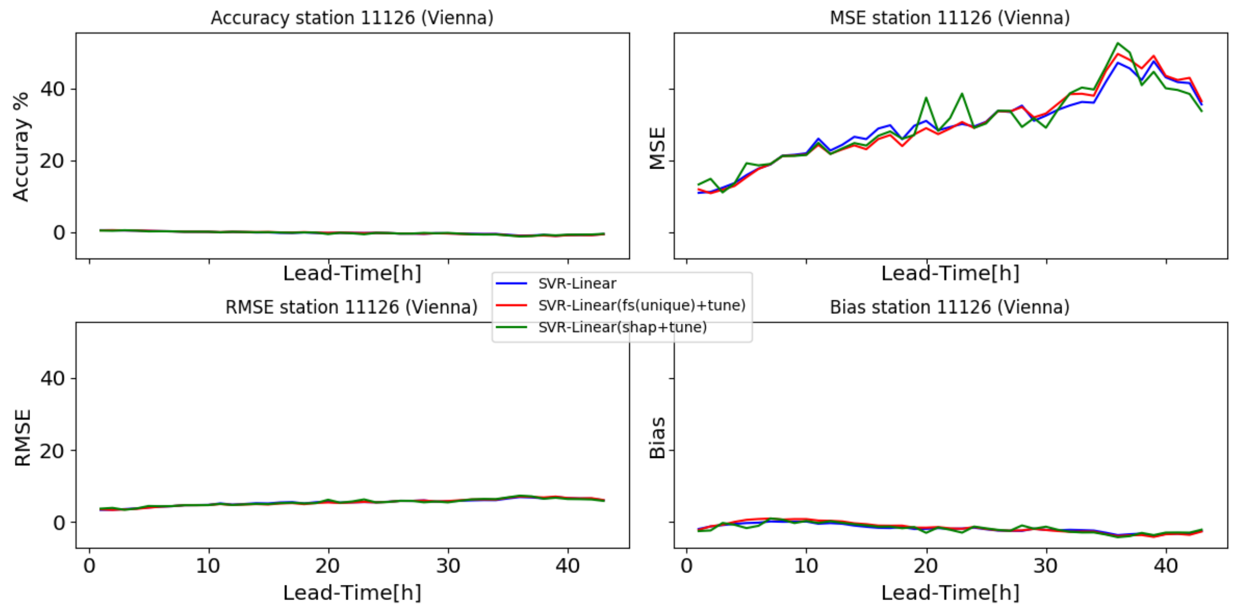


Figure 5.7: Comparison of three SVR models for station 11126.

After achieving the previous results for the four stations, we compared the three models in all stations to see the performance of these models before we took any further action. Fig 5.8 shows the accuracy, MSE, RMSE and bias for all stations.

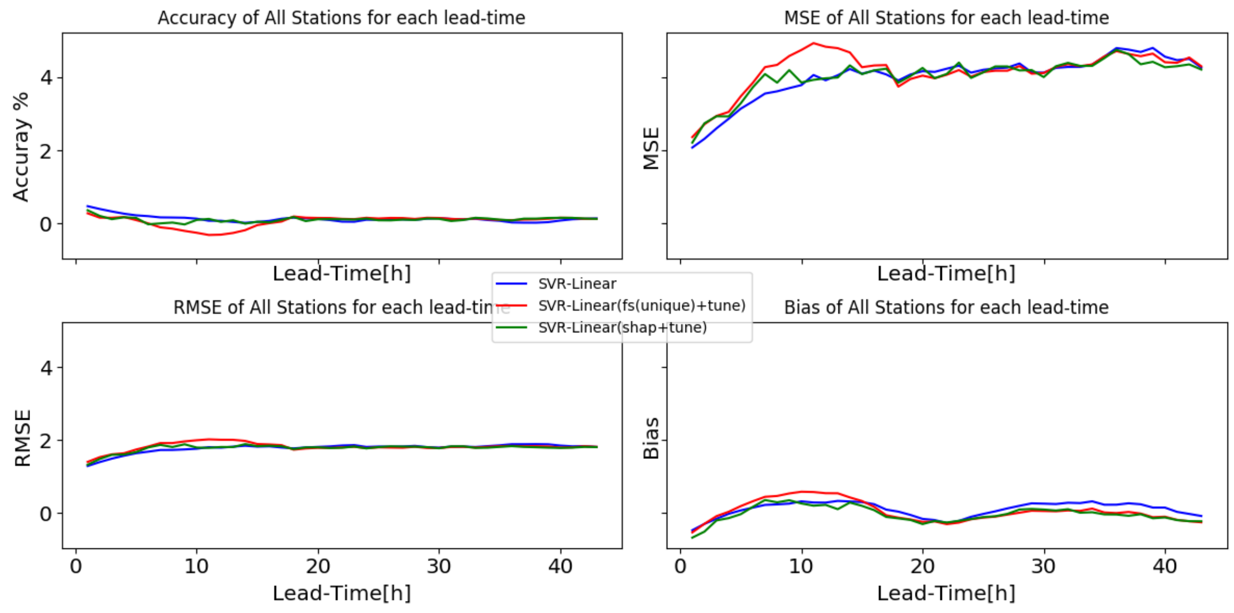


Figure 5.8: Comparison of the three SVR models for all lead times in all stations.

According to the bias shown in Fig 5.8, the difference of the three models is neglectable, but SVR-Linear and SVR-Linear (shap+tune) perform better than SVR-Linear (fs(unique)+tune), especially in almost up to 15 hours forecasting.

Due to high computational costs, we decided to use the SVR-Linear (shap+tune) model for optimization. Fig 5.9 shows the fitting time and prediction time for the two different models: SVR-Linear with almost 132 features and default parameters versus SVR-Linear (shap+tune) with 20 features and tuned parameters. The blue bar shows the fitting time, and the orange bar shows the prediction time. For 1-hour to 3-hours forecasting in SVR-Linear (shap+tune) model, the fitting time is higher than the SVR-Linear model. The overall results show that the fewer the features, the faster the computations and the better the forecast results.

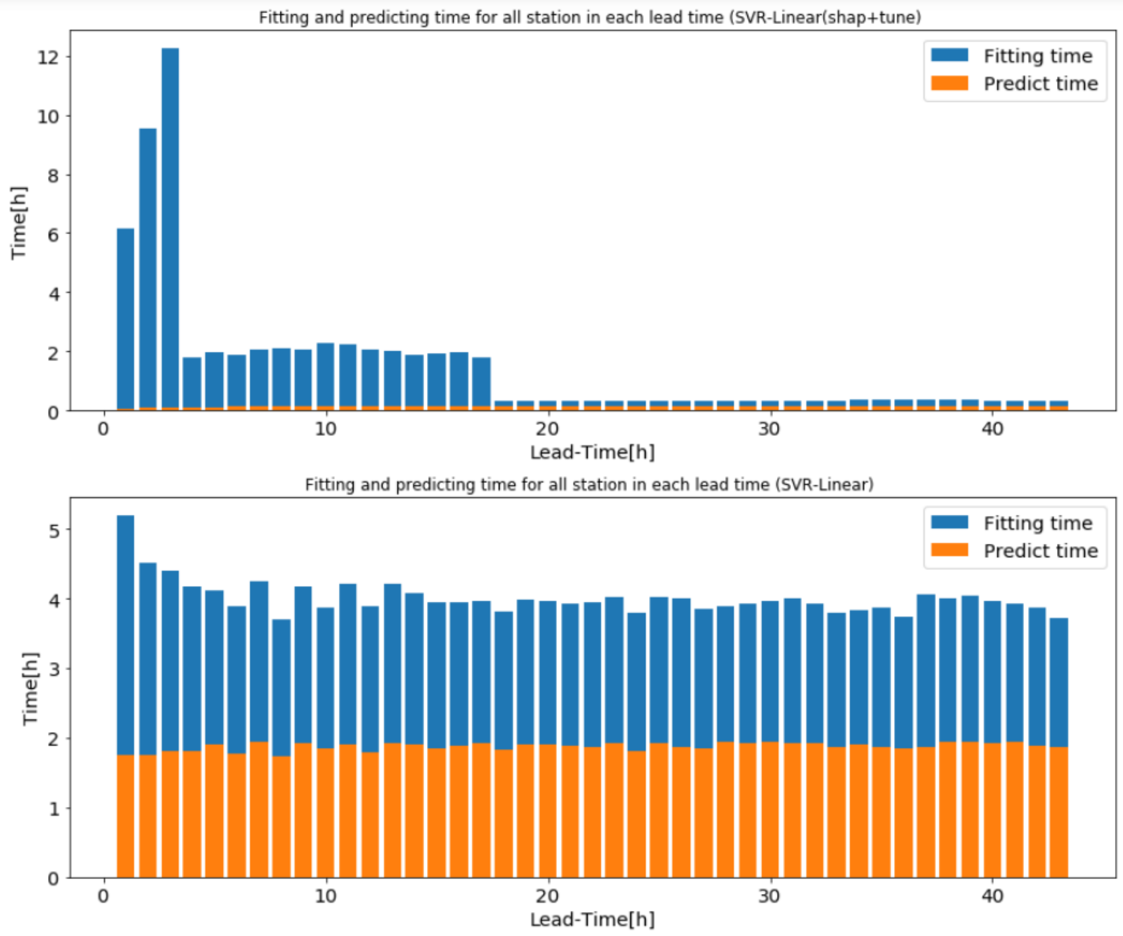


Figure 5.9: Fitting and predicting time, for two models, SVR-Linear(shap+tune) and SVR-Linear.

5.3 Wind Speed Forecast

Next, we will compare the results of the two basic models, Persistence, and AROME models with SVR-Linear (shap+tune). We first compare the results for four stations, 11034, 11044, 11108, and 11126, and then compare the results for all stations.

Fig 5.10, Fig 5.11, Fig 5.12, and Fig 5.13 show the performance of the three models: Persistence, AROME and SVR with the linear kernel using the 20 most important features and tuned regularization parameter C. We expected to see SVR outperform the AROME and Persistence models. All figures show four plots: MSE plot, RMSE plot, bias plot, and Spearman's correlation plot.

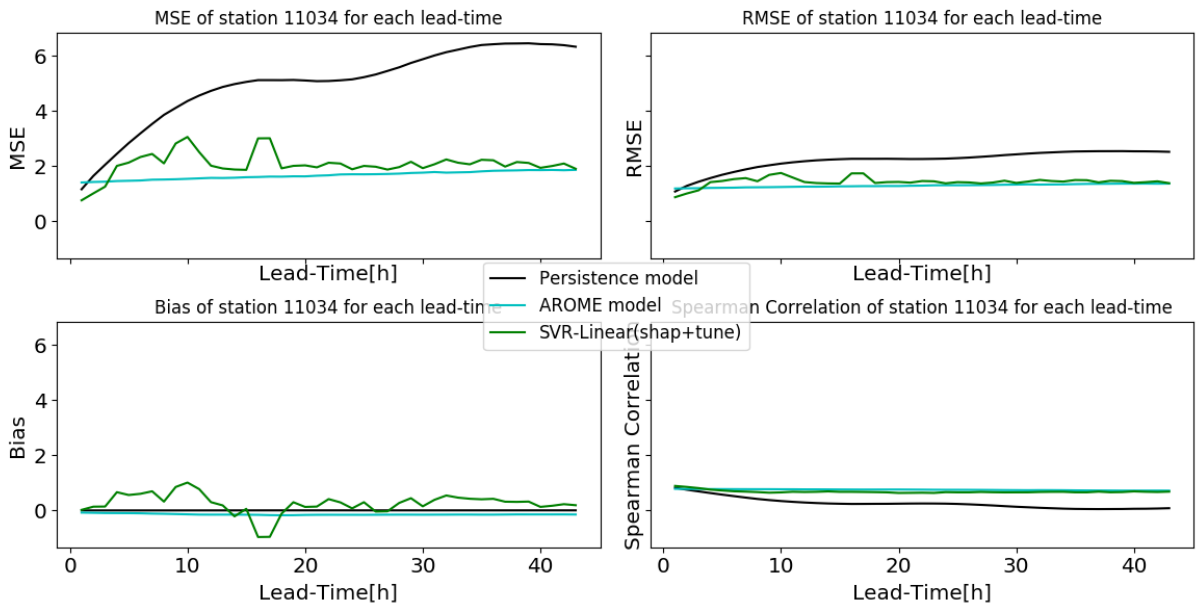


Figure 5.10: Comparison of three models for station 11034: SVR, Persistence and AROME models.

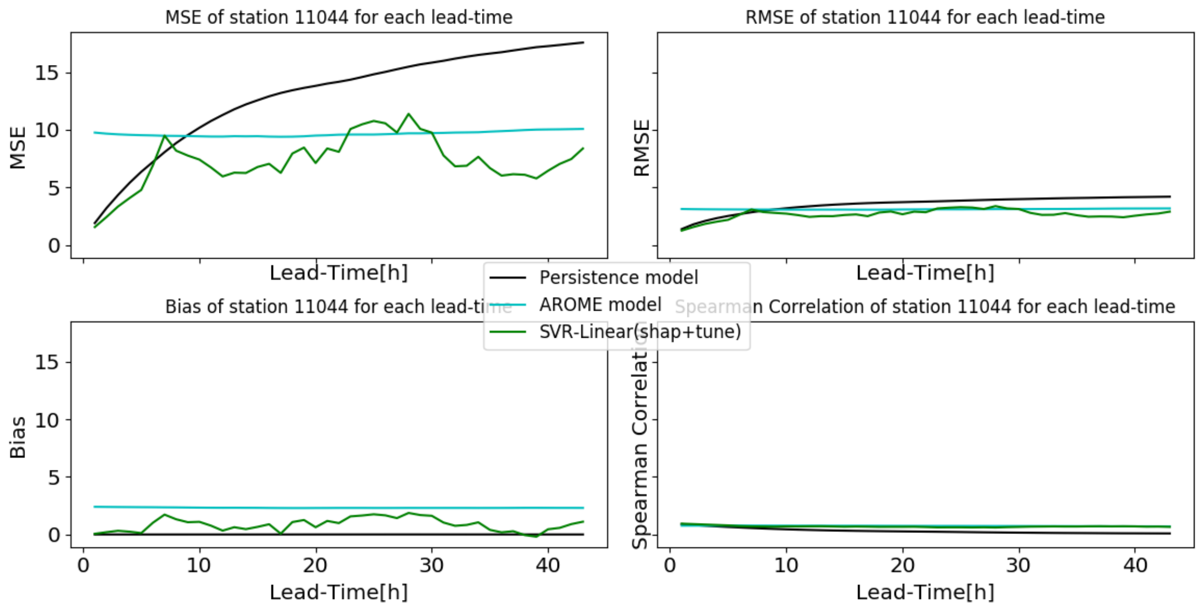


Figure 5.11: Comparison of three models for station 11044: SVR, Persistence and AROME models.

Although both stations 11034 and 11044 are in the Vienna region, station 11044 performed better with SVR than station 11034. In station 11034, we see more errors for SVR in early forecasting. The Persistence model shows errors for each lead time in both stations. As we see in the Spearman's correlation plot, the Persistence model tends to approach zero; when the lead time increases, there is not a high correlation between the predicted wind speed and the actual wind speed. The same result happened in station 11044.

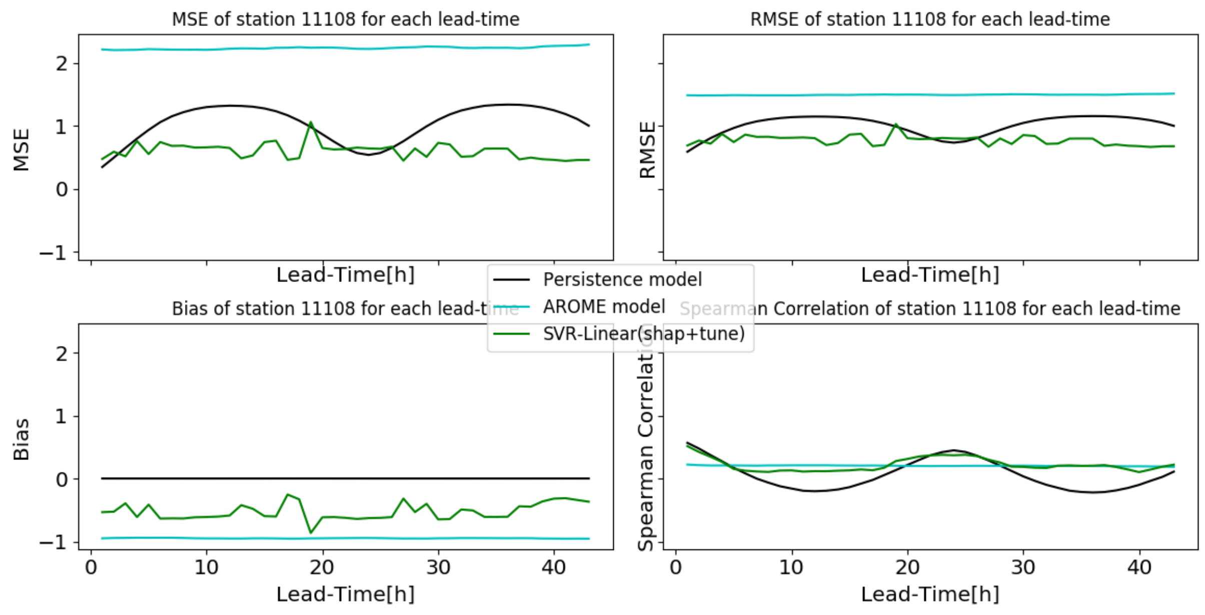


Figure 5.12: Comparison of three models for station 11108: SVR, Persistence and AROME models.

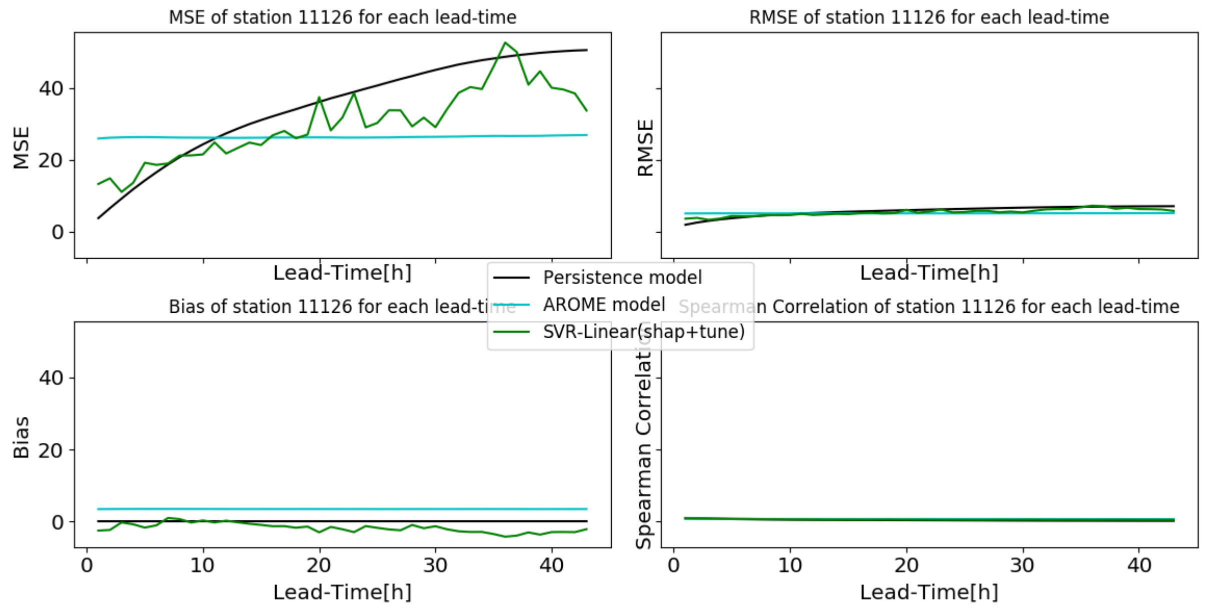


Figure 5.13: Comparison of three models for station 11126: SVR, Persistence and AROME models.

Station 11108 in Fig 5.12 shows the limited performance of the AROME model. As we saw in Fig 5.6, station 11108 also showed unsatisfactory results for different SVR models, but compare to the AROME and Persistence model, the SVR model performed better. The Persistence model shows periodic results in MSE and Spearman's correlation plots, which are presumably diurnal variations in this station. On the other hand, station 11126 in Fig 5.13, shows an inadequate performance of all models, The MSE error is high for all models, and AROME shows an underfitting problem in the bias plot. We assume this is due to the outliers problem, as we saw in Fig 3.2a and Fig 3.5b. Station 11126 had the most outliers among other stations in both AROME and Observation data-sets, and because of the location of station 11126 in Alpine region, wind speed prediction is a challenge.

As an example, we analyzed the wind speed forecasting in station 11044 for 1-hour, 2-hours, 6-hours, 25-hours, 36-hours, and 43-hours lead time. The reason to chose this station is that in Fig 5.11, the MSE plot of station 11044 shows different behavior in each model for each lead time. The black points are the observed wind speed points based on the test month of January 2017. The red line shows the predicted wind speed with the Persistence model; the predicted wind speed with the AROME model is shown in the blue line; and the green line shows the SVR model wind speed prediction.

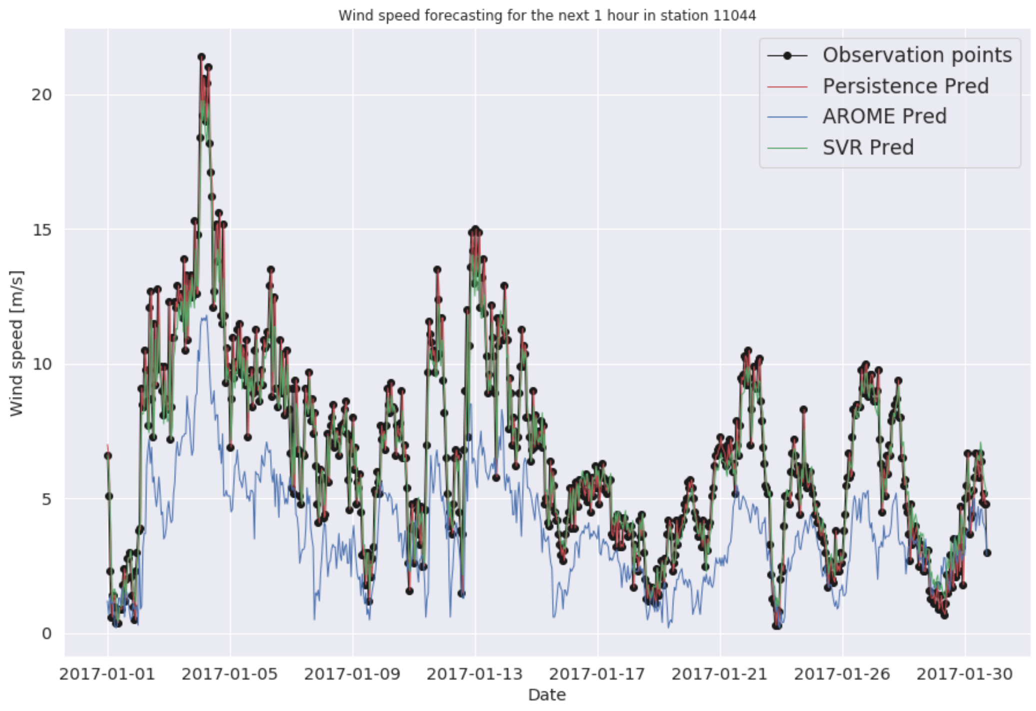


Figure 5.14: 1-hour wind speed forecasting for station 11044.

The Persistence and SVR models show a good forecast in Fig 5.14 and Fig 5.15, whereas the AROME model does not. In Fig 5.16, the Persistence model performed better, but again in 12-hours forecasting (Fig 5.17) performance of the SVR model is superior to the other two models. In 25-hours forecasting, we see in Fig 5.18 that the Persistence model has an insufficient

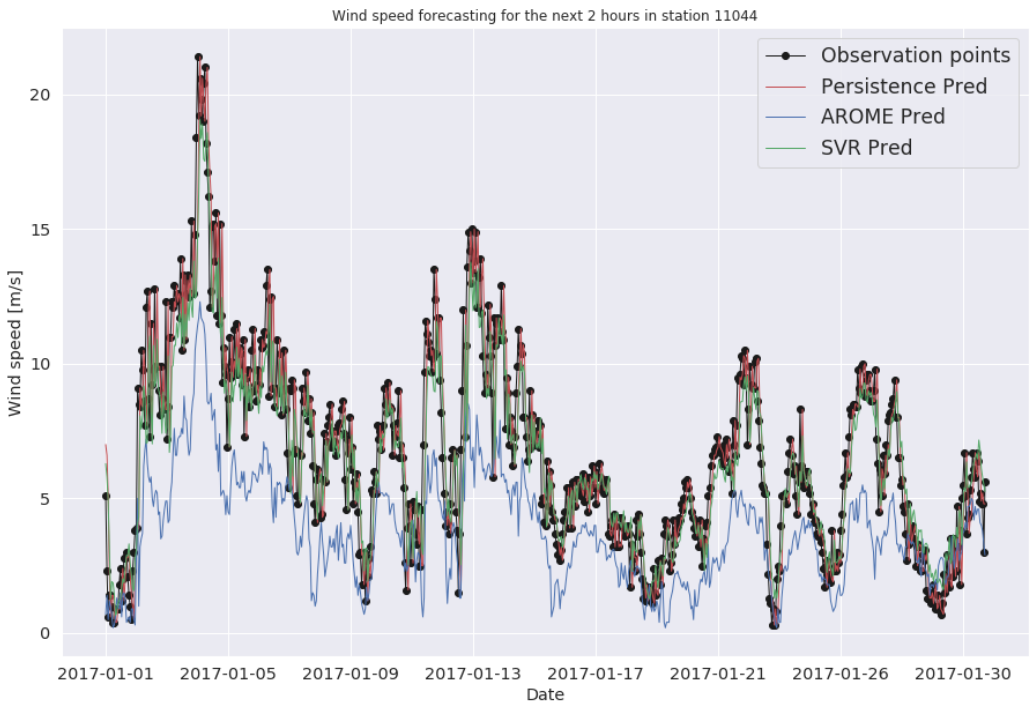


Figure 5.15: 2-hours wind speed forecasting for station 11044.

performance compared to the SVR and AROME models, and the AROME model performed slightly better. The behavior of the 36-hours and 43-hours lead times in Fig 5.19 and Fig 5.20 show better performance again with the SVR model. The differences between the models in the station 11044 could be because of the station's location at a higher altitude, and the data captured by this station is not represented reasonably enough.

Fig 5.21, 5.22 and 5.23 show wind speed predictions for 1-hour until 43-hours using the three models and compared them with observation points for three initial times in stations 11108 and 11126, and all stations respectively. The initial times were chosen randomly. The black color shows the observation points, the teal color shows the wind speed prediction of the SVR model, the blue color is the wind speed prediction of the Persistence model, and the brown color shows the wind speed prediction of the AROME model. It is hard to find a pattern in Fig 5.21 for station 11108, but we can see that the values of predicted wind speeds from the Persistence model have less difference than the observed wind speed compared to other models. Nevertheless, in Fig 5.22, it is more obvious that the AROME model does not have good performance compared to the SVR model in initial time *2017-01-21 05:00:00*, and the Persistence model shows better nowcasting (up to 6-hours) forecast. Overall, both the SVR and AROME models performed better in all initial

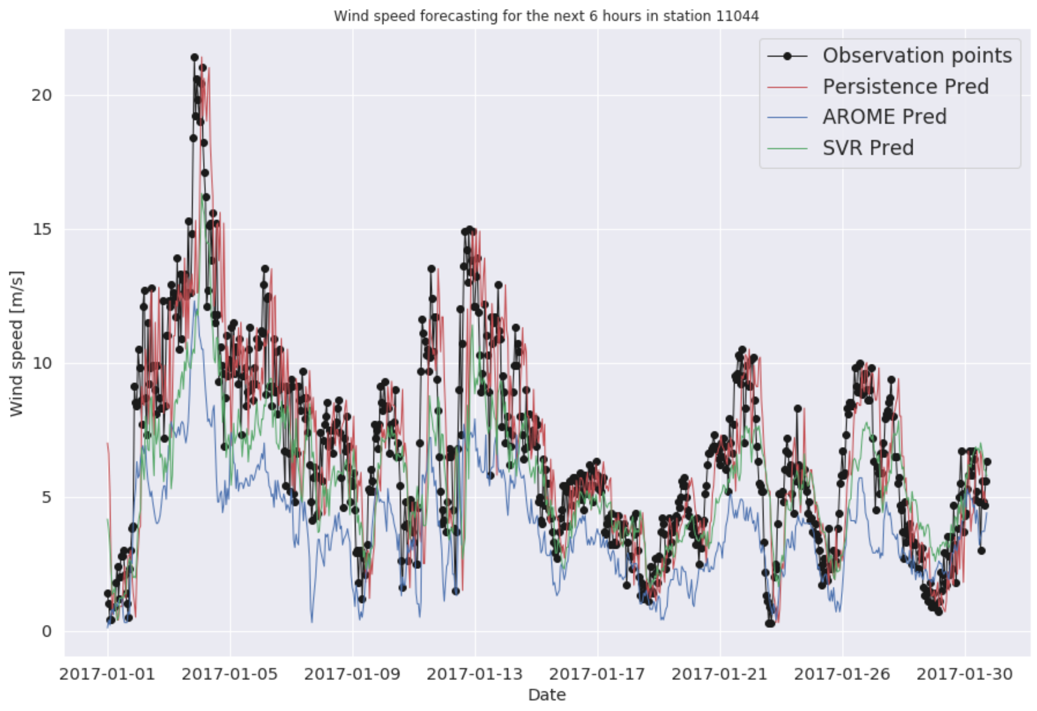


Figure 5.16: 6-hours wind speed forecasting for station 11044.

times compared to the Persistence model in Fig 5.23 for all stations, but in Fig 5.21a, the SVR model does not meet our expectations for predicting wind speed between 16-hours and 26-hours lead times. Some stations do not represent well-performed SVR models in these lead times for the initial time *2017-01-10 12:00:00*. Also, we can see that the Persistence model has a better forecasting performance for nowcasting for all stations in the initial time *2017-01-21 05:00:00*.

In Fig 5.24, we see the results of the three models on all stations. Fig 5.24 shows our SVR model outperforms the Persistence and AROME models. However, we have to consider the lead times between 6-17 hours; better performance of SVR here might suggest an under-fitting problem of the model. In general, the AROME model yields better error scores compared to the Persistence model. In conjunction with Observation data, AROME data provides sufficient information for learning (or improving existing) wind speed forecasts due to its higher spatial resolution and physical parameterizations. Besides, the Persistence model uses the most recent measurement for all forecast hours. Hence, in the Persistence model, we assume the value of the currently observed wind speed for all the future time steps. As we can see from Fig 5.24, this model works well for the next hours, but the quality deteriorates more than with more sophisticated approaches.

Fig 5.25 shows a map with the locations of the 28 selected stations. The different colors of

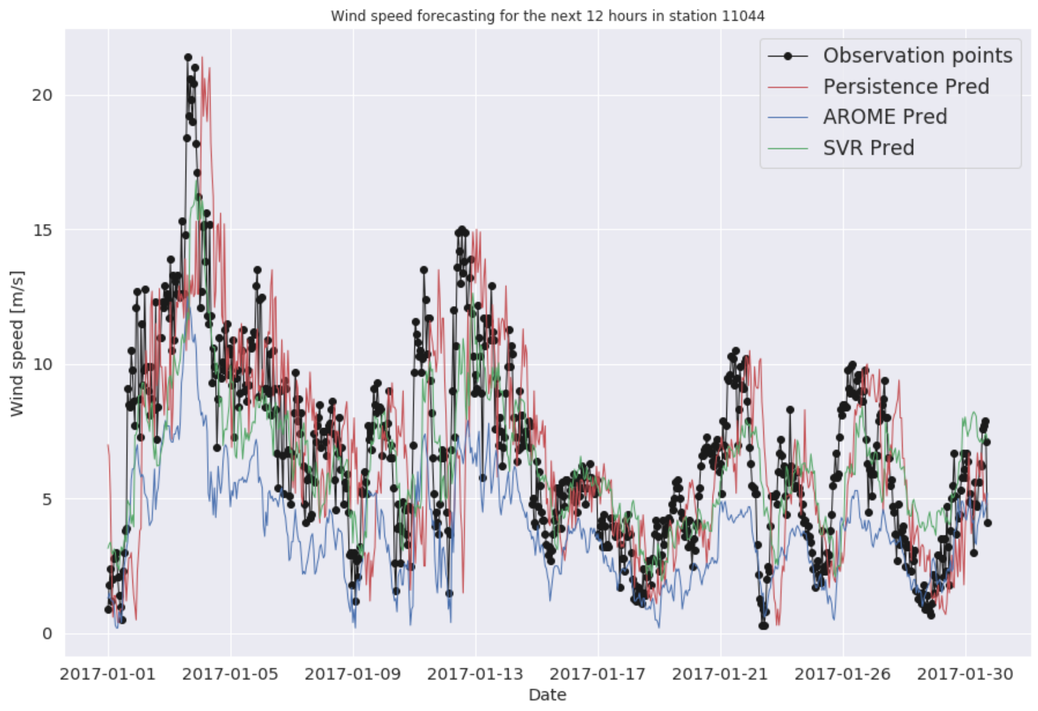


Figure 5.17: 12-hours wind speed forecasting for station 11044.

markers are indications of the MSE performance of each station. The stations with red markers indicate that the SVR performance was better than the AROME and Persistence models in terms of MSE; the blue markers indicate the AROME model had better performance compared to others, and the black markers show the stations with the lower MSE compared to the AROME and SVR models. This figure indicates that the model's performance is somehow related to the station's location for the test month of January 2017.

In Fig 5.26a, we see the bar-plot on their MSE performances of each station. The results show that the performances are region and altitude dependent. For example, in lower Austria, SVR performed better, or in upper Austria, the AROME model performed better. In Vienna, both SVR and AROME performed well in different stations. The reason for these differences might be the occurrence of unusual wind behavior in different parts of Vienna or an unusual occurrence of specific weather types. Station 11126 shows the worst error in all three models, but the AROME model performed best.

Overall results in Fig 5.26 indicate that the SVR with a linear kernel method achieves better forecast in short-range wind speed forecasts. The Persistence model performs better in the nowcasting range. However, for lead times beyond 6-hours, the SVR model performs better than the AROME and Persistence models. It can be concluded that the proposed SVR model with a

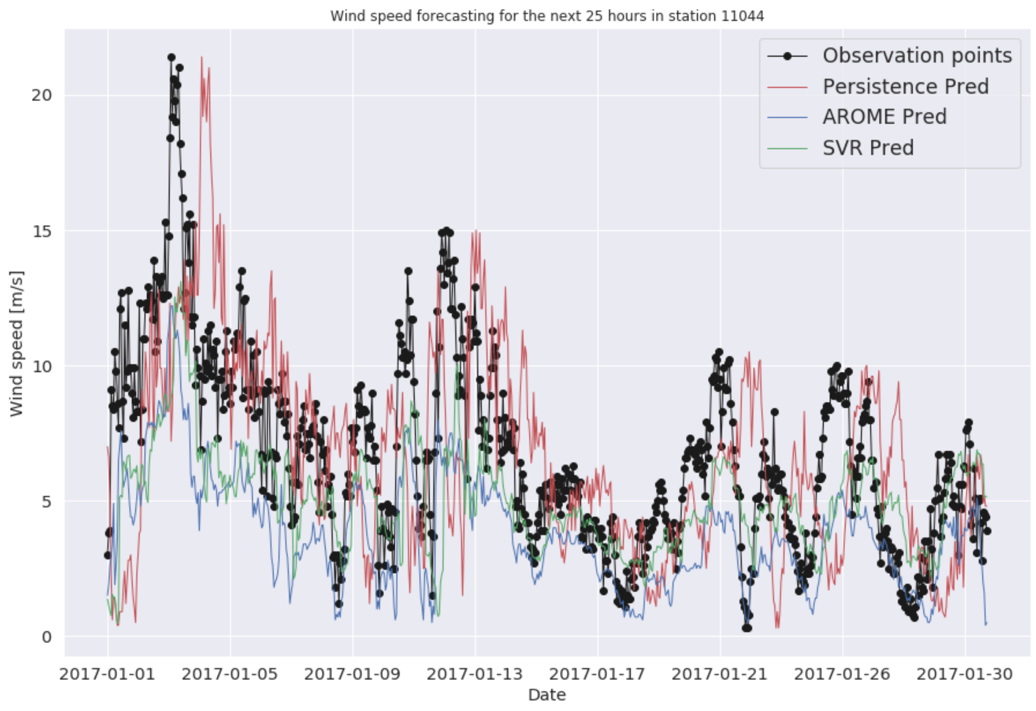


Figure 5.18: 25-hours wind speed forecasting for station 11044.

linear kernel is preferable for short-range wind speed prediction.

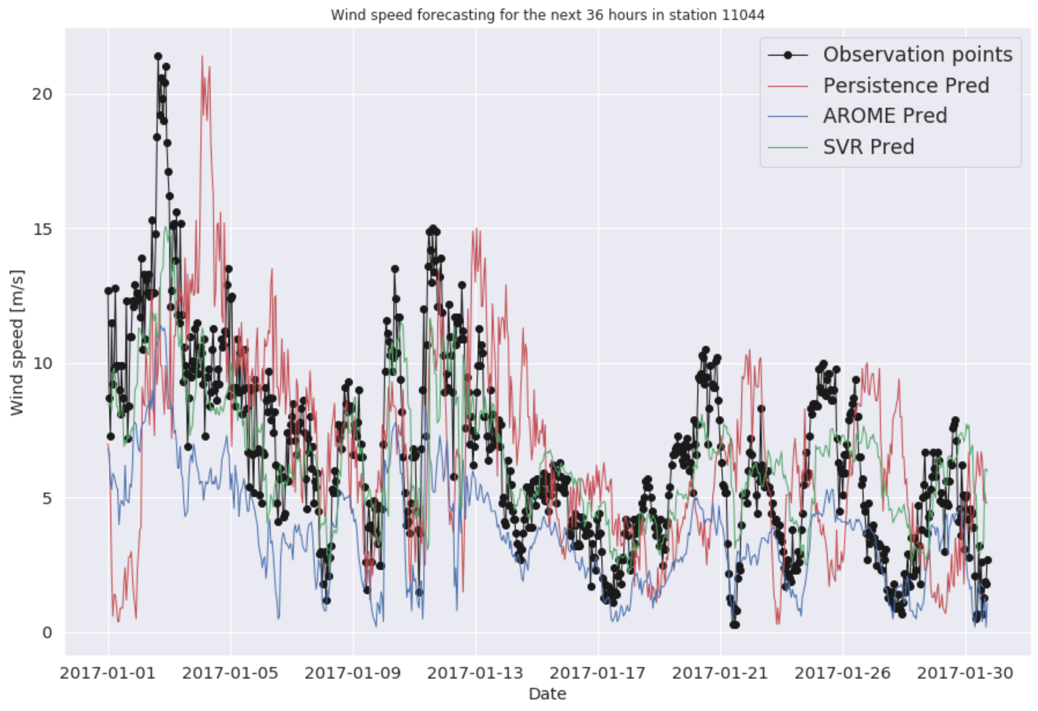


Figure 5.19: 36-hours wind speed forecasting for station 11044.

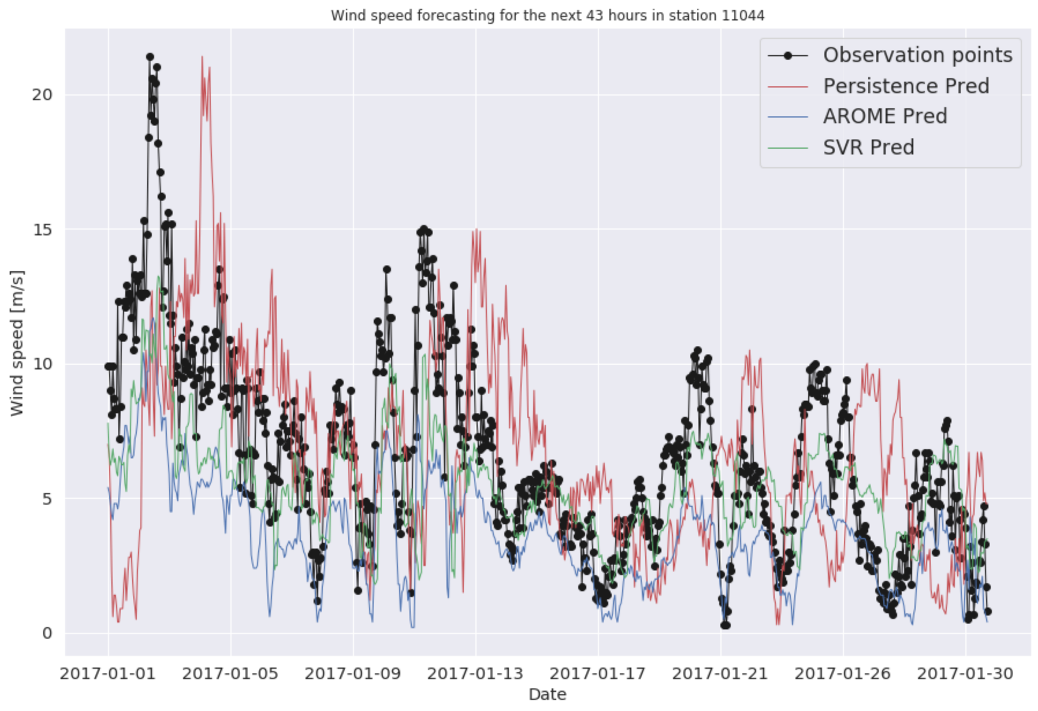
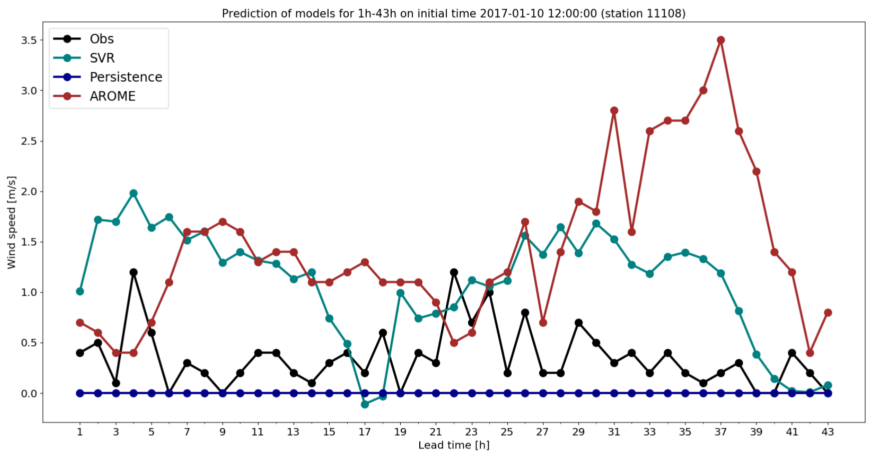
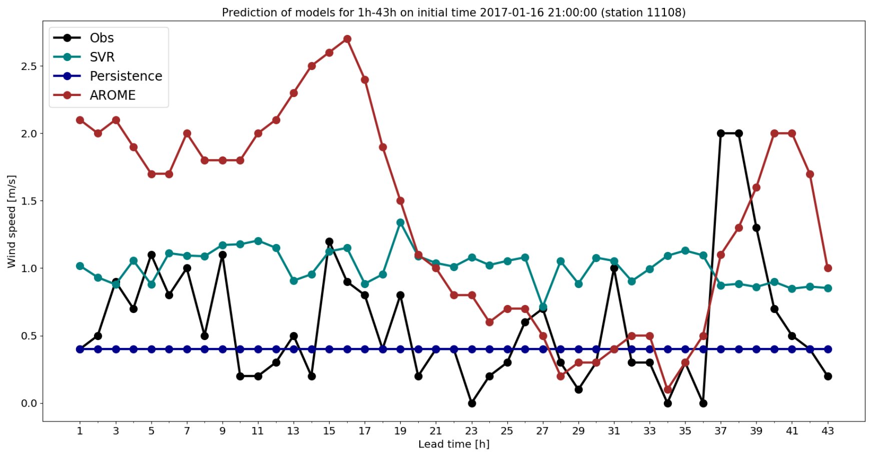


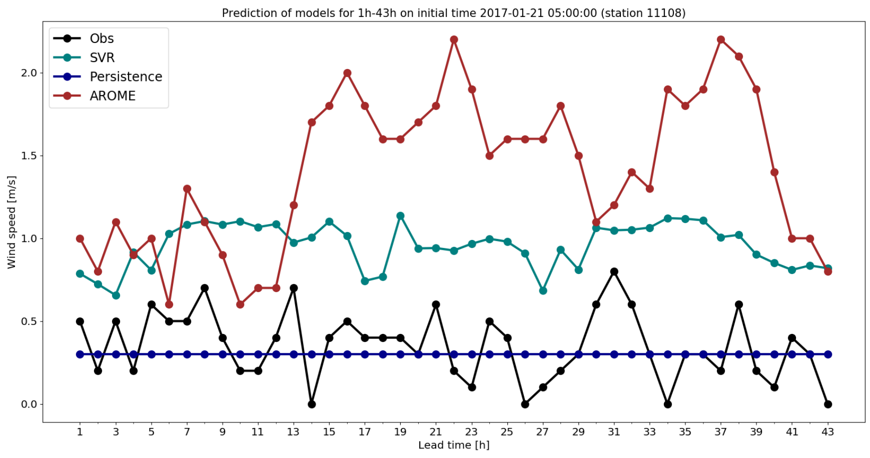
Figure 5.20: 43-hours wind speed forecasting for station 11044.



(a) Initial time: '2017-01-10 12:00:00'

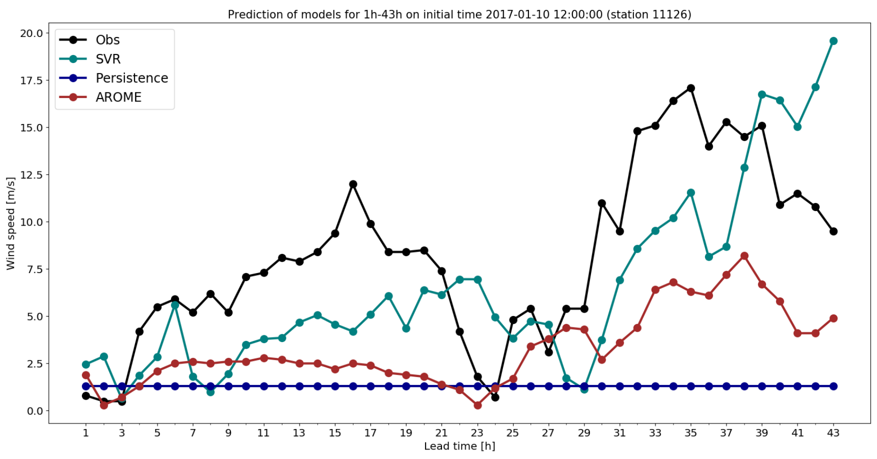


(b) Initial time: '2017-01-16 21:00:00'

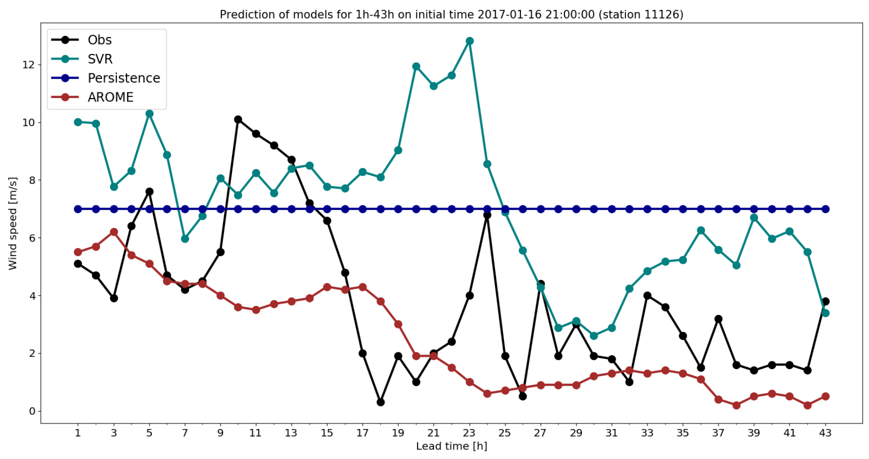


(c) Initial time: '2017-01-21 05:00:00'

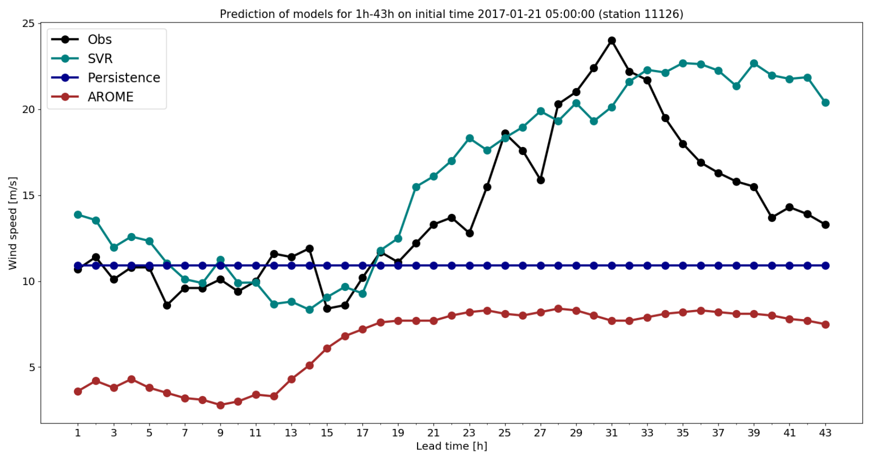
Figure 5.21: Comparison of models with three initial times for the station 11108.



(a) Initial time: '2017-01-10 12:00:00'

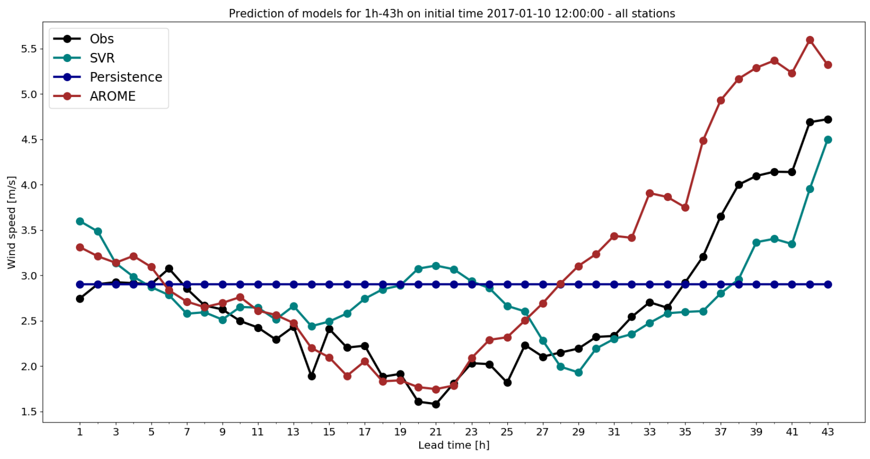


(b) Initial time: '2017-01-16 21:00:00'

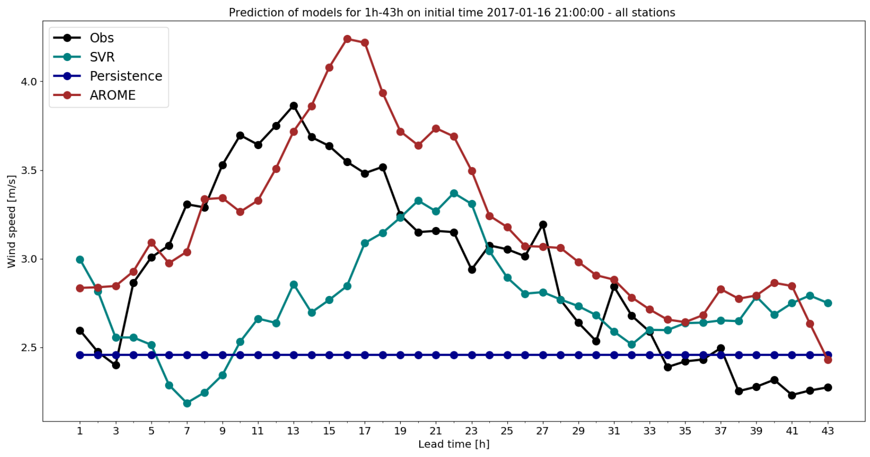


(c) Initial time: '2017-01-21 05:00:00'

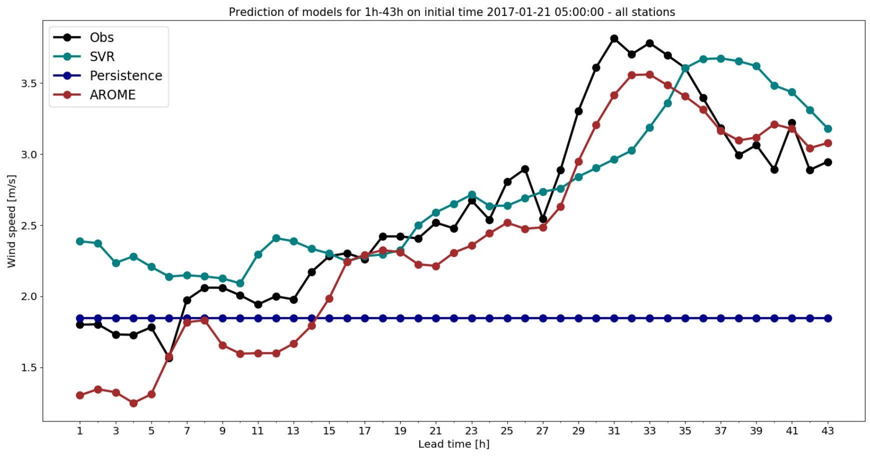
Figure 5.22: Comparison of models with three initial times for the station 11126.



(a) Initial time: '2017-01-10 12:00:00'



(b) Initial time: '2017-01-16 21:00:00'



(c) Initial time: '2017-01-21 05:00:00'

Figure 5.23: Comparison of models with three initial times for all stations.

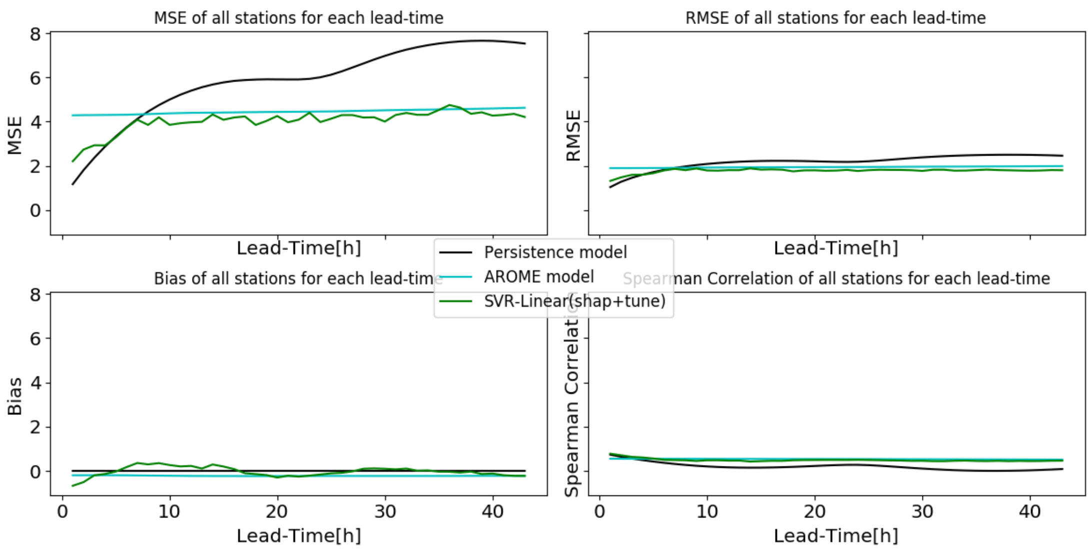


Figure 5.24: Comparison of three models for all lead times in all stations: SVR, Persistence and AROME models.

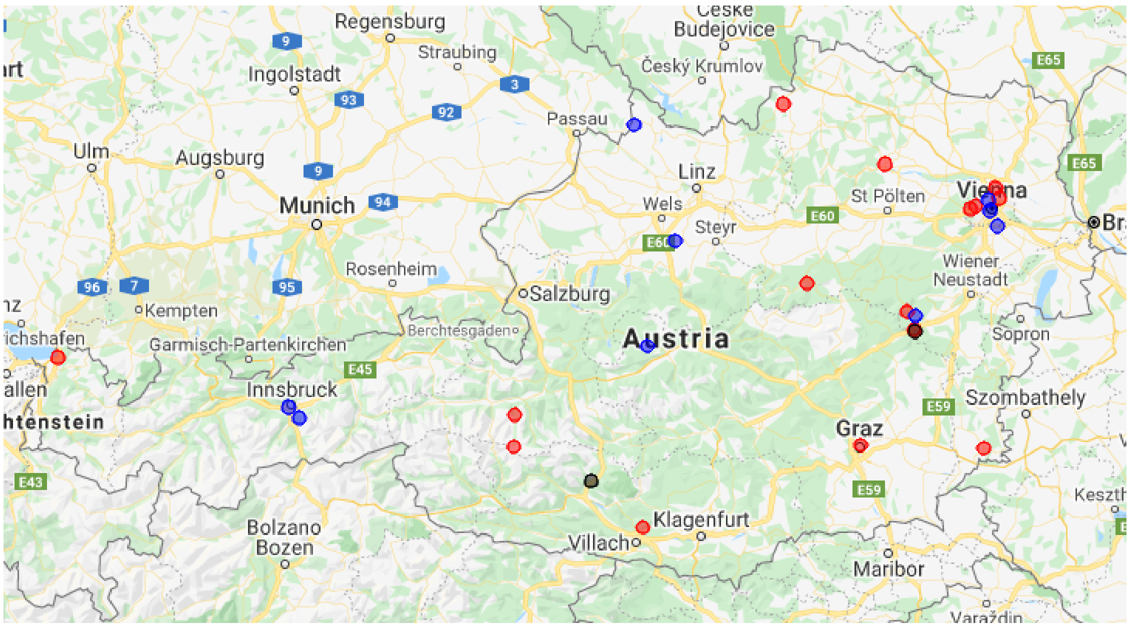
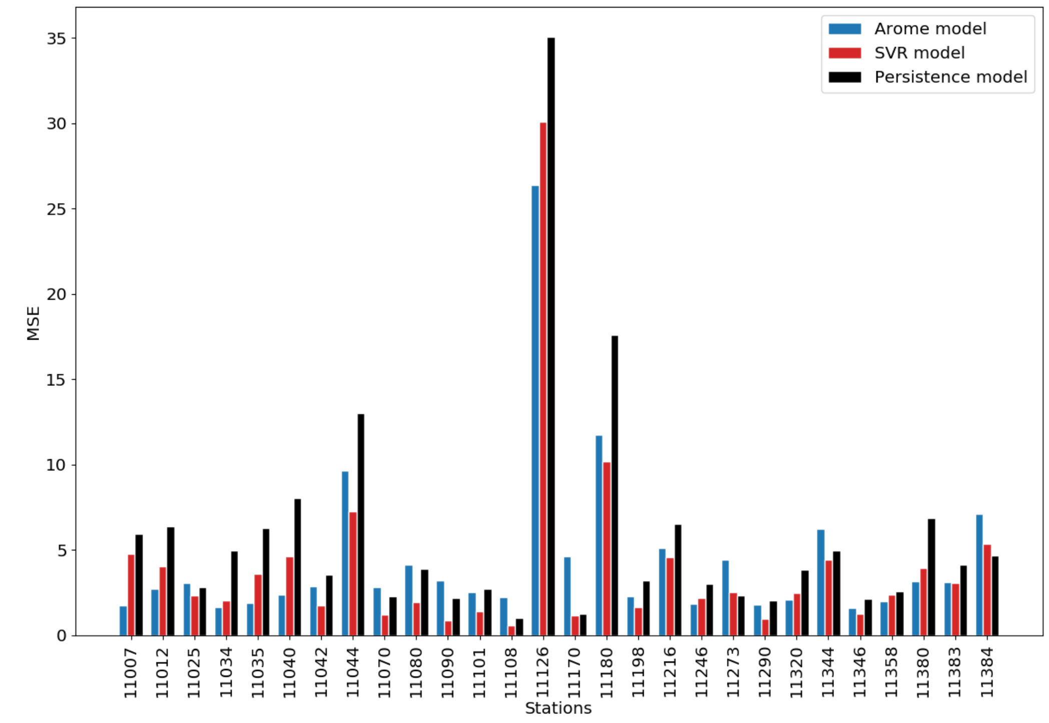
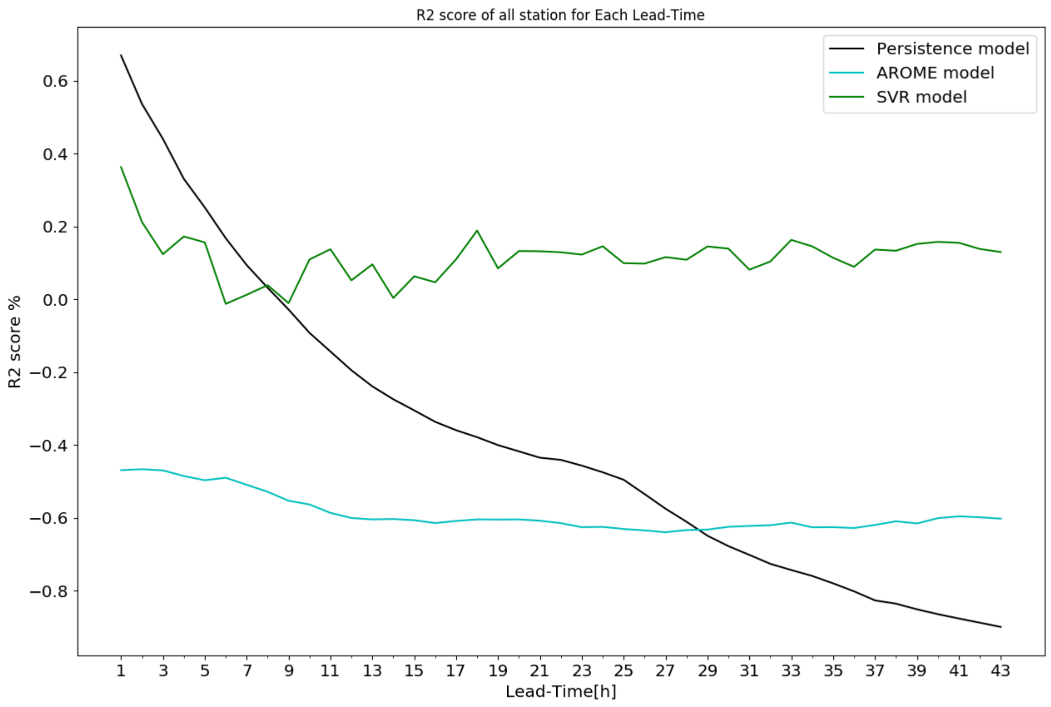


Figure 5.25: Map with stations with best MSE performance. Red: SVR best performance, Blue: AROME best performance, Black: Persistence best performance.



(a) Average MSE performance for 28 stations.



(b) Average R^2 score of all station for each lead time.

Figure 5.26: a) Average MSE of all stations. and b) R^2 score for all station in each lead time.

Chapter 6

Discussion

Discussion of Results

In this thesis, we investigated predicting the wind speed in the short-range (0-43 hours ahead) by using a support vector regression model. The SVR's performance depends on the selection of the kernel function, selection of features, and optimizing the hyper-parameters. Linear, RBF, sigmoid, and polynomial kernel functions were chosen to build the SVR model for the short-range wind speed prediction. We excluded the polynomial kernel from the preliminary experiments because it could not converge with the data-sets. The forecast results show that the SVR model based on the linear kernel function is superior to that of other kernel functions.

We extended the SVR model by using the SHAP method for selecting the most important features. By *Grid Search* method, we optimized the parameters of the SVR's kernel function. Two different approaches for feature selection, namely SVR (fs(unique)+tune) and SVR (shap+tune), were studied and compared with a simple SVR-Linear model. We showed that a selection of the 20 most important features by the SHAP method for each lead time for each individual station, can improve the forecast performance. The SVR model was applied to forecast the short-range wind speed. Compared with the forecast results of the Persistence and AROME model, the SVR model can improve the prediction accuracy of the short-range wind speed.

In comparison to the AROME and Persistence model, we reduce the prediction error by using the SVR model for most stations. In particular, 16 stations out of 28 stations showed fewer prediction errors with SVR. The mean RMSE for all lead times in all stations for AROME, Persistence, and SVR model is 2.07, 2.26, and 1.79 respectively, which means the mean improvement of SVR in the prediction error in comparison to the Persistence model is 22% and to the AROME model is 15%. The average R^2 score for AROME, Persistence, and SVR model is -58%, -38%, 11%, respectively, which shows SVR's superior performance compared to the other models. For some stations, the SVR model does not yield improvements over the AROME model. Further research is necessary, e.g., calculating the SVR parameter choices for each lead times in each station by optimization algorithms, or dividing the data into testing-set and training-set with other different periods.

A disadvantage of the SVR model is the high algorithmic complexity and long training times. The Persistence model performs better, especially for nowcasting with a time horizon up to 6-hours, but for longer time horizons, the machine learning model usually performs better than the Persistence model. Nonetheless, the Persistence model could be an excellent alternative only in nowcasting and simple situations.

To further improve the models, various measures can be taken. More advanced outlier detection can be applied. For instance, we saw that station 11126 suffered from the wrong prediction because it contained many outliers. It would be interesting to investigate more on the nature of the outliers in the data-sets. One idea would be to carry out a transformation for outliers, like log transformations. Another solution for the station with high altitudes would be to use another grid point in those areas or use an average of near grid points. Additionally, it is recommended to do an in-depth investigation of feature selection. In an operational setup, applying one method of feature selection may cause over-fitting or under-fitting, but combining two or three feature selection methods and building one model from those adjusted selections can improve the results and give more knowledge about the stations and their data.

Overall, the performance of the machine learning model support vector regression was more successful when compared to the AROME and Persistence models in predicting short-range wind speed based on available data with reasonable accuracy.

Outlook

The empirical analysis results have highlighted that with applying SVR, we achieved a significant improvement in the overall performance. However, we can not guarantee that this is the global optimum for each station in each lead time. The obtained results suggest the use of SVR as a forecasting model at ZAMG, among their other statistical and physical techniques for short-range wind speed prediction, along with further improvements.

Bibliography

(WMO), World Meteorological Organization (Updated 2010[a]). *"Guide to Meteorological Instruments and Methods of Observation - Part 1, Chapter 5"*.

— (Updated 2010[b]). *"Guide to Meteorological Instruments and Methods of Observation - Part 2, Chapter 2"*.

— (n.d.). *"Definitions of Meteorological Forecasting Ranges"*. URL: `https://www.wmo.int/pages/prog/www/DPS/GDPS-Supplement5-AppI-4.html`.

A, Ucar and Balo F (2009). ‘"Investigation of wind characteristics and assessment of wind-generation potentiality in Uludağ-Bursa, Turkey"’. In: DOI: `https://doi.org/10.1016/j.apenergy.2008.05.001`.

A.Kalogirou, Soteris (2001). ‘"Artificial neural networks in renewable energy systems applications: a review"’. In: DOI: `https://doi.org/10.1016/S1364-0321(01)00006-5`.

A.Sfetsos (2000). ‘"A comparison of various forecasting techniques applied to mean hourly wind speed time series"’. In: DOI: `https://doi.org/10.1016/S0960-1481(99)00125-1`.

A.Shamshad et al. (2005). ‘"First and second order Markov chain models for synthetic generation of wind speed time series"’. In: DOI: `https://doi.org/10.1016/j.energy.2004.05.026`.

Abdi, Hervé and Lynne J. Williams (2010). *"Principal component analysis"*. Springer, New York, NY. DOI: `https://doi.org/10.1007/b98835`.

Academy, Cambridge Coding (n.d.). *"Expanding your machine learning toolkit: Randomized search, computational budgets, and new algorithms"*. URL: `https://cambridgecoding.wordpress.com/2016/05/16/expanding-your-machine-learning-toolkit-randomized-search-computational-budgets-and-new-algorithms-2/`.

AFAC Research Dissemination Pilot Study, Rick McRae for (2012). *"The types of foehn winds and the conditions for theirformation"*. URL: `http://www.highfirerisk.com.au/resdis/hiwx_foehn_g_text.htm`.

Alexandre Costaa and, Antonio Crespob et al. (2007). ‘"A review on the young history of the wind powershort-term prediction"’. In: DOI: http://henrikmadsen.org/wp-content/uploads/2014/05/Journal_article_-_2008_-_A_review_on_the_young_history_of_the_wind_power_short-term_prediction.pdf.

BC, Ross (2014). ‘"Mutual Information between Discrete and Continuous Data Sets"’. In: DOI: https://doi.org/10.1371/journal.pone.0087357.

Boser, Bernhard E., Isabelle M. Guyon and Vladimir N Vapnik (1992). ‘"A training algorithm for optimal margin classifiers"’. In: DOI: https://doi.org/10.1145/130385.130401.

Bossanyi, E.A. (1985). ‘"Short-Term Wind Prediction Using Kalman Filters"’. In: *Wind Engineering*. ISSN: 0309524X, 2048402X. URL: http://www.jstor.org/stable/43749025.

C.M., Salgado et al. (2016). *" Missing Data. In: Secondary Analysis of Electronic Health Records"*. Springer, Cham. ISBN: 978-3-319-43742-2. DOI: https://doi.org/10.1007/978-3-319-43742-2_13.

Carro-Calvoa, L. et al. (2011). ‘"Wind speed reconstruction from synoptic pressure patterns using an evolutionary algorithm"’. In: DOI: https://doi.org/10.1016/j.apenergy.2011.07.044.

Cherkassky, Vladimir and Yunqian Ma (2004). ‘"Practical selection of SVM parameters and noise estimation for SVM regression"’. In: DOI: https://doi.org/10.1016/S0893-6080(03)00169-2.

Coiffier, Jean (2011). *"Fundamentals of Numerical Weather Prediction"*. Cambridge University Press. DOI: 10.1017/CBO9780511734458.

Corazza, Anna et al. (2010). ‘"How effective is Tabu Search to configure Support Vector Regression for effort estimation?"’ In: DOI: https://doi.org/10.1007/s10664-011-9187-3.

Cortes, C. and V. Vapnik (1995). ‘"Machine Learning"’. In: DOI: https://doi.org/10.1007/BF00994018.

Cover, Thomas M. and Joy A. Thomas (n.d.). *"Elements of Information Theory"*. ISBN: 9780471062592. URL: http://staff.ustc.edu.cn/~cgong821/Wiley.Interscience.Elements.of.Information.Theory.Jul.2006.eBook-DDU.pdf.

D.A.Fadare (2010). ‘"The application of artificial neural networks to mapping of wind speed profile for energy application in Nigeria"’. In: DOI: https://doi.org/10.1016/j.apenergy.2009.09.005.

Erdem, Ergin and Jing Shi (2011). ‘"ARMA based approaches for forecasting the tuple of wind speed and direction"’. In: DOI: https://doi.org/10.1016/j.apenergy.2010.10.031.

F.M.Coimbra, Carlos and Hugo T.C.Pedro (2013). *"Solar Energy Forecasting and Resource Assessment"*. Elsevier, pp. 383–406. DOI: `https://doi.org/10.1016/C2011-0-07022-9`.

Fieller, E. C., H. Hartley and E. S. Pearson (1957). *"TESTS FOR RANK CORRELATION COEFFICIENTS. I"*. DOI: `https://doi.org/10.1093/biomet/44.3-4.470`.

G, Xydis, Koroneos C and Loizidou M (2009). ‘"Exergy analysis in a wind speed prognostic model as a wind farm sitting selection tool: A case study in Southern Greece"’. In: DOI: `10.1016/j.apenergy.2009.03.017`.

G., Kariniotakis, Nogaret E. and Stavrakakis G. (1996). ‘"A Fuzzy Logic and a Neural Network Based Wind Power Forecasting Model "’. In:

Hardy, Michael (2002). *"An Illuminating Counterexample"*. URL: `%7Bhttp://www.jstor.org/stable/3647938%7D`.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman (2008). *"The Elements of Statistical Learning"*. Great Britain: Springer. ISBN: 0-13-604259-7.

Heimann, D. et al. (2007). *"ALPNAP Air Pollution, Traffic Noise and Related Health Effects in the Alpine Space"*. 1st ed. Università degli Studi di Trento, Dipartimento di Ingegneria Civile e Ambientale. ISBN: 978-88-8443-208-7.

Hong, Wei-Chiang (2008). ‘"Rainfall forecasting by technological machine learning models"’. In: DOI: `https://doi.org/10.1016/j.amc.2007.10.046`.

Horváth, Gábor (2003). ‘"Neural networks from the perspective of measurement systems"’. In: DOI: `https://doi.org/10.1109/IMTC.2003.1207924`.

Hsu, Chih Wei, Chih Chung Chang and Chih Jen Lin (2003). ‘"A Practical Guide to Support Vector Classification"’. In: DOI: `https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

I.T, Jolliffe (2002). *"Principal Component Analysis"*. ISBN: 978-0-387-22440-4.

J.A., Lujano-Rojas J.M.and Bernal-Agustín J.L.and Dufo-López R.and Domínguez-Navarro (2011). ‘"Forecast of Hourly Average Wind Speed Using ARMA Model with Discrete Probability Transformation"’. In: DOI: `https://doi.org/10.1007/978-3-642-21765-4_125`.

J.Smola, Alex and Bernhard Schölkopf (1998). ‘"A Tutorial on Support Vector Regression"’. In: DOI: `http://www.svms.org/regression/SmSc98.pdf`.

K, Sreelakshmi and Kumar PR (2008). ‘"Performance Evaluation of Short Term Wind Speed Prediction Techniques "’. In:

Kariniotakis, G.N., G.S. Stavrakakis and E.F. Nogaret (1996). ‘"Wind power forecasting using advanced neural networks models"’. In: DOI: https://doi.org/10.1109/60.556376.

Kaushik, Saurav (n.d.). *"Introduction to Feature Selection methods with an example (or how to select the right variables?)"* URL: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/.

Keerthi, S and Chih-Jen Lin (2003). ‘"Asymptotic Behaviors Of Support Vector Machines with Gaussian kernel"’. In: DOI: https://doi.org/10.1162/089976603321891855.

Ketterer, C. et al. (2014). *"Investigation of the Planetary Boundary Layer in the Swiss Alps Using Remote Sensing and In Situ Measurements"*. DOI: https://doi.org/10.1007/s10546-013-9897-8.

Kleynhans, Tania et al. (2017). *"Predicting Top-of-Atmosphere Thermal Radiance Using MERRA-2 Atmospheric Data with Deep Learning"*. URL: https://www.researchgate.net/publication/320916953_Predicting_Top-of-Atmosphere_Thermal_Radiance_Using_MERRA-2_Atmospheric_Data_with_Deep_Learning/figures?lo=1.

Kraskov, Alexander, Harald Stögbauer, Ralph G. Andrzejak et al. (2003). *"Hierarchical Clustering Based on Mutual Information"*. DOI: https://arxiv.org/abs/q-bio/0311039.

Kraskov, Alexander, Harald Stögbauer and Peter Grassberger (2004). ‘"Estimating mutual information"’. In: DOI: https://link.aps.org/doi/10.1103/PhysRevE.83.019903.

Kusiak, Andrew and Wenyan Li (2010). ‘"Short-term prediction of wind power with a clustering approach "’. In: DOI: https://doi.org/10.1016/j.renene.2010.03.027.

Kusiak, Andrew, Haiyang Zheng and Zhe Song (2009). ‘"Short-Term Prediction of Wind Farm Power: A Data Mining Approach "’. In: DOI: https://doi.org/10.1109/TEC.2008.2006552.

Lei, Ma et al. (2009a). ‘"A review on the forecasting of wind speed and generated power"’. In: DOI: https://doi.org/10.1016/j.rser.2008.02.002.

— (2009b). ‘"A review on the forecasting of wind speed and generated power"’. In: 13, pp. 915–920. DOI: https://www.sciencedirect.com/science/article/pii/S1364032108000282.

Li, Gong and Jing Shi (2010). ‘"On comparing three artificial neural networks for wind speed forecasting"’. In: DOI: https://doi.org/10.1016/j.apenergy.2009.12.013.

Li, Gong, Jing Shi and Junyi Zhou (2011). ‘"Bayesian adaptive combination of short-term wind speed forecasts from neural network models"’. In: DOI: https://doi.org/10.1016/j.renene.2010.06.049.

Lin, Amanda Yan, Mengcheng Zhang and Selpi (2017). ‘"Combining Support Vector Regression with Scaling Methods for Highway Tollgates Travel Time and Volume Predictions"’. In: DOI: https://core.ac.uk/download/pdf/94335329.pdf.

Lin, Gwo-Fong et al. (2009). ‘"Support vector machine-based models for hourly reservoir inflow forecasting during typhoon-warning periods"’. In: DOI: https://doi.org/10.1016/j.jhydrol.2009.03.032.

Lin, Hsuan-Tien and Chih-Jen Lin (2003). ‘"A Study on Sigmoid Kernels for SVM and the Training ofnon-PSD Kernels by SMO-type Methods"’. In: URL: https://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf.

Lundberg, Scott M. and Su-In Lee (2017). ‘"A Unified Approach to Interpreting Model Predictions"’. In: DOI: https://dl.acm.org/doi/pdf/10.5555/3295222.3295230?download=true.

M.A.Mohandes et al. (2004). ‘"Support vector machines for wind speed prediction"’. In: DOI: https://doi.org/10.1016/j.renene.2003.11.009.

Manwell, J. F., J. G. McGowan and A. L. Rogers (2009). *"WIND ENERGY EXPLAINED"*. Great Britain: John Wiley and Sons.

Molnar, Christoph (n.d.). *"Interpretable Machine Learning". "A Guide for Making Black Box Models Explainable"*.

Monfared, Mohammad, Hasan Rastegar and Hossein Madadi Kojabadi (2009). ‘"A new strategy for wind speed forecasting using artificial intelligent methods"’. In: DOI: https://doi.org/10.1016/j.renene.2008.04.017.

More, Anurag and M.C.Deo (2003). ‘"Forecasting wind with neural networks"’. In: DOI: https://doi.org/10.1016/S0951-8339(02)00053-9.

O.Carranzaa et al. (2011). ‘"Comparative study of speed estimators with highly noisy measurement signals for Wind Energy Generation Systems"’. In: DOI: https://doi.org/10.1016/j.apenergy.2010.07.039.

Osowski, Stanislaw and Konrad Garanty (2007). ‘"Forecasting of the daily meteorological pollution using wavelets and support vector machine"’. In: DOI: https://doi.org/10.1016/j.engappai.2006.10.008.

P.Pinson et al. (2009). '"Skill forecasting from ensemble predictions of wind power"'. In: DOI: https://doi.org/10.1016/j.apenergy.2008.10.009.

Papazek, Petrina (n.d.). '"Generalization and application of the flux-conservative thermodynamic equations in the AROME model of the ALADIN system"'. In: (). DOI: https://doi.org/10.5194/gmd-2015-279.

Poggi, P et al. (2003). '"Application of wind speed forecasting to the integration of wind energy into a large scale power system"'. In: DOI: https://doi.org/10.1016/S0196-8904(03)00108-0.

R.Ataa and Y.Kocyigit (2010). '"An adaptive neuro-fuzzy inference system approach for prediction of tip speed ratio in wind turbines"'. In: DOI: https://doi.org/10.1016/j.eswa.2010.02.068.

Ribeiro, Marco Tulio, Sameer Singh and Carlos Guestrin (2016). '"Why Should I Trust You?": Explaining the Predictions of Any Classifier'. In: pp. 1135–1144. DOI: https://doi.org/10.1145/2939672.2939778.

Rösemann, Reinhold (2011). *"Solar Radiation Measurement"*. Kipp and Zonen.

Rushing, John et al. (2005). '"ADaM: a data mining toolkit for scientists and engineers"'. In: DOI: https://doi.org/10.1016/j.cageo.2004.11.009.

Russell, Stuart J. and Peter Norvig (1995). *"Artificial Intelligence: A Modern Approach"*. Great Britain: Prentice Hall. ISBN: 0-13-604259-7.

S.Rehman, M.Mohandes ans and S.M.Rahmand (2011). '"Estimation of wind speed profile using adaptive neuro-fuzzy inference system (ANFIS)"'. In: DOI: https://doi.org/10.1016/j.apenergy.2011.04.015.

Salcedo-Sanz, Sancho et al. (2011). '"Short term wind speed prediction based on evolutionary support vector regression algorithms"'. In: DOI: https://doi.org/10.1016/j.eswa.2010.09.067.

Scikit-Learn (n.d.). *"PCA"*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

Seity, Y and P Brousseau (2011). '"The AROME-France Convective-Scale Operational Model"'. In: DOI: https://doi.org/10.1175/2010MWR3425.1.

Shi, Jing, Jinmei Guo and Songtao Zheng (2012). '"Evaluation of hybrid forecasting approaches for wind speed and power generation time series"'. In: DOI: https://doi.org/10.1016/j.rser.2012.02.044.

Silva, Gonçalo Xufre, Pedro M. Fonte and José Carlos Quadrado (2006). ‘"Radial basis function networks for wind speed prediction"’. In:

T.B., Trafalis, Adrianto I. and Richman M.B. (2007). ‘"Active Learning with Support Vector Machines for Tornado Prediction"’. In: DOI: https://doi.org/10.1007/978-3-540-72584-8_148.

Vapnik, Vladimir, Steven E. Golowich and Alex J. Smola (1997). *"Support Vector Method for Function Approximation, Regression Estimation and Signal Processing"*. MIT Press. URL: http://papers.nips.cc/paper/1187-support-vector-method-for-function-approximation-regression-estimation-and-signal-processing.pdf.

Wang, X. et al. (2004). ‘"Wind speed forecasting for power system operational planning "’. In:

Watson, S.J., L. Landberg and J.A. Halliday (1994). ‘"Application of wind speed forecasting to the integration of wind energy into a large scale power system"’. In: DOI: IEEProc.Gener.Transm.Distrib..

Widodo, Achmad et al. (2009). ‘"Fault diagnosis of low speed bearing based on relevance vector machine and support vector machine"’. In: DOI: https://doi.org/10.1016/j.eswa.2008.09.033.

Wikipedia (n.d.). *"Analysis of algorithms"*. URL: https://en.wikipedia.org/wiki/Analysis_of_algorithms.

wikipedia (2011). *"Kernel Machine"*. URL: https://commons.wikimedia.org/wiki/File:Kernel_Machine.png.

— (2012). *"Svm separating hyperplanes"*. URL: https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes.png.

Al-Yahyai, Sultan, Yassine Charabi and Adel Gastli (2010). ‘"Review of the use of Numerical Weather Prediction (NWP) Models for wind energy assessment"’. In: DOI: https://doi.org/10.1016/j.rser.2010.07.001.

Yesilbudak, Mehmet, Seref Sagiroglu and Ilhami Colak (2013). ‘"A new approach to very short term wind speed prediction using k-nearest neighbor classification"’. In: DOI: https://doi.org/10.1016/j.enconman.2013.01.033.

Zhou, Junyi and Jing Shi (2009). ‘"Performance evaluation of object localization based on active radio frequency identification technology"’. In: DOI: https://doi.org/10.1016/j.compind.2009.05.002.