Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Temperature correction in meteorological forecasts

Diploma thesis

2018

Bc. Richard Ižip

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Temperature correction in meteorological forecasts

Diploma thesis

Study programme:	Computer Science
Study field:	2508 Computer Science, Informatics
Department:	Department of Computer Science
Supervisor:	doc. RNDr. Robert Lukoťka PhD.

Bratislava, 2018
Bc. Richard Ižip

62997643

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Richard Ižip
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science, Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Temperature correction in meteorological forecasts

Annotation: The weather forecasts today are mainly based on numerical models, that simulate atmospheric processes. As numerical models do not need to produce unbiased estimation, statistical models can be used to find and correct potential bias in the forecast. The goal of the thesis is to create statistical and machine learning models that could improve temperature predictions of numeric weather forecasts. One of the problems, observed in the forecasts by SHMI forecasters is that when the weather is stable for several days, the forecast typically makes the same error each day (or the errors are corrected only very slowly). The thesis should explore the possibilities how to improve the model reaction during stable weather. A side goal is to look for statistical anomalies in the data that can be used to further improve the numerical forecast model.

Supervisor:	doc. RNDr. Robert Lukoťka, PhD.
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	05.10.2016
Approved:	14.12.2016

prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

....
Student

....
Supervisor

62997643

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Richard Ižip
Študijný program:	informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Temperature correction in meteorological forecasts
Korekcia teplôt meteorologických predpovedí

Anotácia: Základom predpovedí počasia sú numerické modely, ktoré simulujú dianie v atmosfére. Výstup takýchto numerických modelov je možné spresniť s použitím štatistických nástrojov, vďaka ktorým je možné odhaliť a upraviť prípadné vychýlenie predpovedí. Cieľom tejto práce je vytvoriť modely, ktoré sú schopné vylepšiť teplotné predikcie numerických modelov. Jednou z pozorovaných slabín numerických predpovedí je, že ak je niekoľko dní počasie stabilné, predpoveď sa stále dopúšťa rovnakých chýb (resp. chyby sa upravujú veľmi pomaly vzhľadom na veľkosť učiaceho obdobia modelu). Práca by mala hľadať spôsoby ako vylepšiť model pri stabilnom počasí. Bočným cieľom je hľadať štatistické anomálie v predpovediach, ktoré by mohli byť použité na následné vylepšenie numerického modelu.

Vedúci:	doc. RNDr. Robert Lukoťka, PhD.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 05.10.2016

Dátum schválenia: 14.12.2016

prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

....

študent vedúci práce

Acknowledgment I would like to thank my supervisor doc. RNDr. Robert Lukoťka PhD. for his advices and consultations during writing of the thesis.

Abstract

This thesis deals with improving temperature forecasts created by meteorological model Aladin, that is used by Slovak Hydrometeorological Institute. The goal is to further improve results achieved in the diploma thesis written by Michal Hajdin in 2016 and experiment with models which include their previous errors as new features. We present analysis of data we worked with, new approaches that further improved the results, and the results reached after experimenting with various learning algorithms and using previous error terms. Finally, we present improvement obtained for all the meteorological stations, whose data were available for our thesis. Our final model decreased absolute error of Aladin predictions from 18.95% up to 73.61%.

Keywords:

temperature forecasting, linear regression, rolling window

Abstrakt

V práci sa zaoberáme zlepšovaním predpovedí teploty, ktoré boli vytvorené meteorologickým modelom Aladin, používaným Slovenský hydrometeorologickým ústavom. Cieľom práce je ďalej zlepšiť výsledky dosiahnuté Michalom Hajdinom vo svojej diplomovej práci z roku 2016. Taktiež je cieľom experimentovanie s modelmi, ktoré zahŕňajú svoje predchádzajúce chyby vo svojich príznakoch. V práci prezentujeme analýzu dát s ktorými sme pracovali, nové spôsoby pomáhajúce znížiť chybu predpovedí a výsledky dosiahnuté po použití iných učiacich algoritmov a zahrnutí predchádzajúcich chýb modelu. Na záver prezentujeme dosiahnuté zlepšenie pre všetky meteorologické stanice, ktorých dáta sme mali k dispozícií. V porovnaní s modelom Aladin sme dosiahli zmenšenie absolútnej chyby v rozmedzí 18.95% až 73.61%.

Kľúčové slová:

predpoveď teploty, lineárna regresia, posuvné okno

Contents

List of Figures

Introduction

Temperature forecast is an important component in decision-making related to many areas, such as energy management, agricultural production or human and animal health [10]. It influences everyday life of mankind, helps to prepare for extreme conditions or to plan future activity.
Today forecasts are created by meteorological models, that are based on modeling physical processes. For temperature forecast in Slovakia, meteorological model Aladin is used by Slovak hydrometeorological institute (SHMI). Aladin describes weather using partial differential equations which are solved by numerical methods [8, 17].
The goal of this thesis is to improve temperature forecasts created by Aladin using data supplied by SHMI. Our thesis is a continuation of diploma thesis written by Michaj Hajdin in 2016 called *Numerical weather prediction postprocessing.* To improve Aladin predictions, Hajdin used linear regression and a rolling window approach. In our thesis, we continue in similar approach and introduce further improvement.
Additionally, we focus on periods of year when temperature is stable. When using deterministic models and the method of rolling window, models tend to create similar errors when temperature is stable for longer period. To deal with this problem, we experiment with models that include their previous errors in its features and discuss the obtained results.
We first present terminology and methods that were used in our experiments. Later we describe methods used in previous theses, articles and papers related to temperature postprocessing and forecasting. Further we present analysis of data we worked with, various models and results gained with different learning algorithms, and models and results for stable weather periods. Finally, we evaluate our best model for all the stations, present the results and compare them to the Hajdin's thesis.
All the source files that we used are attached to the thesis and can also be found on github *https://github.com/rizip1/diplomovaPraca.* We used python programming language with statistics and machine learning libraries as scikit-learn, keras, numpy, pandas and others.
The dataset that we used for analysis and experiments is not attached to the thesis because it is under a non-disclosure agreement.

Chapter 1

Basic terms

In this chapter we briefly define learning algorithms, terms and approaches that we use later in our thesis. Definitions of the learning algorithms are simplified to mainly remind the reader of those methods and refer to further literature. We expect that the reader has basic knowledge of statistics or machine learning.

Linear regression and rolling window approach

Linear regression is an approach in which dependent variable y is modeled as a linear combination of independent variables [26] $x_1, ..., x_n$ such that

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n + \epsilon$$

where ϵ is an error term and $\theta_0, ..., \theta_n$ are coefficients or weights that the model needs to estimate. In machine learning literature, it is more convenient not to explicitly include an error term in the models definitions, therefore we omit it from our future models. We refer to dependent variables as to *features*, and every feature corresponds to one data property. For m training examples all consisting from n features, we store the independent variables in so called *design matrix* X [16], as showed in Equation 1.1.

$$X = \begin{pmatrix} 1 & x_1^1 & ... & ...x_n^1 \\ 1 & x_1^2... & ... & x_n^2 \\ ... & ... & ... & \\ 1 & x_1^m... & ... & x_n^m \end{pmatrix} \quad \theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ ... \\ \theta_n \end{pmatrix} \quad y = \begin{pmatrix} y^1 \\ y^2 \\ ... \\ y^m \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon^1 \\ \epsilon^2 \\ ... \\ \epsilon^m \end{pmatrix} \tag{1.1}$$

There is 1 added to each row of the design matrix, often called *bias* term [16], so that a model takes into account, that data are shifted from the origin. The goal is to estimate vector of coefficients θ, such that the sum of squared error terms, for equation $y = X\theta + \epsilon$, is the lowest possible. This method is called *ordinary least squares (OLS)* [26] and can be computed analytically using equation $\theta = (X^t X)^{-1} X^t y$ [26], often called as *Normal equations* [20]. For larger datasets gradient descent algorithm might be more appropriate [20]. Estimated coefficients can be used to make predictions for new, previously unseen, data. From statistical point of view, OLS requires further assumptions so that the estimate is interpreted correctly [26]. In our thesis we ignore those assumptions and use OLS more from machine learning point of view, thus focus mostly on minimizing error.

In our thesis we use *rolling window* [23] approach. Suppose that for a given time series, the rolling window length is a constant l. The idea is that when we are at time t and want to predict value in next timestep $t + 1$, we estimate model coefficients using a training set consisting from data records from times between $t - l + 1$ and t. In next timestep the rolling window is shifted by one so that the test record from time $t + 1$ is included in the training set and the record from time $t - l + 1$ is removed. We continue while the whole timeseries is not processed.
Consider a time series consisting from values a_1, a_2, ..., a_{10}, timestep t equal to 7 and rolling window length l equal to 3. In that case a training set would consist from values a_7, a_6, a_5, and we would predict value of a_8. In the next time step the training set would consist from a_8, a_7, a_6, and we would predict value of a_9.

Regularized regression

Lasso and *ridge regression* are both examples of regularized regression. Goodfellow et al. [16] define regularization as:

> Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Often regularization is used in cases when overfitting is observed. Overfitting occurs when validation error is significantly higher than training error [16]. Opposite to overfitting, underfitting occurs in a case when a model is even not able to fit training data [2]. In cases when data are i.i.d. we usually have a single training set and we can compute training error at once. In case of the rolling window approach training error can not be computed at once because values are being predicted in a sequence as the rolling window is shifted. In such case it is possible to compute mean train error as an average train error across all rolling window positions and compare it with mean error

for predicted values.
Regularization in ridge and lasso regression is achieved by a penalty aimed for magnitude of estimated model parameters. In ridge regression we minimize the objective

$$\| X\theta - y \|_2^2 + \alpha \| \theta \|_2^2$$

so we use L2 norm of vector of coefficients as the penalty for high coefficients values. In lasso regression the objective is

$$\frac{1}{2n} \| X\theta - y \|_2^2 + \alpha \| \theta \|_1$$

and L1 norm is used instead. In both the cases α stands for a regularization term which controls an amount of penalty applied to θ. The higher value of α the more regularization is added. The higher regularization the more we force a learning algorithm to choose coefficients with lower magnitudes. The n in the lasso equation is the number of samples in a training set.
When we use both ridge and lasso regularization terms in one model, thus regularize using both L1 and L2 norms, we get so called *elastic net* model [2].

SVR

Support vector machines (SVM) is a supervised learning algorithm designed for classification, that maps original data into feature space using a nonlinear function known as a *kernel* [28]. The algorithm constructs a hyperplane that divides two groups of categorical data. Robustness of SVM is that it does not construct arbitrary hyperplane that separates the groups, but it chooses the hyperplane that holds the largest possible margin between the groups of data [21]. For multiclass classification, e.g. one-versus-all technique can be used [5].
A decision function for SVM is fully defined by data points that define the boundary, also known as support vectors [4,21]. As a kernel any function that satisfies the Mercer theorem can be used [4,21]. Usually a radial basis function (RBF) kernel or a polynomial kernel is used [9]. If data were not transformed by any kernel, we say that *linear kernel* was used.
SVM can also be used for regression problems. In that case we refer to *Support vector regression (SVR)*. Instead of constructing a hyperplane, that maximises a margin between different categories of data points, it constructs a hyperplane that lies the closest to data as possible [10]. SVR creates so called regression tube [4,10], which

radius is determined by a parameter of tolerance ε. Distances within the regression tube are ignored. For SVR, support vectors are points, that define the regression tube boundaries. The detailed explanation of how model parameters are estimated is out of scope of our thesis. Additional information can be found in Smola and Schölkopf article *A Tutorial on Support Vector Regression* [4].

MLP

Multilayer perceptron (MLP) is a type of a feedforward neural network. Same as linear regression it learns a function to map dependent variables to independent variables, but the function can be more complex usually using a lot of nonlinearity. In the most frequent case, MLP can be represented as an oriented acyclic graph containing three or more layers, while every layer consists of vertices called neurons [16]. Every neuron from a previous layer is connected to all neurons in a next layer. Each neuron in a hidden layer maps values from a previous layer using weighted linear summation and applying a non-linear activation function $g(\cdot) : R \to R$ [2]. As such function usually sigmoid, tanh or relu is used [16]. First layer is called an input layer and contains one neuron for every feature plus a neuron for the bias term. Last layer is referred to as an output layer and its activation usually differs from activations in hidden layers. In case of regression, an identity function in the output layer is used [2, 16]. Other layers between the input and output layer are called hidden, and there can be arbitrary number of them. MLP can be trained with a backpropagation algorithm [16] which propagates error back after a fixed number of processed inputs. The network knowledge is stored in its weights which get updated during the training. Also other algorithms like Adam or Newton's method can be used to train MLP [2, 16].

LSTM and GRU

In the algorithms we discussed so far every input is mapped to its corresponding output, but without considering any time dependencies. *Recurrent neural networks (RNN)* were directly designed for learning sequences, allowing information to flow through time as a network is learning. RNNs can be viewed as networks with loops. The very simple model of RNN is Elman network shown in Figure 1.1. Compared to MLP, a hidden layer in Elman network is calculated from both input and context layer. The context layer is just a copy of the hidden layer from a previous timestep. Past knowledge is therefore stored in a weight matrix between the context and the hidden layer. For training backpropagation through time or truncated version of it can be used [18].

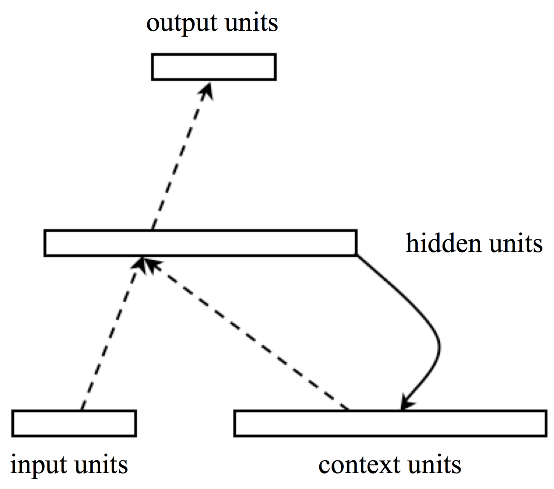


Figure 1.1: Example of Elman network [6].

However, Elman's networks are difficult to train for longer sequences as they suffer e.g. from a vanishing gradient problem [18]. *Long short-term memory (LSTM)* networks introduced by Hochreiter and Schmidhuber solve some of those problems [18]. The difference in LSTM compared to Elman network is how the hidden layer is computed. LSTM introduce a concept of cells. The cell can be understood as a function that outputs the hidden layer based on the previous hidden layer and a given input. In every timestep t the cell keeps it internal state C which can be mutated by gates. Gates allow the cell to learn which information to forget, add and output in each time step. LSTM has three gates, forget gate, input gate and output gate [3]. Gates are in fact sigmoid layers. An example of the LSTM cell is shown in Figure 1.2 [3] where the cell state C flows through the upper arrow. The hidden layer is h, the input is x, σ is the sigmoid activation function and $+$ and $*$ are the point wise operators for multiplication and addition. The detailed explanation of how the cell state is being computed is out of scope of this thesis and can be found in the original article [18]. After the cell state is updated, the next hidden state is computed using the output gate. A mapping from hidden to output layer is performed same as for Elman network. In the LSTM terminology, the number of neurons in the hidden layer is often referred to as the number of units. As MLP can consist of multiple hidden layers, also LSTM layers can be stacked one after another. The input to the ith LSTM layer, where i is greater than one, is the previous hidden representation from $i - 1$th layer and the previous representation from the current LSTM layer.

In the 2014, a concept of *Gated recurrent units (GRU)* was introduced. GRU is similar to LSTM, but the difference is in a gate structure which is simplified. The gate state is removed and a forget and an input gate is combined into a single update gate [11]. The fact that two gates are merged into one makes GRU more computationally efficient compared to LSTM.

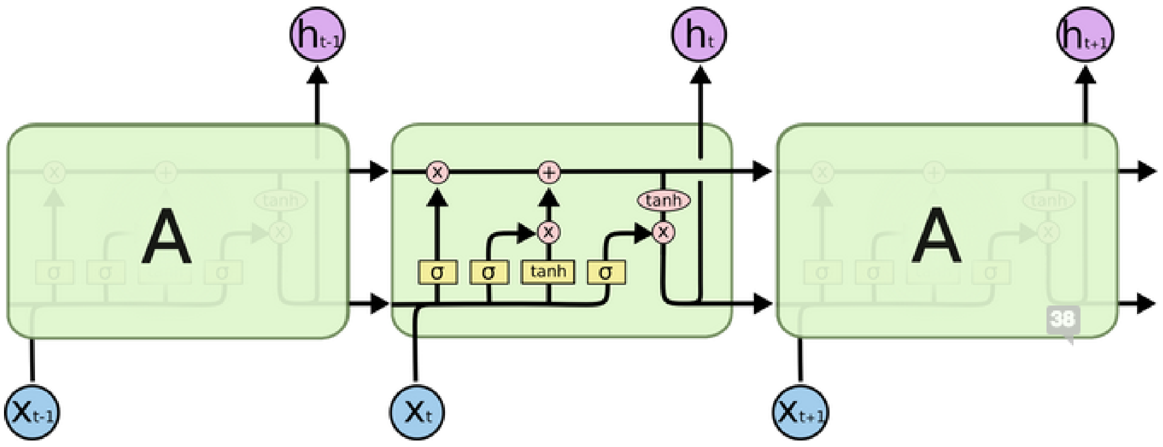


Figure 1.2: Example of LSTM cell [3].

ARIMA

We consider it important to at least mention a standard time series modeling approach called *ARIMA*, even though we do not use it in our thesis for the reasons presented below. ARMA process is a shortcut for Autoregressive and Moving average processes. Autoregressive model estimates a next time series value from its p previous values ($AR(p)$). Moving average model estimates a next time series value from its q previous errors ($MA(q)$). When $AR(p)$ and $MA(q)$ are used together they form $ARMA(p, q)$ model which can be written as

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

where c is a constant, θ and φ are vectors of parameters that the model estimates, y_t is a value of time series y at time t and ε_t is an error term at time t.
$ARIMA(p, d, q)$ is an extension of $ARMA(p, q)$ model. The I in ARIMA stands for integrated what means that the model is capable of creating a stationary time series from a non stationary. The stationarity is achieved by differencing [25] when values of y at time t (y_t) are replaced by values $y_t - y_{t-1}$ for all t. The value d represents an order of differencing, thus the number of times the differencing is applied.
For modeling seasonality in data seasonal $ARIMA(p, d, q)(P, D, Q)_s$ can be used where P, D, Q are orders of seasonality for $AR(p)$, differencing and $MA(q)$ respectively. First nonseasonal and then seasonal differencing is applied.
It can be difficult to decide appropriate orders of ARIMA models and the problem is even bigger for multivariate ARIMA-like models [23,29]. In our case each station forms different time series and probably requires different orders.
As we mentioned earlier, OLS or SVR are not directly capable of handling time series data. However, using those techniques it can be easier to achieve higher accuracy when compared to standard ARIMA models [27] [29], for example using the rolling window approach.
Moreover, those models usually need much longer sequences to be trained on [27]. Also in [29] Nardi and Rinaldo described the theoretical properties of lasso technique, establishing various types of consistency, when applied to autoregressive time series. Due to those facts we rather experimented with recurrent neural networks instead of the standard time series models.

Feature scaling

When using linear regression, magnitude of features does not matter when coefficients are estimated using Normal equations [20]. It matters when using gradient descent so it can converge faster [20]. However, for small datasets using analytical solution does not cost much time. When using regularization, SVR or neural networks the magnitude of features matters [20]. When features have different scales regularization has different effect based on features scales.
One scaling method we use is called *standardization*. Each feature is centered by subtracting its mean from itself and divided by its standard deviation as shown in Equation 1.2.

$$x_{scaled} = (x - x_{mean})/x_{std} \tag{1.2}$$

The second scaling method we use is called *min-max normalization*. Features are transformed into interval [a,b] using their corresponding minimum and maximum values as shown in Equation 1.3. For min-max normalization we scale by default to a range [0,1].

$$\begin{aligned} x_{tmp} &= (x - x_{min})/(x_{max} - x_{min}) \\ x_{scaled} &= x_{tmp} * (b - a) + a \end{aligned} \tag{1.3}$$

When applying feature scaling minimum/maximum or mean/std values are picked based on a training set and are also used to perform scaling for a test set.

Error metrics

In our thesis, we evaluated our models using two error metrics. Those are *mean absolute error (MAE)* and *mean squared error (MSE)* defined in Equations 1.4 and 1.5. In those equations, $h(x_i)$ is the value that a model predicts for the record x_i and m is the total number of records. Those metrics were chosen to easier compare with the results from the Hajdin's thesis. In all the results we refer to, both MAE and MSE are measured in degree Celsius ℃.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|h(x_i) - y_i| \tag{1.4}$$

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2 \tag{1.5}$$

Grid search and cross validation

The problem with regularized regression or neural networks is that hyperparameters (regularization parameter, number of neurons, ...) must be chosen before the learning happens. One possible strategy is to try various values and pick the best. The problem with this approach is that each station data forms different time series and picking those values solely based on results for one station might not generalize well.
The second approach to choose hyperparameters is to use a *grid search* for time series. Grid search is a method that is used to pick hyperparameters while trying all possible values from a fixed set. It usually evaluates those hyperparameters using cross-validation [2, 16]. The *cross-validation* for time series data differs from the methods that can be used for i.i.d. data. The strategy we used is that we cut a training set into equal buckets. In the beginning, we use the first bucket as the training data and evaluate a model using the remaining buckets. Then we use the first and second bucket as the training data and again evaluate on the remaining buckets. In each step we add one more bucket to the training set and in the last step we use $n - 1$ buckets as the training set and evaluate a model on the last bucket. The number of buckets is n. For most models we used n equal to 5 and used the scikit-learn implementation of grid search using time series cross-validation [2].

Wilcoxon paired test

To test whether improvement in MAE or MSE gained with some model compared to another is significant we later use *Wilcoxon signed-rank test* [19] with 95% confidence level, also known as *Wilcoxon paired test* in R statistics software. The test compares if means of two dependent samples differ and is appropriate in situations when samples do not follow normal distribution [19]. Absolute and squared errors of the models we use later, do not follow normal distribution what can be confirmed by Kolmogorov-Smirnov test [19]. Therefore, we do not use Welch t-test which requires samples to be normally distributed [19].
Paired version of the test was used because we assume that the absolute and squared errors are not independent. It is because the errors of two compared models can be viewed as *before* and *after* values with relation to a given time. Figure 1.3 shows absolute errors dependence for two compared models.

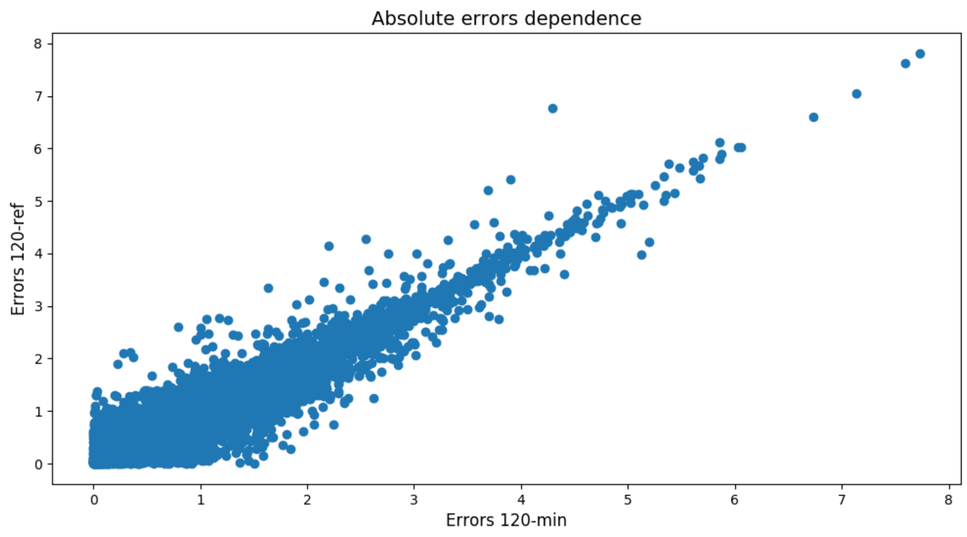


Figure 1.3: Absolute errors dependence between the 120-ref and 120-min models. The models are defined in Chapter 4.

When testing improvement significance, we test two null hypotheses. First states that absolute errors of tested models have equal means. In the second hypothesis we test for equal means of squared errors of the tested models. If the p-value for the null hypothesis is less than 5% we deny it and trust the alternative hypothesis.

Chapter 2

Related work

This chapter describes theses, articles and papers that are related to the problem of improving temperature forecasts or forecasting next values based solely on past observations.

Numerical weather prediction postprocessing

In the thesis written by Michaj Hajdin [17] the author showed that Aladin forecasts are with high probability systematically biased using t-test. Systematic bias means that Aladin creates similar errors for similar conditions, making additional postprocessing reasonable.
Data were collected by SHMI and contained records about observations and forecasts for 37 hydrometeorological stations. The observations contained temperature, humidity, pressure, wind direction, wind speed and rainfall records for every hour of a day from 1 January 2013 to 20 December 2015. SHMI creates forecasts at 0AM and 12AM for 72 hours ahead. Those forecasts were also included in the data.
To avoid overlapping periods, the author focused on forecasting temperature for 12 hours ahead. Predictions were obtained using a linear regression model combined with the rolling window approach. When predicting temperature at time t, a training set was constructed using records at times $t_{-12}, t_{-24}, ..., t_{-(12*m)}$ when m stands for the length of the rolling window.
Results of models created from various combinations of features were presented in the thesis. The author first started with a model that could be written as

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h \qquad h \in 1, ..., 12 \tag{2.1}$$

where i is the time when Aladin made a forecast, y_i^h is measured temperature at time $i+h$, T_i is measured temperature at time i and P_i^h is temperature forecasted by Aladin for h hours ahead starting from time i.
The author used $m = 60$ as the length of the rolling window because he stated that with the growing m the accuracy did not improve and was even worse. To pick a final model the author used mean absolute and mean squared error metrics using single station data which location was near Bratislava airport. The features that were picked for the final model were measured temperature, temperature forecasted by Aladin, wind direction and humidity. Equation 2.2 describes that model

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 V_i + \theta_4 D_i \qquad h \in 1, ..., 12 \tag{2.2}$$

where V_i, D_i are humidity and wind direction at time i. Other parameters are the same as in Equation 2.1. In the later text we refer to this model as to the *hajdin-best*. To make newer prediction more important than the older, each observation was assigned a weight. More precisely each data record was multiplied by a weight function $w(i) = \sqrt{a^i}$, where i was ith newest record in the rolling window and $a \in (0, 1)$. Reasonable values of a are between 0.95 and 0.99. The author claimed $a = 0.97$ to have best results.
The author also considered so called autocorrection models that should learn from their previous errors. The first autocorrection model learnt from its error, that it made 24 hours before the time for which the prediction was made. The second autocorrection model learnt from temperature predicted by itself and measured temperature, both for 24 hours before the time for which the prediction was made. Also other features from the hajdin-best model were added. The autocorrection models improved neither MAE nor MSE metrics however the author claimed that the first mentioned model had better results compared to the second one. Finally, the author created combined model by averaging the predictions for the hajdin-best model and the second autocorrection model. The final MAE and MSE values for the station near Bratislava airport were 0.924 and 1.4898 respectively. It is not clear why the author did not use the first autocorrection model for averaging, as it had better results in both MAE and MSE. Later the combined model was evaluated for other stations. Across all the stations the combined model gained improvement in MAE characteristic from 13% up to 57%.
The author also tried adding and transforming features, like adding the third root of measured temperature, lag of measured temperature for 1 or 24 hours and decomposition of wind direction into two features using *cos* and *sin*. According to the results, none of those features decreased models errors.

Temperature postprocessing of ensemble forecasts

Improving temperature forecasts using statistical postprocessing data was also described in Feldmann's diploma thesis *Statistical Postprocessing of Ensemble Forecasts for Temperature: The Importance of Spatial Modeling* and similar approaches were introduced in Feldmann's et al. article called *Spatial Postprocessing of Ensemble Forecasts for Temperature Using Nonhomogeneous Gaussian Regression.*
Current state of the art meteorological forecasting models are based on ensemble strategy [14]. Rather than creating one expert model, multiple different models are created. After all the models make their forecasts, those values are merged to create a final forecast. For example an average of ensemble values can be used to get the final value. The meteorological ensemble models still suffer from bias [14], so additional postprocessing is needed.
In the Feldmann's thesis, the prediction system used for forecasting was COSMO-DE-EPS which is a 20 member ensemble system, created by German Meteorological Service [14]. Forecasts were created for 21 hours ahead for surface temperature in Germany. Data in the Feldmann's thesis consisted from the forecasts for each of the ensembles. This differs from our thesis, as we had only data about final Aladin forecasts and had no additional information about individual ensembles, the final forecasts were created from.
The author used the rolling window approach for constructing a training set which length was 25 days for each rolling window position. The length of the rolling window was not longer according to the size of the dataset which contained data from 10 December 2010 until 30 November 2011. Therefore, the predictions were made for interval between 5 January 2011 until 30 November 2011. The author does not specify if the training set consisted from all records for last 25 days or if only records for specific hours were picked. The author used different statistical models as BMA (Bayesian model averaging) or EMOS. BMA and other models that the author used are statistical approaches for combining competing statistical models [14].
The only used features were measured temperature and temperature forecasted by the ensembles. No observations about wind or humidity were used. When simplified the model could be written as

$$y_s = f(y_{1s}, y_{2s}, ...y_{20_s})$$

where the y_s is temperature of interest at location s. The y_{n_s} is nth ensemble prediction for location s. Using EMOS the author improved MAE from 1.57 to 1.43 and MSE from 2.27 to 1.83.
Spatial model, for modeling dependencies between locations, was also considered.

When simplified the goal was to estimate vector Y given F_m where

$$Y = y_s : s \in S$$

$$F_m = f_{ms} : s \in S$$

and S is a set of locations, y_s is a temperature of interest at location s and f_{ms} stands for forecast of mth ensemble for location s. Spatial model was used to predict minimum temperature along highway A3 in Germany using nearby stations. The improvement for MAE was from 1.92 to 1.21 and for MSE from 2.33 to 1.55.
We can hardly compare the Hajdin and Feldmann results. Both the approaches learns from different data and the ensemble information was available neither in the Hajdin's and nor in our thesis. The methods used in the Feldmann's thesis are suitable for situations when the information about all ensembles is available. Taking into consideration the fact that also Feldmann used the rolling window approach, we decided to use it later in our experiments. We consider spatial modeling interesting however we did not incorporate it into our experiments.

Learning only from observations

After the introduction of postprocessing applied to meteorological forecasts, we focus on approaches where temperature is being predicted only from previous observations. Even though this is not our case, as forecasted values are inarguably our very important predictor as shown in Section 3.2, we were inspired by usage of different learning algorithms and data features. The articles introduced later use more complex learning algorithms compared to the previous statistical methods. We introduce approaches that use support vector regression, multilayer perceptron and ARIMA models.

SVR and MLP approaches

Article published by Chevalier et al. called *Support vector regression with reduced training sets for air temperature prediction: a comparison with artificial neural networks* [10] deals with short-time temperature forecast for 1 to 12 hours ahead. The authors followed on the article published by Smith et al. using data between years 1997 and 2005. The data were collected from 75 stations in state of Georgia using AEMN model that was established at the College of Agricultural and Environmental Sciences of the University of Georgia [15]. Smith et al. created a general MLP model for whole year period and a MLP model that only focused on the winter months. Temperature was predicted for 1 to 12 hours ahead. Instead of using the rolling window, the usual traning-validation-test approach was used. The training set consisted from records between years 1997-2000 (approx. 1.25 million records), the validation set from records between years 2000-2003 (approx. 1.25 records) and the test set consisted from records between 2004-2005 (approx. 800 000 records). The validation set was used to tune the models hyperparameters. Accuracy was evaluated on the test data which the models did not seen before.

Input features for MLP were temperature, solar radiation, wind speed, rainfall and humidity. Each of those features was lagged by 1 up to 24 hours. By *lag* by i hours we mean the feature value at time $t-i$. Therefore, for the feature x_t we also added features $x_{t-1}, ..., x_{t-24}$. This caused an increase from 5 features to 125. Also differences between subsequent features were added. For feature x_t we also added $x_t - x_{t-1}, ..., x_{t-24} - x_{t-25}$. The features count was therefore doubled resulting in 250 features. The authors also used *dummy variables* which are features whose values can only be zeros or ones. Four dummy variables representing time of a day and four dummy variables for seasons were used. Only the dummy variable corresponding to a current season or a current time of a day, for a given record, was set to one. Therefore, the total features count reached 258.

Neural networks are sensitive for appropriate feature scaling [24], therefore the authors scaled all the features into a range 0.1 to 0.9.

The Chevalier's goal was to create SVR model and compare its performance to the mentioned MLP. SVR should be less prone to overfitting compared to MLP [10] and it always finds a global minimum [10]. Global minimum property does not hold true for MLP as it usually only converges to a local minimum [16]. Disadvantage of SVR might be higher computational time. The dataset and the features that Chevalier's used were same as for the Smith's MLP.

Due to the fact that training of SVR needed more time that MLP, Chevalier's used reduced training sets for parameters tuning. As the training time for SVR grows with decrease in regression tube ε [10], the authors decided to choose larger value of ε for hyperparameters tunnig. The hyperparameters that needed to be tuned were penalty factor C and the kernel width γ as the RBF kernel was used. For parameter tuning reduced training sets of 15,000 randomly sampled patterns were used. The samples were selected with replacement. While tuning the hyperparameters, ε was set to 0.05. Each trained model was evaluated on the validation set. After values for C and γ were found, the value of ε was selected different for each prediction step (1-12 hours). It was again selected using randomly sampled training sets, trying values from 0.003 to 0.01 with a step size of 0.001. The final SVR model and the MLP were trained on both training and validation sets.

Average error for SVR for winter data was 0.514°C when predicting temperature for 1 hour ahead and 2.303°C for 12 hours ahead. Using all the data and SVR the corresponding values were 0.513°C, 1.922°C respectively. The results show that forecasting winter values might be more difficult. In comparison with MLP, the SVR had better results for winter months. The difference at MAE was 1,108% and the SVR was also better at forecasting temperature for each hour.

In Y.Radhika and M.Shashi article called *Atmospheric Temperature Prediction using Support Vector Machines* [28] the average temperature for next day was forecasted. Data were supplied by University of Cambridge for period from 2003 to 2008. Train and validation sets consisted from records between year 2003 and 2007 and test set from the remaining. The only features used were average temperatures from previous days. Competing algorithms that the authors used were MLP and SVR. The MLP consisted from one hidden layer with $2i + 1$ sigmoidal perceptrons where i was the number of input layer units. Backpropagation algorithm was used for weight optimization. The data for SVR were scaled using standardization. For MLP min-max scaling into [0, 1] interval was used. The authors stated that the polynomial kernel had worse results compared to RBF kernel. The kernel parameters were found using grid search. Tried features were average temperatures for 2 up to 10 days before. As the final features, the average temperatures for last 5 days were used. MSE for MLP was 8.07 and 7.15 for SVR.

ARIMA approach

A seasonal ARIMA model was used for temperature forecasting in the paper published by Nury et al. called *Time Series Analysis and Forecasting of Temperatures in the Sylhet Division of Bangladesh* [7]. The seasonal ARIMA model was used due to the seasonality in climatic data. Data consisted from records about minimum and maximum temperature measured for months starting from year 1977 to 2012. Subsequent values were later forecasted. The seasonality value s was set to 12, corresponding to one year seasonality. The value for seasonal differencing was set to one. To determine ARIMA parameters the authors used Box-Jenkins method and AIC criterion [25]. As the final model the ARIMA$(1, 1, 1)(1, 1, 1)_{12}$ was chosen, but the authors did not supply specific error metrics and results in the paper.

We presented few approaches for temperature postprocessing or forecasting. It is difficult to present state-of-the-art technique for postprocessing of temperature due to the fact, that known experiments are created from different datasets which are not always public. In our thesis we therefore focused directly on data granted by SHMI and compared our results with the Hajdin's thesis.

Chapter 3

Data processing

In this chapter we describe data that we worked with, their processing and methods that we used for missing and invalid values exploration. We also present estimated correlations among features and features importances. In the later text when we refer to the *reference* station, we mean the station that Hajdin used for model construction, located near Bratislava airport with ID 11816.

Dataset description

Dataset that we used in our thesis contained same attributes as in the Hajdin's thesis. Available observation attributes were:

- temperature
- humidity
- pressure
- rainfall during last hour
- wind direction
- wind speed

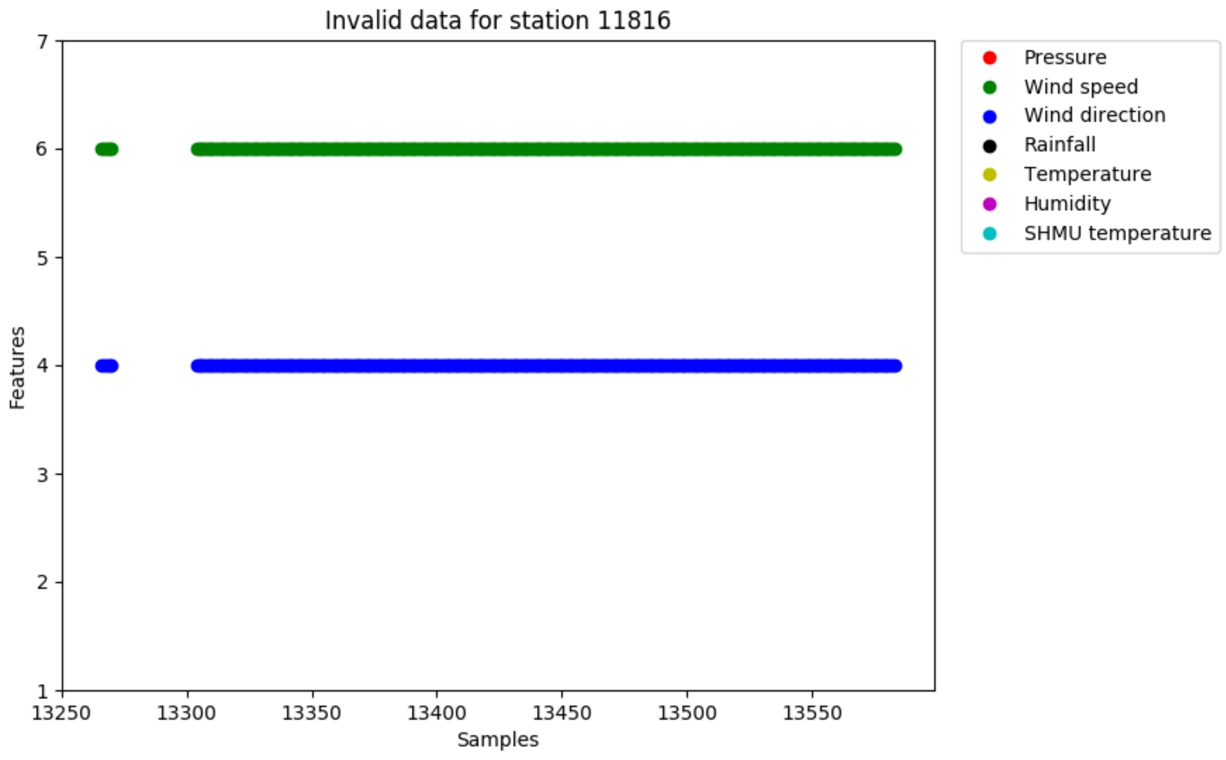
We obtained data for stations in a single csv file which contained records for 37 stations. The file contained station ID, gps coordinates and a location name. Observations were obtained in 36 csv files (one for each station) and forecasts in 2167 csv files. Every row in a forecasts file contained ID of the station it belonged to, its validity time and forecasted values for various weather properties. The dates and times when the forecasts were created were included in the files names. *Validity time* is the time for

which forecast was created and by *reference time* we mean the time when forecast was created.
We stored the data in a postgresql database in three tables called *forecasts*, *observations* and *weather_ stations*. To feed the database with the data we used python scripts. For merging observations and forecasts data into single csv files, one for each station, we used *data_ analysis* script. Since, for two stations there were missing observations resp. forecasts, we ended up with data for 35 stations. For storing the data in the database we used scripts *load_forecasts*, *load_ stations* and *load_ observations*.
The difference to the Hajdin's thesis [17] is that we had more data records for each station, including all the data that were used in the Hajdin's thesis. The summary of total amount of data, that we had for each station, can be found in Table 3.1.

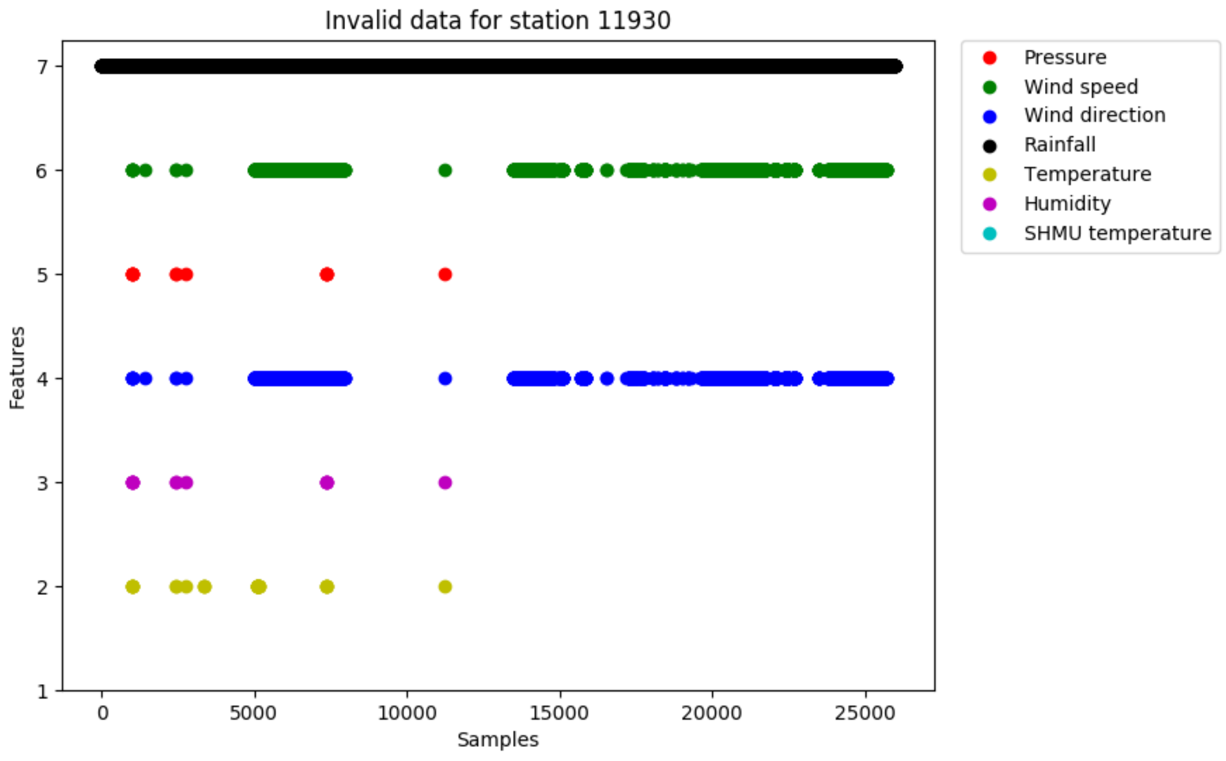
Missing and invalid values analysis

Before combining existing features into various models, we decided to first investigate invalid and missing data values. By data with invalid values we mean measurements that had values of -999. By missing data we mean whole records of observations that were missing for a specific hour. Predictions that learn from data with invalid or missing values can be skipped or replaced by an average of other values in case, that missing or invalid intervals are short.
We created separate plot for each station where the x-axis corresponds to timestep and y-axis contains values between 1 and 7, one for each feature. The timestep is represented as a distance in hours from an initial measurement. For measurements with invalid value at timestep t we drew a circle with x-axis set to t and y-axis set to its matching feature value (1 to 7). Each feature is represented by a different color and a legend is shown in a top left corner. The examples of those plots can be seen in Figures 3.1a and 3.1b.
After investigating generated plots we realised that pressure was invalid for 8 stations in more than half of the measurements. For rainfall same situation occurred for 5 stations. For other features there were not invalid values in more than half of the measurements. Nevertheless, wind direction was almost uniformly invalid across 18 stations forcing models with this feature to skip a lot of predictions. Wind speed was also invalid for many stations, but the intervals with no invalid data were longer compared the to wind direction. There were almost no invalid values for stations with IDs 11894 and 11938. However, for station with ID 11894 all the values for rainfall were equal to zeros, and also many records were missing as shown later. We therefore suppose that those rainfall values were invalid, but did not have values of -999. Also for other stations, rainfall values were often equal to zeros, and we could not determine

whether they were correct values or the errors in the measurements occurred.
Due to those results we considered rainfall and wind direction as the most problematic predictors. In order to use them in our models, we would have to throw away a lot of observations. More precisely, we would have to skip all the observations where any of values in a model was invalid. This is getting even worse when we consider growing length of the rolling window. For temperature and humidity there were occasionally invalid values, so models based only on those features, and features derived from them, would not have to skip many predictions. There were no invalid values for temperature that was predicted by Aladin. Table 3.1 contains invalid values for all the stations.
When we searched for missing data we compared a distance d in hours of validity times for two consecutive records. When d was greater than one, it denoted that $d-1$ records were missing. For the reference station and records until 2015-01-01 there were two missing records. Let t denote timestep for which missing record occurred. We replaced this record as an average of records in timestep $t-24$ and $t+24$. If some feature in any of these two records had value of -999, the resulting feature in new record was set to -999. It could happen, that a record at time $t-24$ or $t+24$ was also missing. In that case we created new record with all values except reference and validity times set to -999. We performed this replacement only if a gap of missing values was less or equal to 24 hours. Longer gaps are more difficult to replace, so for gaps longer than 24 hours new records features were all set to -999 except for the time values.
After adding the missing values, we performed replacement of invalid values using the same strategy. If a feature at time t had invalid value, it was replaced by an average of that feature values at times $t-24$ and $t+24$, in case that those values existed and none of them was invalid. Otherwise, no replacement was performed.
Adding missing values was more important that replacing invalid values, because missing records can cause, that a model learns from data related to other hour than expected. Invalid values only cause skipping of the predictions what we consider less significant problem.
The replacement strategy for the missing and invalid values could be more comprehensive e.g. we might choose a different strategy for different kinds of features. Due to the fact that in the later models we mostly use temperature, which is invalid in the lowest amount of cases, we consider this approach sufficient.
The replacement was performed using *data_ analysis* script. For the station with ID 11894 there were too many missing values, so we decided not to use this station for later calculations. Only for station with ID 11963 there were no missing values, but a lot of invalid values for pressure was present. The amount of missing records for each station can be found in Table 3.1.



(a) Invalid data for the reference station until 2015-01-01.



(b) Invalid data for the station with ID 11930 using all data.

Figure 3.1: Invalid data

Table 3.1: Missing and invalid measurements. *Station* denotes station id, *Count* denotes total count of observations, *Missing* denotes total number of missing observations and other columns denotes number of invalid rows for a specific feature.

Station	*Count*	*Missing*	*Hum.*	*Pressure*	*Rain*	*Temp.*	*Wind dir.*	*Wind sp.*
11801	25809	183	12	12	25809	12	99	99
11803	18803	1905	3710	18803	0	1345	2255	74
11805	16986	1033	15	16986	0	0	2985	106
11812	25643	349	3125	0	25643	0	8731	8731
11813	25968	24	0	85	5	0	0	0
11816	25973	19	0	0	0	0	291	291
11819	25738	254	0	0	0	0	1383	0
11826	22866	200	0	0	0	0	398	65
11855	25598	394	0	0	0	0	237	9
11856	25801	191	0	2453	0	0	0	0
11857	21337	18	0	21337	0	0	5573	92
11858	25911	81	0	1	0	0	24	24
11867	25425	567	1	110	0	1	449	6
11878	21335	43	6	21335	0	0	6008	384
11880	25763	229	76	0	0	0	1439	0
11894	13183	12809	0	0	0	0	1	1
11900	25780	212	0	0	0	0	210	0
11903	25389	604	0	0	0	0	1569	692
11916	24888	1104	966	749	24888	749	2310	2309
11917	13616	4884	2	13616	0	2	5283	326
11918	25881	111	17	0	0	0	0	0
11919	20506	2920	272	20506	20506	273	2273	924
11927	25764	228	3	0	0	0	1248	0
11930	25967	25	13	13	25967	36	8488	8488
11933	25263	729	93	12	0	3	23	5
11934	16227	5443	174	0	0	0	279	24
11938	25941	51	0	0	0	1	0	0
11952	25960	32	3	3	2	3	191	3
11958	25247	745	2030	4399	0	1970	189	196
11962	20316	871	1	20316	0	1	1710	155
11963	13566	0	0	13566	0	0	642	5
11968	25730	262	124	1	0	110	547	147
11976	25802	190	1	1	0	1	319	1
11978	25827	166	28	29	15	97	26	26
11993	25752	240	0	3	0	0	810	0

Correlations and features importances

In the Hajdin's thesis the best chosen model consisted of temperature, humidity and wind direction. However, adding other features than temperature caused very little improvement in MAE (only about 0.1 for the reference station). It might imply that other than temperature features (measured temperature and temperature predicted by Aladin) are poor predictors that have very little impact on improvement of the model. That improvement also comes with a cost of more skipped predictions as other features are invalid in much more cases compared to temperature as shown in Section 3.2.
To test the hypothesis that other than temperature features are little important we trained extra trees ensemble model, with 500 trees, using all features we had. Extra trees algorithm creates multiple decision trees and each tree is grown on a random sample of data [2]. In the splitting time each tree chooses random candidate features to split upon and also generates a random decision threshold for each feature [2]. Then it picks the best suitable threshold as the splitting rule [2]. After all trees are trained it uses simple averaging [2].
We used values that were available at the reference time for all the features to avoid learning from future values. The ensemble model was trained using the first 13000 records for the reference station because there were no missing and invalid values for this interval. Extra trees algorithm can be used for getting features importances after it is trained. According to the scikit-learn documentation [2]

> The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance.

Figure 3.2 shows the corresponding feature importances. We can see that the importance is highest for temperature predicted by Aladin and for temperature measured at prediction time. Humidity seems significantly less significant, and other features were assigned almost no importances. Vertical lines for each bar shows inter-trees variability. Note that the training that was used varies from the one that we use later. In this case, the model was trained for all hours at once.
We did not investigate features importances for other stations using extra trees algorithm. The purpose was to get initial idea of the feature importances and due to the extreme differences between temperature and non temperature features, we consider this sufficient. Also because of only minor improvements when using other than temperature features in the Hajdin's thesis and a huge amount of invalid records for those features, we do not use other than temperature features in the later experiments except humidity. The reason we gave humidity a try, is that its feature importance was a little

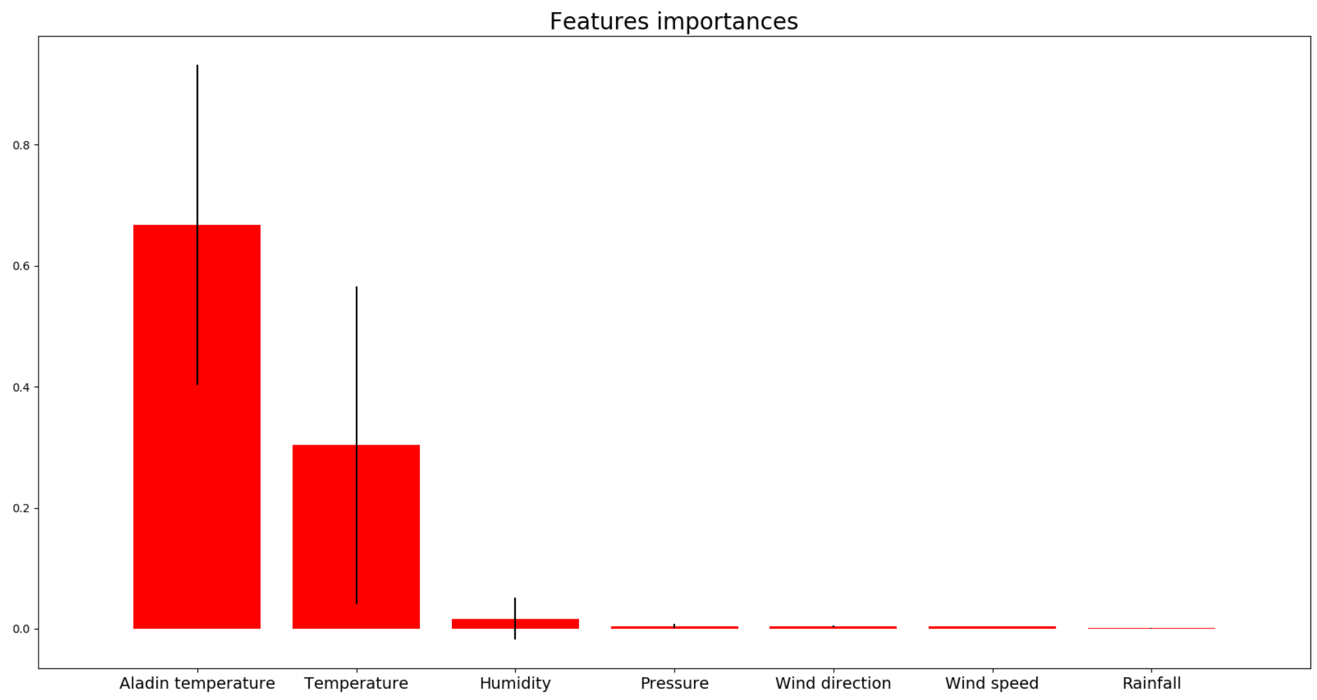


Figure 3.2: Features importances using first 13000 records for the reference station.

higher compared to other features. It was also incorporated in the hajdin-best model, and the amount of missing data for it was not as high as for other features.

We also decided to investigate correlations between features using a correlation matrix. The correlation matrix was creating from the first 13000 rows from the reference station, but using observations values from validity time instead of reference time. As expected, very strong correlation was estimated between measured temperature and temperature predicted by Aladin. Negative correlation between temperatures and humidity can be caused by a fact, that with increased humidity there often comes rainfall and temperature decreases. Therefore, we would also expect high positive correlation between rainfall and humidity. However, the graph shows very little of the correlation. This may be caused by the invalid values for rainfall containing zeros instead of -999 as shown in Section 3.2. Because the correlation matrix was used mainly for intuitive purposes, we did not test for significance of the correlations among the features.

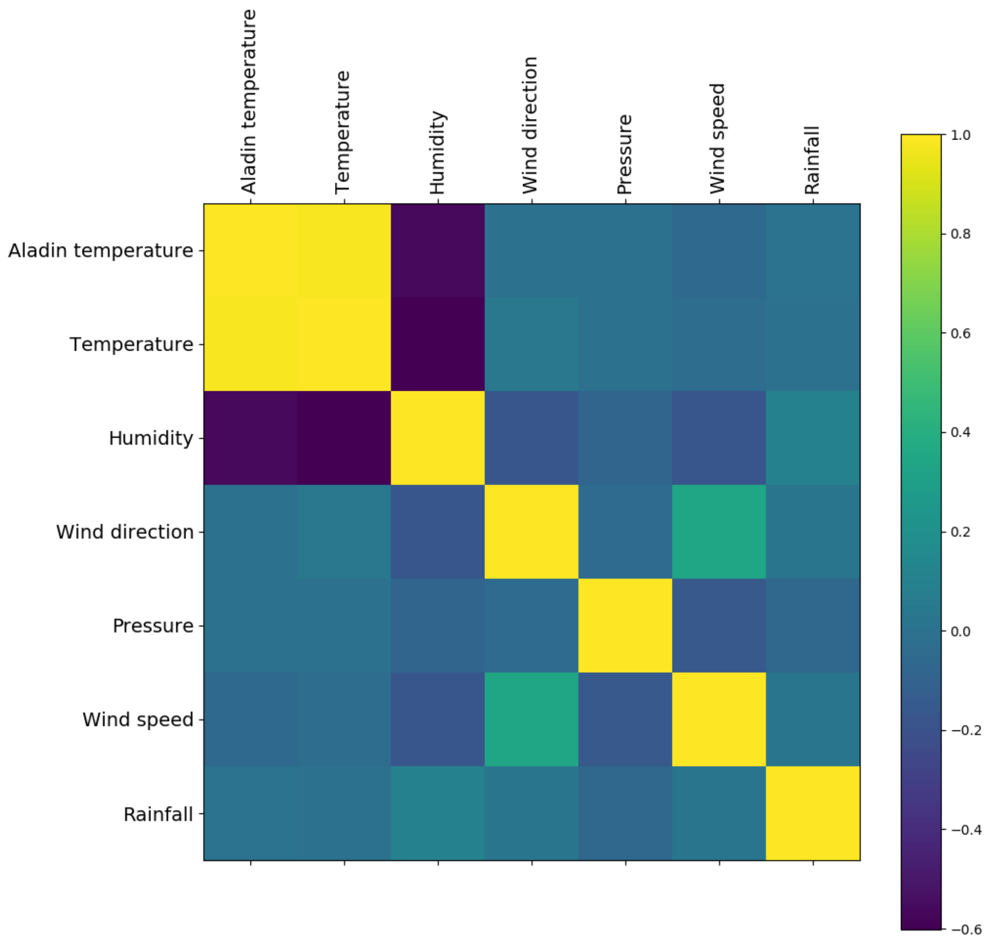


Figure 3.3: Correlation matrix using first 13000 records for the reference station.

Chapter 4

General model

In Chapter 3, we presented arguments about low importance of features like wind speed or rainfall. In this chapter we create various models that consist only from temperature features, features derived from them and humidity.

Rolling window

In the Hajdin's thesis the rolling window of size 60 with 12 hour period was used. Remind that the model that uses only measured and predicted temperature can be defined as in Equation 4.1, and we refer to it as to *reference* model or simply to *ref.* The model is the same as the one from Equation 2.1.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h \qquad h \in 1, ..., 12 \tag{4.1}$$

Hajdin claimed that when using the reference model, the longer rolling window did not improve the results. However, when using the period of 12 hours, two groups of records related to two different hours of a day are obtained in a training set. The model then learns from Aladin predictions made for two distinct hours of a day which might not be related. Using matrix notation and period of 24 hours, the reference model is defined in Equation 4.2.

$$\begin{pmatrix} 1 & T_i & p_i^h \\ 1 & T_{i-24} & P_{i-24}^h \\ 1 & T_{i-48} & P_{i-48}^h \\ ... & ... & ... \\ 1 & T_{i-(m-1)*24} & P_{i-(m-1)*24}^h \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} T_{i+h} \\ T_{i-24+h} \\ T_{i-48+h} \\ ... \\ T_{i-(m-1)*24+h} \end{pmatrix} \tag{4.2}$$

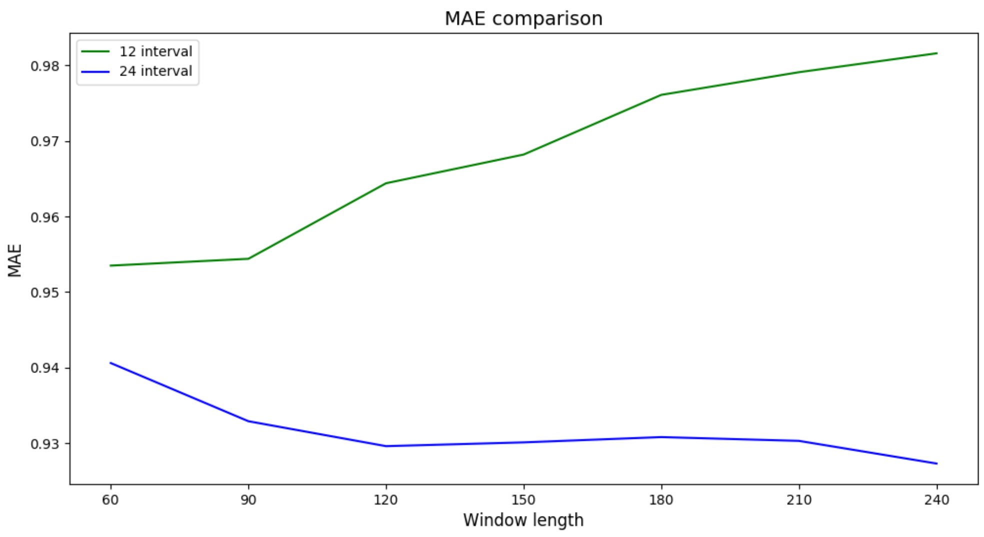


Figure 4.1: MAE comparison for 12 to 24 hour periods of the rolling window. The window length on x-axis is relative to 12 hour period.

To investigate if the 24 hour period is more plausible than the 12 hour period, we plotted MAE and MSE metrics for the growing length of the rolling window using those periods. Figures 4.1 and 4.2 shows those plots. For both MAE and MSE it can be seen that with the increasing rolling window length those metrics grows for the 12 hour period and decrease for 24 hour period. Tables 4.1 and 4.2 shows more comprehensive results. We evaluated all the models using the same dataset, consisting of 14615 records for the reference station. To directly compare our results with the Hajdin's thesis, we used only the data that were also available for Hajdin.

Table 4.1: Results for the reference model with 12 hour period.

Window length	60	90	120	150	180	210	240
MAE	0.9535	0.9544	0.9644	0.9682	0.9761	0.9791	0.9816
MSE	1.5891	1.5825	1.6075	1.6121	1.6273	1.6404	1.6458

Due to the trend that can be seen from the plots, showing that 24 hour period outperformed the 12 hour one in both MAE and MSE, we later use 24 hour period in all of our future models if not said otherwise. Using this approach we have distinct training sets

Table 4.2: Results for the reference model with 24 hour period.

Window length	30	45	60	75	90	105	120
MAE	0.9406	0.9329	0.9296	0.9301	0.9308	0.9303	0.9273
MSE	1.5593	1.5262	1.5189	1.5128	1.5136	1.5132	1.5032

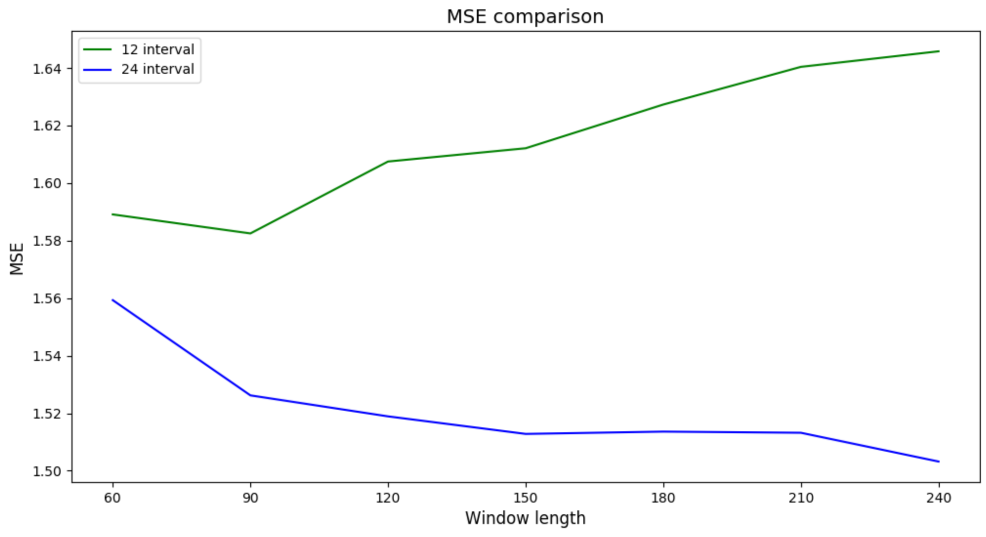


Figure 4.2: MSE comparison for 12 to 24 hour periods of the rolling window. The window length on x-axis is relative to 12 hour period.

for every hour of a day. Also when we later refer to reference model without specifying the rolling window length, we mean the length of 120 samples.

Adding new features

In order to further improve MAE and MSE metrics, we decided to create new features and append them to the reference model. We created those features based on temperature measurements while ignoring other features. All the models, described below, were tested based on the reference station data and using only the records, that were available in the Hajdin's thesis. A period of 24 hours for the rolling window was used. In the later text, when we prefix any model name with integer, it specifies the rolling window length.
In the later text some models are compared using slightly different amount of samples. It is because of the fashion the features are added to the models. When we use past data, for some samples we just do not have the past information available and those samples are therefore not included in a training set. However, when we compare two models, we always compare them on exactly the same samples otherwise the comparison could be misleading.

Minimum and maximum temperature

One of the features we experimented with were minimum and maximum from the temperature values which were measured twelve hours before the time the forecast was created, up to the time it was created (reference time). For example if we predicted temperature for 5AM, we took temperature values between 12AM up to 12PM of the previous day. 12AM is the time when the last Aladin forecast was made. We therefore got thirteen temperature values and picked minimum respectively maximum from them. More formally, using symbols from Section 4.1, when predicting temperature y_i^h we took minimum and maximum from measured temperature values between T_{i-12} up to T_i. In the later text, we refer to those features simply as to *minimum* and *maximum*. We tested adding of minimum and maximum using various rolling window lengths. The results of adding minimum to the reference model are shown in Table 4.3. Later we refer to this model as to *min* and the model is defined in Equation 4.3. Similarly, *max* model can be defined.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3(min(T_{i-12}, ..., T_i)) \qquad h \in 1, ..., 12 \tag{4.3}$$

Table 4.3: Models for rolling window lengths were evaluated on the same 14615 records.

Model	60-ref	60-min	90-ref	90-min	120-ref	120-min
MAE	0.9296	0.9238	0.9308	0.9192	0.9273	0.9147
MSE	1.5189	1.5152	1.5136	1.4924	1.5032	1.4756

From the results we see that adding the minimum improved both MAE and MSE for all the rolling window lengths. Using Wilcoxon paired test with 95% confidence level, we tested if the difference between means of absolute errors and means of squared errors for the reference model and the min model is significant. Based on the test, the p-values for both the hypotheses, when rounded to six decimals, were equal to zeros. Thus, we deny those hypotheses and consider that the means of absolute errors resp. squared errors of the min and reference models are significantly different.
We expected that adding of the maximum would have similar impact on the improvement as the minimum had. The max model improved MAE for all the examined rolling window lengths however MSE worsen for lengths of 60 and 90 samples. The results are captured in Table 4.4.

By including both the minimum and maximum features in one model we got the best MAE and MSE results for the rolling window length of 120. We call this model *min-max*. The model also had the best MAE from the previously tried models, but got worse in MSE for the rolling window size of 60 compared to the min model. The results are captured in Table 4.5.

Table 4.4: Models for all rolling window lengths were evaluated using the same 14615 records.

Model	60-ref	60-max	90-ref	90-max	120-ref	120-max
MAE	0.9296	0.9290	0.9308	0.9243	0.9273	0.9187
MSE	1.5189	1.5406	1.5136	1.5137	1.5032	1.4950

Table 4.5: Models for all rolling window lengths were evaluated using the same 14615 records.

Model	60-min	60 min-max	90-min	90-min-max	120-min	120-min-max
MAE	0.9238	0.9224	0.9192	0.9114	0.9147	0.9045
MSE	1.5152	1.5349	1.4924	1.4891	1.4756	1.4629

To investigate that the improvement for the rolling window size of 120 did not occur by chance, we decided to test errors differences for the 120-min and 120-min-max models. Using Wilcoxon paired test, rounded p-values were 0.000007 for absolute error test and 0.000039 for squared error test. Thus, we consider the 120-min-max and 120-min models errors to be significantly different, and we only use the 120-min-max model for the later computations. The difference between the 120-min and 120-max models was also significant with p-values 0.040309 and 0.002992 for absolute and squared errors respectively. Eventually the combination of both features improved the MAE from 0.9273 to 0.9045 and MSE from 1.5032 to 1.4629 compared to the reference model.

For further analysis of improvement we decided to construct graphs that visualize improvement for all seasons separately. In the later text we refer to them as to *improvement graphs*. The improvement graph compares two models improvement in absolute error. It is divided into four subgraphs, one for each season. On the x-axis there are hours for which temperature was predicted and on the y-axis there is absolute improvement. The absolute improvement is computed relative to Aladin predictions. There is a gap between 12AM and 1PM to explicitly emphasize that new Aladin predictions were made during that time.
Figure 4.3 shows comparison of the 120-min and 120-max models. It can be seen that the 120-max has significantly greater improvement in autumn and winter between 2AM to 6AM. For hours between 1PM to 12PM the 120-min seems better.
In Figure 4.4 we compare the 120-ref and 120-min-max models. The significant improvement from the 120-max model remained as also the improvement during 1PM and 12PM. Nevertheless, the 120-min-max model also got worse for some hours. The greatest difference between improvement during autumn and winter compared to spring and summer is between 6AM to 10AM where the improvement during autumn and winter

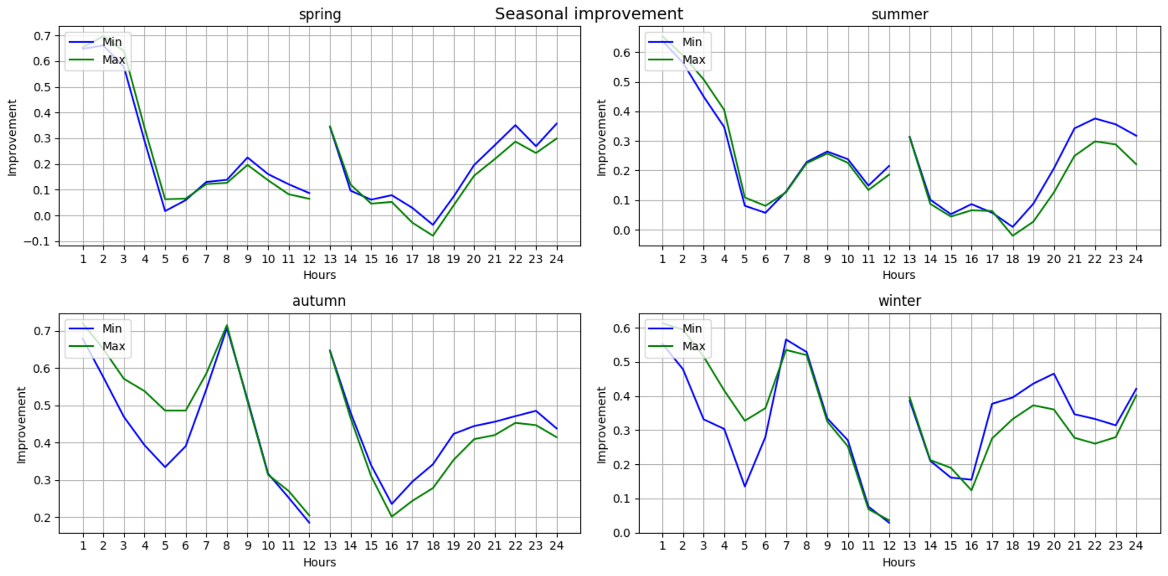


Figure 4.3: Improvement comparison for the 120-min and 120-max models.

is more significant.

We investigated if the improvement results are similar across different stations, but unfortunately the results seems to be very station specific. Thus, it is difficult to make general conclusion about the improvement gained with those features. In practise, we would suppose to examine the model improvement behaviour across the year. For hours and periods when the model get worse, either compared to the reference model or Aladin, the predictors can be turned off.

Aladin errors

Since addition of the minimum and maximum features decreased MAE and MSE metrics, we continued in adding other features derived from temperature values. We decided to include Aladin error that was made 24 hours before the hour for which we predict. The model is formally defined in Equation 4.5. In the later text we refer to this model as to *aladin24*.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 (P_{i-24}^h - y_{i-24}^h) \qquad h \in 1, ..., 12 \tag{4.4}$$

We evaluated the model using the same rolling windows lengths as for the min and max models. All the models were evaluated on the same 14603 samples. The results are captured in Table 4.6. The model worsen both MAE and MSE for the rolling window size of 60, for size of 90 at least the MAE improved and for size of 120 both MAE and MSE improved. However, even for the rolling window size of 120 samples, the improvement seemed very minor compared to the min or min-max models. Nevertheless,

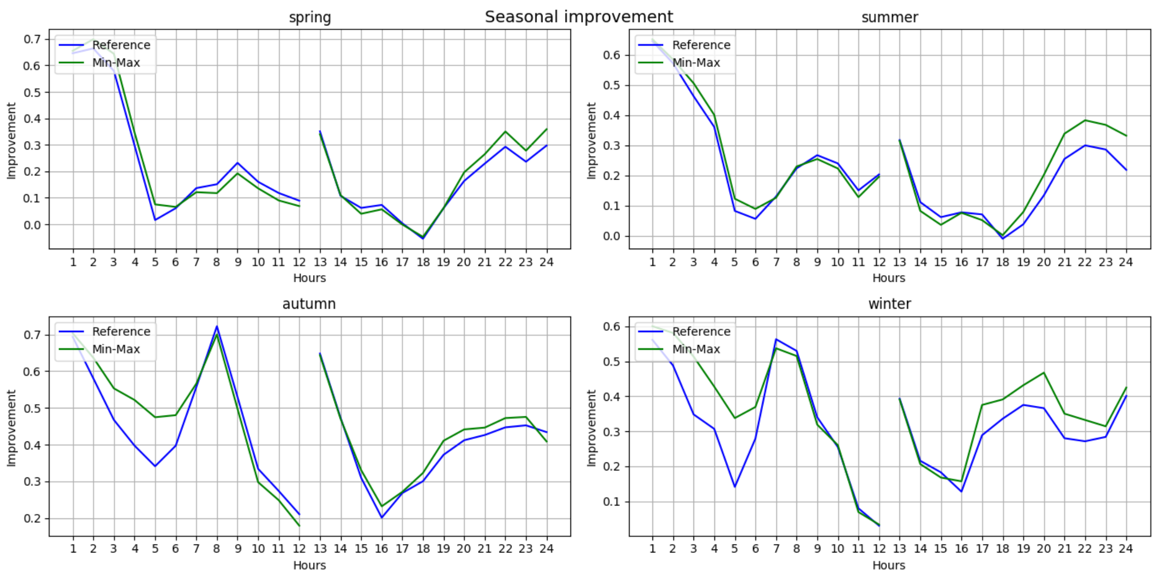


Figure 4.4: Improvement comparison for the 120-ref and 120-min-max models.

we decided to test if the improvement, even though small, is significant for the 120 samples rolling window size. Using Wilcoxon paired test, the p-values were 0.046574 for absolute error and 0.011380 for squared error. So with 95% confidence we can deny the null hypothesis and trust that the real means for both absolute and squared errors are different. Even though the 120-aladin24 is significantly better than the 120-ref the improvement seems unsatisfying.

Table 4.6: Models for all rolling window lengths were evaluated using the same 14603 records.

Model	60-ref	60-aladin24	90-ref	90-aladin24	120-ref	120-aladin24
MAE	0.9286	0.9335	0.9298	0.9291	0.9262	0.9252
MAE	1.5146	1.5290	1.5089	1.5101	1.4982	1.4970

Despite the significance of improvement for the rolling window size of 120 samples, we decided not to use aladin24 for the later computations, as the decrease in both MAE and MSE was very small. We trusted that adding Aladin error to the model might improve the results, thus we tried slightly different version of the model. Instead of using error that Aladin made 24 hours before, we used Aladin error from reference time (the time when Aladin made the prediction) as a new feature for the reference model. Thus, we added Aladin error from 0AM or 12AM based on hour for which we predicted. For hours between 1PM up to 12PM we added Aladin error from 12AM. For the rest hours error from 0AM was used. The model can formally be written as

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 (P_{i-12}^{12} - y_{i-12}^{12}) \qquad h \in 1, ..., 12 \tag{4.5}$$

by $P^{12}_{i-12} - y^{12}_{i-12}$ we mean Aladin error from reference time. In the later text we refer to this model as to *aladinP*. We evaluated the model using 14614 samples and captured the results in Table 4.7.

Table 4.7: Models for all rolling window lengths were evaluated using the same 14614 records.

Model	60-ref	60-aladinP	90-ref	90-aladinP	120-ref	120-aladinP
MAE	0.9296	0.9228	0.9308	0.9206	0.9273	0.9151
MSE	1.5190	1.5103	1.5137	1.4970	1.5033	1.4789

From MAE and MSE metrics in can be seen that the aladinP model performed better than the reference model for all the examined window lengths. Using Wilcoxon test for the improvement significance, the p-values for both null hypotheses were, when rounded to six decimals, equal to zeros. We therefore claim aladinP to have significantly smaller absolute and squared errors that the reference model. The improvement gained with the aladinP model was greater than the one gained with the aladin24 model. When we examined the improvement graph for the aladinP model, the curves followed similar shapes as those for the min or min-max models, but with smaller improvement for morning hours in winter. The greatest improvement gained with the aladinP model was between 2AM to 6AM during autumn months. For other periods and hours the improvement was minor. When we examined the improvement graph for the aladin24 model the improvement almost disappeared. The only systematical improvement was seen between 16PM and 24PM in autumn moths.
Finally, we tried to combine those models into one. We constructed model from Equation 4.6. We later refer to it as to *aladinCombined*.

$$y^h_i = \theta_0 + \theta_1 T_i + \theta_2 P^h_i + \theta_3 (P^h_{i-24} - y^h_{i-24}) + \theta_4 (P^{12}_{i-12} - y^{12}_{i-12}) \qquad h \in 1, ..., 12 \tag{4.6}$$

When we compared the aladinP and aladinCombined models, MAE for the aladinCombined increased from 0.9134 to 0.9144 and MSE increased from 1.4718 to 1.4738. The models were compared using 14590 samples.
From the results we obtained when using Aladin errors we do not suppose that Aladin error from 24 hours before can be useful in improving predictions accuracy. On the other hand, Aladin error from reference time seemed to improve the results for various rolling window lengths. It might be caused by a fact, that Aladin errors from aladinP model are closer to the times for which we forecast.

Fixed hours temperatures

Next feature we tried to include in the reference model was simply measured temperature from chosen fixed time. The reference model already contains the temperature from reference time. We decided to add temperature from 6AM when predicting temperature for 1PM up to 12PM and from 6PM when predicting temperature for 1AM up to 12AM. The choice of 6AM and 6PM was mostly random, but we wanted to choose the hour that is not too close to 12AM or 12PM. Because of that we expected that we could capture temperature growth or decrease which might improve the predictions. The model is defined in Equation 4.7 and is later referred to as a *6am6pm*.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 T_{i-6} \qquad h \in 1, ..., 12 \tag{4.7}$$

The results of using 6am6pm model are captured in Table 4.8.

Table 4.8: Models for all rolling window lengths were evaluated using the same 14615 records.

Model	60-ref	60-6pm6am	90-ref	90-6pm6am	120-ref	120-6pm6am
MAE	0.9296	0.9120	0.9308	0.9079	0.9273	0.9043
MSE	1.5189	1.4902	1.5136	1.4691	1.5032	1.4564

Addition of 6AM/6PM helped to improve both MAE and MSE for all examined rolling window lengths. P-values for the improvement hypotheses were, when rounded to six decimals, zeros. We therefore consider the improvement to be significant. When investigating the improvement graph we figured out that the improvement is visually similar to the one gained with the min-max model. It would be better if the improvement happened for distinct periods or hours, so we could expect that combining the models would further improve MAE and MSE metrics. Based on MAE and MSE values, the improvement to the reference model was greater than with the aladinP model and also greater than the improvement gained with the min-max model.

Lagged temperature

The approach we used in the min-max model is similar to what is in the time series terminology referred to as lagging or lagged variables. The lag is in fact a delay. In the 6am6pm model we lagged measured temperature from reference time by six hours. More common approach is to include all delayed values up to the final lagged value. Instead of using only T_i and T_{i-6} temperatures for 6am6pm model, it would mean to

also include T_{i-1}, T_{i-2}, T_{i-3} and T_{i-4} temperatures. The problem when adding all those features to our models is that the rolling window size, which is by default 120, might be too small for our models to work as desired.
We first tried to lag measured temperature by one hour from reference time getting the model

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 T_{i-1} \qquad h \in 1, ..., 12$$

When tested on 14626 samples MAE increased from 0.9270 to 0.9315 and MSE from 1.5024 to 1.5196 compared to the reference model. Therefore, we assume that the feature did not help. We also experimented with models where measured temperature and temperature predicted by Aladin were lagged by more hours from reference time and with the combinations of those. However, none of those resulted in improvement that we considered satisfying. When lagging temperature predicted by Aladin by one hour from reference time, we in fact get a very similar model as the aladinP.
We also tried lagging from validity time. It is necessary to lag measured temperature at least by twelve hours, otherwise the models would learn from information that was not accessible at the time of making the prediction. We examined models

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 T_{i-24} \qquad h \in 1, ..., 12$$

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 P_{i-24}^h \qquad h \in 1, ..., 12$$

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 T_{i-24} + \theta_4 P_{i-24}^h \qquad h \in 1, ..., 12$$

For now, we refer to those as to A, B, C starting from top to bottom. The results of evaluating the models using 14579 samples are captured in Table 4.9.

Table 4.9: Lagged models

Model	120-ref	120-A	120-B	120-C
MAE	0.9260	0.9246	0.9264	0.9268
MSE	1.4982	1.4928	1.4936	1.4968

The models seemed to introduce no satisfying improvement. The A model made minor improvement for both MAE and MSE. The B and C models only improved MSE, but even less than the model A. When using Wilcoxon paired test we could not deny the null hypothesis that the 120-ref and 120-A models have equal means of absolute errors. The p-value for this test was 0.065393 which is greater than 0.05, so we could not deny it with 95% confidence level.
We also tried model which lags temperature predicted by Aladin by one hour from validity time. The model is defined in Equation 4.8, and we refer to it as to *Aladin-lag1*.

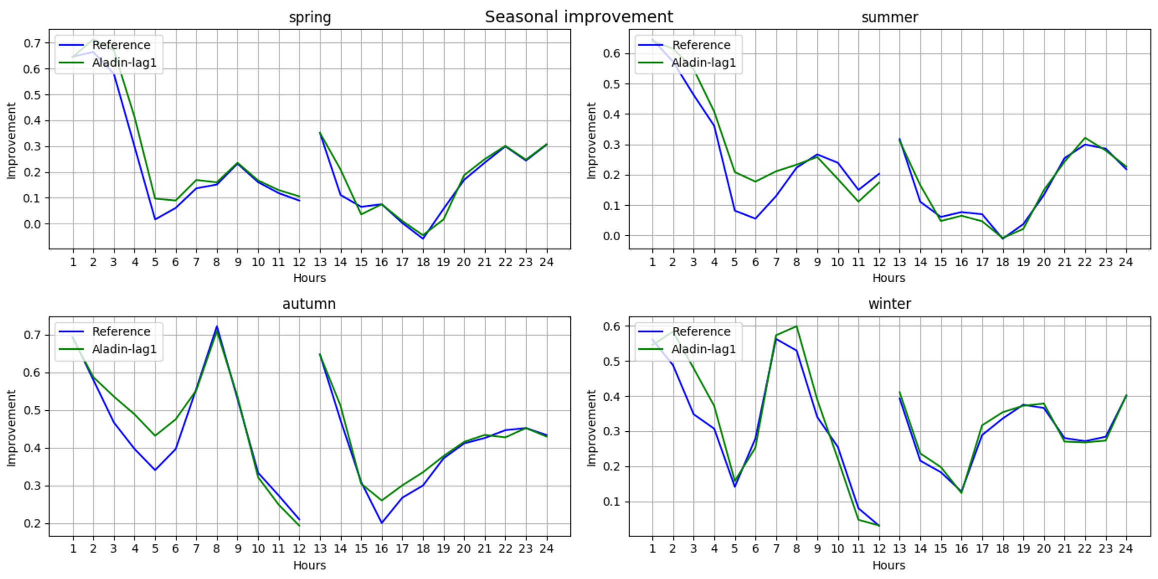


Figure 4.5: Improvement comparison for the 120-ref and 120-Aladin-lag1 models.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 P_i^{h-1} \qquad h \in 1, ..., 12 \tag{4.8}$$

Compared to the reference model with 120 rolling window length using 14626 samples the model improved MAE from 0.9270 to 0.9087 and MSE from 1.5024 to 1.4684. Using Wilcoxon paired tests we could confirm that the improvement is significant with 95% confidence level. The important property of this model is that from the improvement graph (Figure 4.5) it seems, that it improves the forecasts during different hours for summer months compared to the min-max or 6am-6pm models. New improvement for summer is seen between 5AM and 7AM. This could indicate that combining Aladin-lag1 model with the min-max model might further improve MAE and MSE.

Lastly we tried to lag the aladinP model by 24 hours. It means that we included both Aladin error from reference time and Aladin error from 24 hours before that reference time. For that model both MAE and MSE increased compared to the aladinP model. We considered it important to experiment with lagged values as it is common approach in many time series prediction approaches. Unfortunately this type of features seemed to introduce improvement only for the Aladin-lag1 model. Other lagged features did not seem to decrease models errors.

Moments

The last features we decided to try were moments. The moments were created from temperature values measured twelve hours before reference time up to the reference time, thus using the same temperature values as for the minimum or maximum features. The models were defined as

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 moment \qquad h \in 1, ..., 12$$

where *moment* in one of the mean, variance, skewness and kurtosis, created from temperature values between T_{i-12} and T_i. We refer to those models as to *Mean*, *Var*, *Skew* and *Kur* respectively. The results of using those models are captured in Tables 4.10, 4.11 and 4.12.

Table 4.10: Moments models evaluated using 14615 samples with the rolling window length of 60.

Model	60-ref	60-Mean	60-Var	60-Skew	60-Kur
MAE	0.9296	0.9093	0.9162	0.9348	0.9387
MSE	1.5189	1.4818	1.5012	1.5443	1.5641

Table 4.11: Moments models evaluated using 14615 samples with the rolling window length of 90.

Model	90-ref	90-Mean	90-Var	90-Skew	90-Kur
MAE	0.9308	0.9042	0.9125	0.9356	0.9360
MSE	1.5136	1.4611	1.4822	1.5332	1.5354

Table 4.12: Moments models evaluated using 14615 samples with the rolling window length of 120.

Model	120-ref	120-Mean	120-Var	120-Skew	120-Kur
MAE	0.9273	0.8999	0.9076	0.9313	0.9277
MSE	1.5032	1.4461	1.4696	1.5211	1.5089

We can see that kurtosis and skewness worsen MAE and MSE for all the rolling windows lengths. On the other hand, mean and variance improved MAE and MSE for all the tested models. Mean had the best improvement impact from the moments we tried. As both mean and variance seem to decrease MAE and MSE we decided to use both these features in the next model getting so called *Mean-Var* model.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 mean + \theta_4 var \qquad h \in 1, ..., 12$$

We compared the Mean-Var and Mean models using the rolling window size of 120 and 14615 samples. Compared to the Mean model MAE increased from 0.8999 to 0.9017

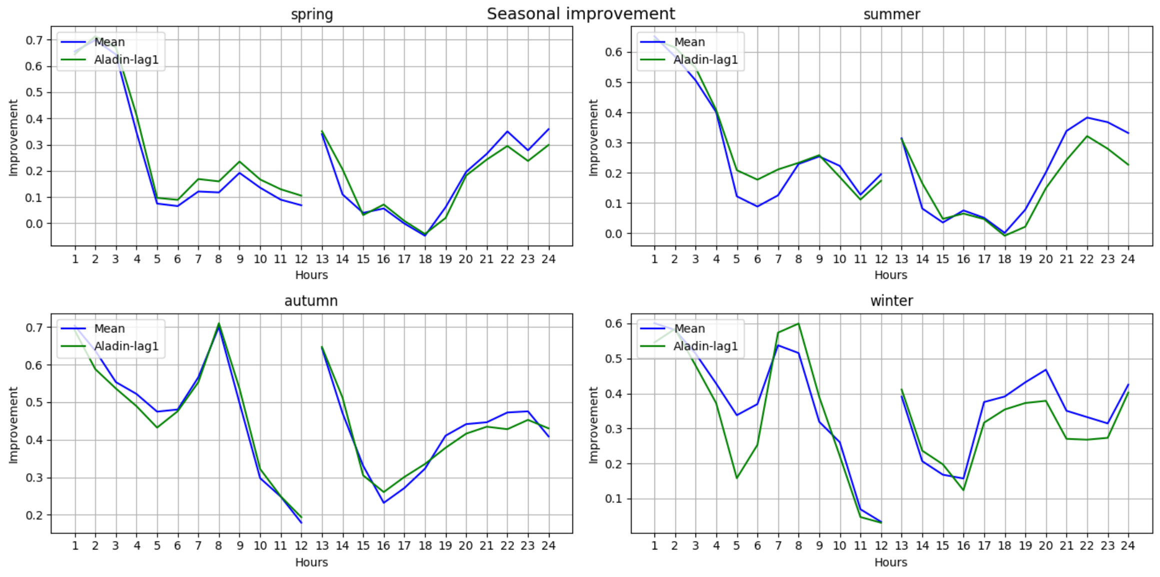


Figure 4.6: Improvement comparison for the 120-Aladin-lag1 and 120-Mean models.

and MSE increased from 1.4461 to 1.4620. Using Wilcoxon paired test, we could not deny that the models have equal means of absolute errors, because the p-value was 0.998514. We also tried to test if the means of squared errors are the same. The p-value for this hypothesis was 0.884324 thus we could not deny it. Because the Mean-Var model did not introduce any improvement, we decided not to use it for further experiments.

The improvement comparison for the 120-Aladin-lag1 and 120-Mean is shown in Figure 4.6. As the improvement is different for those models it might be good to combine them. For other models the improvement comparison with the Mean model had less different results.

We also tried to figure out why addition of variance causes improvement. It could mean that Aladin is prone to change accuracy based on variability in weather. We visualized improvement graphs for various stations to see if we could catch some specific behaviour about that improvement. The behaviour seemed highly station specific, and we could not make general assumptions about the improvement gained with variance.

Combining features

After investigating the improvement that was gained by creating new features from measured and predicted temperature, we combined the features that caused enhancement into new models.

The most naive approach was to use all the features that improved the reference model. We used model composed of the min-max, AladinP, 6am-6pm, Aladin-lag1 and Mean models. The model is defined in Equation 4.9.

$$\begin{aligned}y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3(min(T_{i-12}, ..., T_i)) + \theta_4(max(T_{i-12}, ..., T_i)) + \\ \theta_5(P_{i-12}^{12} - y_{i-12}^{12}) + \theta_6 T_{i-6} + \theta_7 P_i^{h-1} + \theta_8 mean(T_{i-12}, ..., T_i) \qquad h \in 1, ..., 12\end{aligned} \tag{4.9}$$

The model was evaluated using 14577 samples and introduced improvement compared to the reference model. When compared using the same sample size MAE decreased from 0.9261 to 0.9003 and MSE decreased from 1.4984 to 1.4707. However, when compared to the 120-Mean model, MAE increased from 0.8991 to 0.9003 and MSE increased from 1.4416 to 1.4707. We therefore do not consider it suitable to use this model instead of using e.g. 120-Mean model. The model uses eight features and the intercept term, so the rolling window size of 120 samples might be too small. Anyhow, larger rolling window size would cause too many observations to be skipped. Also the rolling window size would be getting close to the count of unseen samples, what might cause problems with generalization.
Rather than using all those features, we created next models based on the improvement graphs. The improvement graph for the Aladin-lag1 model seemed different compared to the min-max or Mean models. Therefore, we decided to examine models defined in Equations 4.10, 4.11 and 4.12. We later refer to those as to *C1*, *C2* and *C3*. All those models are composed of the Aladin-lag1 model. They are further composed of the min-max, Mean and 6am-6pm models, based on the order in which they are defined.

$$\begin{aligned}y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3(min(T_{i-12}, ..., T_i)) + \\ \theta_4(max(T_{i-12}, ..., T_i)) + \theta_5 P_i^{h-1} \qquad h \in 1, ..., 12\end{aligned} \tag{4.10}$$

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 P_i^{h-1} + \theta_4 mean(T_{i-12}, ..., T_i) \qquad h \in 1, ..., 12 \tag{4.11}$$

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 P_i^{h-1} + \theta_4 T_{i-6} \qquad h \in 1, ..., 12 \tag{4.12}$$

The models were evaluated using 14614 samples. From the results in Table 4.13 we can see that all those models improved both MAE and MSE compared to the models they were composed of. Using Wilcoxon paired tests, we could with 95% confidence confirm that the improvement is significant for both MAE and MSE. The lowest MAE and MSE was gained with C2 model. Since this model is composed of the Mean and Aladin-lag1 models, and from the models we introduced the Mean model had the largest impact on both MAE and MSE improvement, it is quite expected.

Table 4.13: Evaluated using 14614 samples with the rolling window length of 120.

Model	ref	min-max	Aladin-lag1	Mean	6am-6pm	C1	C2	C3
MAE	0.9273	0.9045	0.9091	0.8999	0.9043	0.8952	0.8907	0.8935
MSE	1.5033	1.4629	1.4693	1.4462	1.4565	1.4460	1.4279	1.4336

Because combination of different models, constructed for same goal, often results in better outcomes [16], we averaged the outcomes of the C1, C2 and C3 models. So when we predict temperature at time i, the y_i^{Cj} is the value that the Cj model predicted at time i and $j \in \{1, 2, 3\}$. The final value that we predict is simply $y_{final} = \frac{\sum_{i=1}^{j} y_i^{Cj}}{j}$. We refer this model as to *ensemble1* model. After we evaluated it using 16413 samples MAE was equal to 0.8891 and MSE to 1.4242.
Finally, we constructed ensemble from the Aladin-lag1, AladinP, min-max, Mean and 6am-6pm models, but the results were worse compared to the results for the *ensemble1* model.

Humidity

Until now we have been using only temperature related features in our models. However, in Chapter 3 we discussed feature importances and from Figure 3.2 we see, that humidity should be more important compared to other non temperature related features. Therefore, we decided to add measured humidity from reference time into the reference model. The model is defined in Equation 4.13, and we refer to it as to *hum*.

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 humidity_i \qquad h \in 1, ..., 12 \tag{4.13}$$

From Table 4.14 we see that adding humidity improved the results compared to the reference model for all tried rolling window lengths. The disadvantage of adding humidity is that for other stations we must skip more predictions, due to more invalid records obtained in the data, compared to models without humidity.

Table 4.14: Evaluated on 14627 samples.

Model	60-ref	60-hum	90-ref	90-hum	120-ref	120-hum
MAE	0.9292	0.9241	0.9303	0.9189	0.9270	0.9136
MSE	1.5179	1.5169	1.5125	1.4920	1.5023	1.4747

Weights and rolling window lengths

To make newer records more important than the older and to further improve the results we multiplied each record in the rolling window by a weight function. The function was the same as in the Hajdin's thesis and was previously defined in Chapter 2. As the ensemble1 model had the best results from the models we tried so far, we decided to use this model. Table 4.15 shows the results for various values of a, the base of our weight function. The best results were obtained for a equal to 0.97.

Table 4.15: Evaluated using 14613 samples with the rolling window length of 120 and ensemble1 model.

Weighting base	1	0.99	0.98	0.97	0.96	0.95
MAE	0.8891	0.8852	0.8829	0.8818	0.8820	0.8832
MSE	1.4242	1.4150	1.4098	1.4084	1.4104	1.4154

The ensemble1 model is constructed from the models that all have same rolling window lengths. As ensemble of different models improved the results, we also tried to change the rolling window lengths for those models. Using a equal to 0.97 for the weight function and combining different rolling window lengths for the models from the ensemble1, we could not further improve the results. The window size of 120 seemed to be most appropriate for all the models in the ensemble1. We experimented with the rolling window sizes of 30, 60, 90 and 120.
Because the addition of humidity improved the results, we decided to adjust the ensemble1 model by adding measured humidity from reference time into every model in the ensemble. We refer to this model as to *ensemble2*. The results are shown in Table 4.16.

Table 4.16: Evaluated using 14613 samples with the rolling window length of 120 and ensemble2 model.

Weighting base	1	0.99	0.98	0.97	0.96	0.95
MAE	0.8882	0.8841	0.8818	0.8811	0.8817	0.8836
MSE	1.4266	1.4168	1.4118	1.4112	1.4145	1.4214

After evaluating the ensemble2 model, we got the best MAE for the weighting base equal to 0.97, but slightly worse MSE compared to the ensemble1 model. The advantage of the ensemble1 is that it can perform more predictions for other stations because of fewer invalid values obtained in the data. But according to the lowest MAE error, we chose the ensemble2 for later experiments.

Regularized models and different learning algorithms

Until now we were using only linear regression models. In this section we investigate usage of regularized regression, support vector regression and neural networks.

Regularized regression

The regularized regression models that we tried are ridge, lasso and elastic net regression. To check if the regularized regression improves the results we first applied it to the hum model.

Table 4.17: Evaluated on 14627 samples with the rolling window length of 120, using the hum model and no scaling.

Model	OLS	Ridge	Lasso	Elastic net
MAE	0.9136	0.9143	0.9211	0.9216
MSE	1.4747	1.4750	1.4912	1.4904

Table 4.18: Evaluated on 14627 samples with the rolling window length of 120, using the hum model and standardization.

Model	OLS	Ridge	Lasso	Elastic net
MAE	0.9136	0.9154	0.9225	0.9242
MSE	1.4747	1.4776	1.4950	1.4965

Table 4.19: Evaluated on 14627 samples with the rolling window length of 120, using the hum model and min-max normalization.

Model	OLS	Ridge	Lasso	Elastic net
MAE	0.9136	0.9261	0.9232	0.9399
MSE	1.4747	1.4958	1.4951	1.5325

Due to difference in scales between temperatures and humidity, we evaluated the models in three cases. In the first case we did not use feature scaling, in the second we applied standardization and finally, we used min-max normalization.
We used 5 as the number of splits in the scikit-learn implementation of time series cross validation and used it with grid search to find the proper values of regularization terms for each position of the rolling window. From the results in Tables 4.17, 4.18 and 4.19 we see that neither ridge, lasso or elastic net improved the results for the hum model.

Table 4.20: Evaluated on 14614 samples with the rolling window length of 120, using the C2 model and no scaling.

Model	OLS	Ridge	Lasso	Elastic net
MAE	0.8907	0.8930	0.9063	0.9104
MSE	1.4279	1.4300	1.4577	1.4684

Table 4.21: Evaluated on 14614 samples with the rolling window length of 120, using the C2 model and standardization.

Model	OLS	Ridge	Lasso	Elastic net
MAE	0.8907	0.8974	0.9054	0.9180
MSE	1.4279	1.4393	1.4564	1.4882

Also no improvement was seen for the C2 model and models with more features. In all the tried cases regularization did not help. The results for the C2 model are stored in Tables 4.20, 4.21 and 4.22. Same as for the hum model, we got the smallest errors when no scaling was used.

We could observe small improvement for some concrete values of regularization terms, but due to possible generalization problems when applied to other stations, we decided not to include them and rely on the grid search results. Finally, we tried to add polynomial features. The results for those models were significantly worse with and without regularization.

Therefore, we decided not to further experiment with regularized regression models. Originally we expected, that ridge regression could improve the results because linear regression can suffer from multicollinearity in data [22].

SVR

Support vector regression is markedly different learning algorithm compared to linear regression. We decided to test SVR for the hum model as we did with regularized regression. First we tried a linear kernel. In all cases we used standardization because scaling is required for SVR and standardization is often being used [21]. By default, we used 0.1 for ϵ that controls size of a regression tube because it is the default parameter in the scikit-learn library [2]. To test if the smaller value of ϵ changes the results, we also used value of 0.01. The smaller ϵ did not improve the results which can be found in Table 4.23. For linear kernel it seemed that different values of C and ϵ had almost no effect on the results and also caused very little changes to the training errors.

For the hum model SVR did not improve MSE and MAE was improved very little for

Table 4.22: Evaluated on 14614 samples with the rolling window length of 120, using the C2 model and min-max normalization.

Model	OLS	Ridge	Lasso	Elastic net
MAE	0.8907	0.9244	0.9057	0.9402
MSE	1.4279	1.4979	1.4568	1.5380

C equal to 50 and 100. The hum model estimated using linear regression had MAE equal to 0.9136 and MSE equal to 1.4747. SVR training is more time consuming than linear regression, so we do not prefer to use it.
Using RBF kernel more complex functions can be learnt [21]. The problem with the RBF kernel was that it was easy to overfit or underfit the data, and we could not find a model to generalize well. We consider RBF kernel too complex for the problem we solved. With the hum model we underfitted the data for C equal to 1 and for C equal to 10 the model seemed to overfit. For C set to 50 the model fit the data much better than the linear kernel, but it did not generalize well to new samples.

Table 4.23: Evaluated on 14627 samples with the rolling window length of 120, using the hum model and standardization. The 50eps is the model with ϵ equal to 0.01.

C	1	50	50eps	100
MAE	0.9164	0.9123	0.9143	0.9123
MAE train	0.8571	0.8518	0.8503	0.8518
MSE	1.5156	1.5064	1.5124	1.5066
MSE train	1.3636	1.3542	1.3576	1.3543

Table 4.24: Evaluated on 14627 samples with the rolling window length of 120, using the hum model, standardization and RBF kernel.

C	1	2	5	10	50	100
MAE	2.2665	1.6161	1.4447	1.3118	1.2562	1.2920
MAE train	1.1616	0.8604	0.7848	0.7166	0.6284	0.5976
MSE	11.6196	6.0917	4.7783	3.7919	3.3989	3.6262
MSE train	3.2032	1.6351	1.3318	1.1233	0.9519	0.8982

We grid searched over possible values of C from the set {1, 3, 5, 10, 20, 50}. For linear kernel the MAE was 0.9140 and MSE was 1.5089. For RBF kernel MAE was 1.2750 and MSE was 3.5505 making RBF kernel inappropriate to use. Trying different *gamma* values, which controls the shape of RBF function, also did not help. The results were best with default gamma value set to 1 / (number of features) [2].

Feedforward and recurrent networks

The last family of learning algorithms we decided to try were neural networks. First we experimented with feedforward architectures. We continued using the rolling window approach and did not follow the approach where the network is trained on a large dataset, contains a lot of lagged values and all predictions are made at once as described in F.Chevalier's article [10]. We used lbfgs solver from the scikit-learn library because it is recommended for short datasets [2] and we consider our 120 samples rolling window to be quite short. We could not improve the results with feedforwarward networks, so we present only short report. The results for the reference model were slightly worse than when using linear regression. For models with more features the results worsen even more. We tried various values of regularization parameter, number of neurons, more hidden layers, activation functions and both min-max normalization and standardization. For example when grid search over hidden layer sizes of (10, 20, 30) and regularization parameter values (2, 1, 0.5, 0.01) using time series cross validation and relu activation function MAE was 0.9316 and MSE 1.5192. For comparison with linear regression MAE was 0.9273 and MSE was 1.5032.

As the last algorithm we tried recurrent neural networks and compared them to simpler linear regression models. First we used the approach in which we applied rolling window and learned new weights for every position, same as for the previous models. The problem was that learning of recurrent networks is much slower compared to linear or regularized regression and the computation was very slow. Instead, we used the approach, in which we shared network weights across the computations. When first training the network for unseen target hour, we stored its weights and associated them with that hour, resulting in 24 separate weights. During initial training we used 50 or 100 epochs, based on the network architecture. For later predictions we moved the rolling window, but instead of training the network from scratch, we used the already trained weights for that hour. As with the shift in the rolling window one training example is appended and one removed, for each shift we executed two additional epochs using the previous weights. That way we sped up the computations and also had last seen record incorporated in the weights before making new prediction.
Even though we sped up the computations using shared weights, the evaluation still took long time. Therefore, we evaluated the networks only on the first 2000 records of the reference station. For same reason we did not use grid search to find proper hyperparameters, but instead tried few chosen architectures. The aim of this thesis was not primary to design proper recurrent network architecture, but rather investigate if some of the tried architectures could outperform or at least gain similar results compared to linear regression.

We used GRU networks instead of LSTM, to further save time and because the results for few tried architectures were similar for GRU and LSTM. First we used features from the reference model, thus measured temperature at time of making the prediction and predicted temperature for target hour. The GRU consisted from 30 units, tanh activation and other parameters were left default according to the keras library settings [1]. We used keras for all recurrent computations. As the optimizer we used RMSprop, because keras recommends it for recurrent architectures [1]. We used MAE as the loss and the batch size was 4 samples long. Data were scaled to a range -1, 1 to fit the range of tanh activation. We also constrained weights not to exceed value 3 in L2 norm as our default regularization technique.
To compare with linear regression we used the reference model, containing the same features. With mentioned recurrent architecture and also with others, we could not outperform linear regression. However, the models are significantly different, so we expected that an average of those models could result in better values than linear regression. Table 4.25 presents the results that compares the mentioned GRU network to the linear regression reference model. The averaged model slightly improved both MAE and MSE. In MAE, the GRU network was better for 1002 samples and linear regression for 998, but when the errors were summed GRU was outperformed.

Table 4.25: Evaluated on 2000 samples with the rolling window length of 120 using features from the reference model.

Model	OLS	GRU	Averaged
MAE	0.8642	0.8733	0.8513
MAE	1.3578	1.3887	1.3361

When we used MSE as the loss function, GRU error increased to 0.8827 and 1.4132 for MAE and MSE respectively, so we retained MAE as the loss function for all next computations.
Quoc V. Le et al. proposed the method for efficiently training simple recurrent networks (RNN) [13]. They used rectifier linear unit (relu) [16] as the activation and used an identity matrix to initialize recurrent weights [13]. Inspired by this approach we reused our previous architecture, but used RNN instead of GRU with relu activations, used the identity matrix for the recurrent weights initialization and scaled the data to a range [0,1]. The results are stored in Table 4.26.

The simple RNN model performed worse in case of both MAE and MSE compared to the GRU model. With the averaged model we got better results for MAE, but worse for MSE.
Using features from the hum or C2 models we could not reproduce improvement gained

Table 4.26: Evaluated on 2000 samples with the rolling window length of 120 using features from the reference model.

Model	OLS	GRU	Averaged
MAE	0.8642	0.8956	0.8606
MAE	1.3578	1.4771	1.3672

with the averaged model. We experimented with different amount of units, dropout regularization [16] and also using no regularization. We also tried to stack two GRUs layers, which lead to the worst results from the tried architectures. Interesting observations is that even when the network errors differed a lot from linear regression errors, the average of those models was in all cases close to the linear regression results.
We tried to analyse if the network does not strongly overfit the data. We used the same strategy as for SVR. We could not observe significant difference between train and validation error. The results were however harder to interpret as the loss was computed from scaled features.

We showed that GRU networks could reach similar results using the features from the reference model compared to linear regression and also that the combination of those models could improve the results. Due to computational requirements we did not continue with GRU approach, but we consider the averaging with linear regression to be the approach that could further enhance model accuracy. For networks with multiple features, we were not able to improve the results compared to linear regression.
Because using regularized regression, SVR and MLP did not bring improvement over linear regression, except few small improvement with fixed hyperparameters (not the ones we got using grid search), we decided to further use linear regression models. Combining models which were estimated using combination of the tried learning algorithms also did not decrease errors compared to the combination of linear models.
We want to mention that one more technique that we tried was differencing of features similar as for ARIMA models. However, this significantly worsen measured results, so we did not use it in any of the models.

Chapter 5

Stable weather model

In this chapter we describe results achieved by adding model errors from the past as the new features. We also describe approach by which we detect stable weather, normality of errors and their autocorrelation.

Deciding stable weather

Before applying model errors from the past as new features, we first needed to detect when stable weather occurs. We suggest 3 functions that could detect stable weather. In the later text we refer to them as to *s1*, *s2* and *s3*. The functions are defined in Equations 5.1, 5.2 and 5.3. The $predicted_i$ is temperature predicted by Aladin for time i and $temp_i$ is temperature measured at time i.
In the s1 function we declare weather at time i to be stable, if the absolute distance between temperature that was predicted for that hour by Aladin and temperature that was predicted by Aladin for 24 hours before that hour, is not greater than one. The same rule must hold for temperatures at times $i-5, ..., i-1$.
In the s2 function we considered measured temperature instead. We declare weather at time i to be stable if none of measured temperatures at times $i-26, ..., i-22$ differs more than by one from their corresponding temperatures measured 24 hours before.

$$s1(i) = \begin{cases} \text{stable}, & abs(\text{predicted}_j - \text{predicted}_{j-24}) \leq 1 \qquad \forall j \in i-5, ..., i \\ \text{not stable}, & \text{otherwise} \end{cases} \tag{5.1}$$

$$s2(i) = \begin{cases} \text{stable}, & abs(\text{temp}_j - \text{temp}_{j-24}) \leq 1 \qquad \forall j \in i-26, ..., i-22 \\ \text{not stable}, & \text{otherwise} \end{cases} \tag{5.2}$$

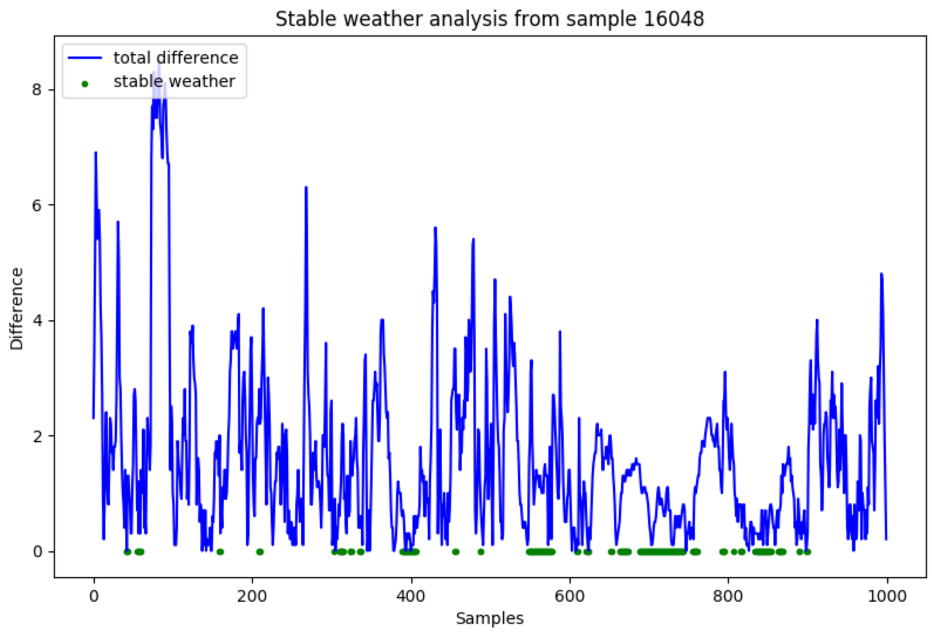


Figure 5.1: Stable weather detection using the s1 function for the reference station for 1000 samples starting from position 16048. The blue line represents absolute distance between temperature measured at given time and 24 hours before. The green line represent times for which we decided stable weather.

$$
\begin{gathered}
n = 5 \\
w = [0.15, 0.6, 1, 0.6, 0.15] \\
h = [-2, -1, 0, 1, 2] \\
ws = \sum w \\
s3(i) = \begin{cases} \text{stable}, & \frac{\sum_{j=1}^{n} abs(\text{temp}_{i-24+j} - \text{temp}_{i-48+j}) * w[j]}{ws} \leq 1 \\ \text{not stable}, & \text{otherwise} \end{cases}
\end{gathered}
\tag{5.3}
$$

Finally, in the s3 we use similar approach as in the s2, but we weight the distances according to Gaussian weights, sum them up and norm by the sum of those weights. If the result is less or equal to one we declare the weather to be stable.

The s1 function has the advantage compared to the s2 and s3 functions, that it does not need to look too far to the past because it uses predicted and not measured temperatures. The disadvantage is that the temperatures are only estimated. The s3 function should tolerate higher deviations from stable weather for more distant hours from target time. The example of using the s1 function is shown in Figure 5.1.

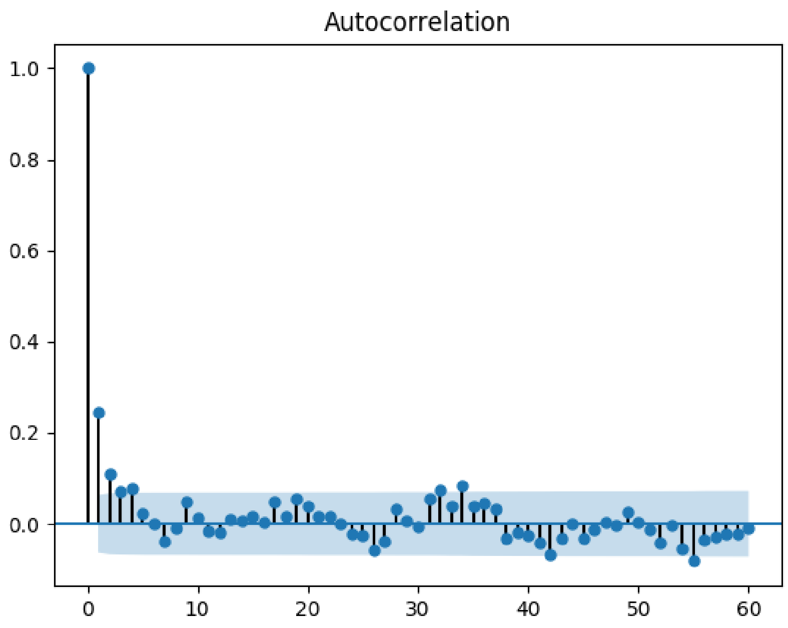


Figure 5.2: Autocorrelation graph for 10PM records using 120-hum model. The dots outside the blue area shows true autocorrelation with 95% confidence level.

Residuals autocorrelation and normality

We investigated residuals of our models on new data to check if they do not follow normal distribution and also tested for autocorrelation. When we refer to errors or residuals of our models we mean the errors that our model made on new data when sliding the rolling window. If we detected no autocorrelation and the residuals were normal, we would not consider adding previous errors useful as the errors could be just white noise. However, for all the models we tried, the p-values for the null hypothesis stating that the model errors come from normal distribution were very close to zero. Therefore, the null hypothesis was rejected. We used Kolmogorov-Smirnov test to test for normality. The histogram of residuals for the 120-hum model is shown in Figure 5.3 and is compared to a curve following normal distribution with mean and standard deviation calculated from the residuals.

The autocorrelation was also seen, but it differed across the models and hours. In Figure 5.2 we can see autocorrelation graph for 120-hum model and temperatures being predicted for 10PM. We can see significant autocorrelation for 1st and also for 2nd lag. As the model errors does not follow normal distribution and autocorrelation was seen in the data for many models, we decided to experiment with including previous model errors in our further models as the new features.

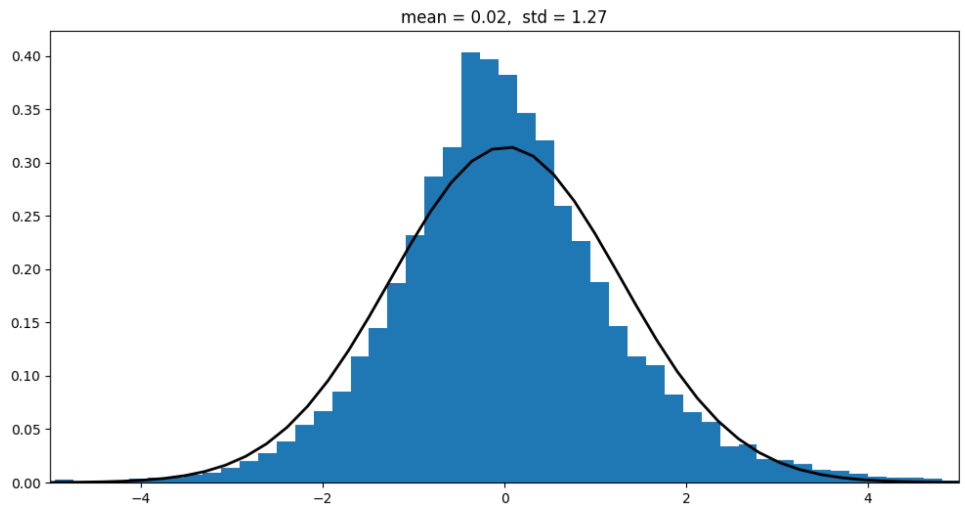


Figure 5.3: Histogram of model errors for 120-hum model against true normal distribution curve with mean and standard deviation calculated from errors.

Autoregressive models

In our thesis we consider *autoregressive models* to be those, where previous model errors are used as the new features. Most straightforward autoregressive approach is probably to provide error that a model made 24 hours before the time for which temperature is being predicted. To the error that our model made in the past we refer as to *autoregressive term*. When combined with the reference model we get the model from Equation 5.4

$$y_i^h = \theta_0 + \theta_1 T_i + \theta_2 P_i^h + \theta_3 (predicted_{i-24}^h - y_{i-24}^h) \qquad h \in 1, ..., 12 \tag{5.4}$$

where $predicted_{i-24}^h$ is the temperature that our model predicted 24 hours before the validity time. We refer to this model as to *auto1*. In the case when the autoregressive term is not available at least for one record in a rolling window, we do not include this term in a training data. Therefore, before using the autoregressive term in the auto1 model the count of predictions that must be predicted without autoregressive term is equal to

$$size(rollingWindow) + 1.$$

Autoregressive models are also sensitive to invalid data records. Invalid records are being skipped and when valid data appears, we might miss some autoregressive terms (due to the skipped records) and can not use those terms for some time. If the invalid data appears too often the autoregressive models can hardly be used.

We evaluated the auto1 model using the rolling window size of 120 on all the data for the reference station. Until now, we have been using only subset of the data for this station to compare with the Hajdin's results. However, the amount of records which are labeled as stable weather records is quite small so using more data helps.

Table 5.1: Compared on 2765 records using s1 as the stable weather function.

Model	ref120	auto1_120
MAE	0.8657	0.8455
MSE	1.2771	1.2145

Table 5.2: Compared on 1972 records using s2 as the stable weather function.

Model	ref120	auto1_120
MAE	0.8786	0.8646
MSE	1.3467	1.3049

Table 5.3: Compared on 5385 records using s3 as the stable weather function.

Model	ref120	auto1_120
MAE	0.9307	0.9193
MSE	1.5135	1.4785

Tables 5.1, 5.2 and 5.3 contain the results of evaluating the auto1 model with different stable weather detecting functions and compared to the reference model. When evaluating the reference model using all data for the reference station MAE and MSE was 0.9568 and 1.6361 respectively. Using any of the tried stable weather detecting functions, both MAE and MSE were lower indicating that during the stable weather Aladin makes more accurate predictions. Using the auto1 model compared to the reference model on the stable weather records helped in case of all stable data functions. Based on the Wilcoxon paired test the improvement was also significant.
For the rest of the thesis we use the s1 function because we consider it most plausible, the amount of records it decides to be stable seemed most realistic from our point of view, and also both MAE and MSE were lowest for those records.

We also tried adding different autoregressive terms and compared those models with the auto1 model. We created *auto2* model which is like the auto1 model, but also consist of autoregressive term for error that our model made 48 hours before. The model is defined in Equation 5.5.

$$y_i^h = \theta_0+\theta_1 T_i+\theta_2 P_i^h+\theta_3(predicted_{i-24}^h-y_{i-24}^h)+\theta_4(predicted_{i-48}^h-y_{i-48}^h) \qquad h \in 1,...,12 \tag{5.5}$$

As applying mean to past temperatures and using it as a predictor improved the results in Chapter 4, we decided to apply mean to errors that our model made 36 up to 24 hours before the time for which we predict. We appended this term to the auto1 model resulting in *auto3* model described in Equation 5.6. We expected that the mean could describe changes in the errors that the model does during that time. We also tried applying variance, but the results were worse compared to the mean, therefore we do not include those models in the thesis.

$$\begin{gathered}y_i^h = \theta_0+\theta_1 T_i+\theta_2 P_i^h+\theta_3(predicted_{i-24}^h-y_{i-24}^h)+\\ \theta_4 mean((predicted_{i-36}-T_{i-36}),...(predicted_{i-24}-T_{i-24})) \qquad h \in 1,...,12\end{gathered} \tag{5.6}$$

In the *auto4* model we use mean from error before 24 up to 12 hours before the time for which we predict. The model is described in Equation 5.7.

$$\begin{gathered}y_i^h = \theta_0+\theta_1 T_i+\theta_2 P_i^h+\theta_3(predicted_{i-24}^h-y_{i-24}^h)+\\ \theta_4 mean((predicted_{i-24}-T_{i-24}),...(predicted_{i-12}-T_{i-12})) \qquad h \in 1,...,12\end{gathered} \tag{5.7}$$

Finally, the *auto5* model use mean from error before 30 up to 18 hours before the time for which we predict, so the that error from 24 hours is in the middle of this interval. The model is described in Equation 5.7.

$$\begin{gathered}y_i^h = \theta_0+\theta_1 T_i+\theta_2 P_i^h+\theta_3(predicted_{i-24}^h-y_{i-24}^h)+\\ \theta_4 mean((predicted_{i-30}-T_{i-30}),...(predicted_{i-18}-T_{i-18})) \qquad h \in 1,...,12\end{gathered} \tag{5.8}$$

From the results in Table 5.4 we see that the only model that gained better results with respect to the auto1 was the auto4 model. However, we can not confirm that this improvement was significant based on the Wilcoxon paired test neither for MAE nor MSE. Later we experiment with the auto1 and auto4 autoregressive models.

Table 5.4: Compared on 2763 records using s1 as the stable weather function and the rolling window size of 120 records.

Model	ref	auto1	auto2	auto3	auto4	auto5
MAE	0.8653	0.8451	0.8499	0.8525	0.8434	0.8499
MSE	1.2766	1.2139	1.2201	1.2286	1.1956	1.2178

We decided to compare the hum model to the auto4 model with humidity included as the next feature to see if the autoregressive terms from the auto4 also improve the results for models with more features. The hum model with autoregressive terms from the auto4 model improved MAE from 0.8398 to 0.8255 and MSE from 1.2067 to 1.1548 when evaluated on stable records determined by the s1 function. Based on the Wilcoxon paired test, the improvement was significant for both error metrics.
The final ensemble that was determined as our best model consisted from the C1, C2 and C3 with included humidity in all those models. In this section we refer to those models as to *C1-hum*, *C2-hum* and *C3-hum*. We therefore compared those models with autoregressive terms from the auto4 model to the original models. We used weight base of 0.97 as it seemed to have the best results. The results can be found in Table 5.5.

Table 5.5: Compared on 2763 records using s1 as the stable weather function and the rolling window size of 120 records. When using * it means that model included autoregressive terms from the auto4 model.

Model	C1-hum	C1-hum *	C2-hum	C2-hum *	C3-hum	C3-hum *
MAE	0.7918	0.7951	0.7839	0.7878	0.7836	0.7864
MSE	1.0920	1.0922	1.0569	1.0606	1.0551	1.0587

Autoregressive terms from the auto4 model worsen the results for all three models as opposed to the reference or hum models. We tried to include only autoregressive term from the auto1 model. The results are stored in Table 5.6.

Table 5.6: Compared on 2763 records using s1 as the stable weather function and the rolling window size of 120 records. When using ** it means that model included autoregressive terms from the auto1 model.

Model	C1-hum	C1-hum **	C2-hum	C2-hum **	C3-hum	C3-hum **
MAE	0.7918	0.7913	0.7839	0.7824	0.7836	0.7814
MSE	1.0920	1.0866	1.0569	1.0525	1.0551	1.0481

By adding autoregressive term from the auto1 model, little improvement can be seen for MAE and MSE for all the models. However, the improvement was not significant according to the Wilcoxon paired test in any case.

Finally, we decided to test whether it is better to use autoregressive terms for the C1-hum, C2-hum and C3-hum models only during stable weather or include them always.

Table 5.7: Compared on 20194 records using s1 as the stable weather function and the rolling window size of 120 records. When using *stable* it means that model included autoregressive term from the auto1 model only when the weather was determined as stable. The *always* means that the autoregressive term was always used.

Model	C1-hum	C1-hum stable	C1 always
MAE	0.9253	0.9253	0.9299
MSE	1.5780	1.5777	1.5914

Table 5.8: Same description as for Table 5.7.

Model	C2 hum	C2 hum stable	C2 always
MAE	0.9178	0.9176	0.9232
MSE	1.5510	1.5506	1.5637

Table 5.9: Same description as for Table 5.7.

Model	C3	C3 hum stable	C3 always
MAE	0.9201	0.9200	0.9249
MSE	1.5585	1.5579	1.5704

When always using the autoregressive terms, the results were worse compared to the case when using them only during stable weather. However, the improvement when using the autoregressive terms only during stable weather is small.
Based on the reference station data, we do not consider autoregressive terms to cause significant improvement to results. They could improve both MAE and MSE for simpler models as the reference and hum, but for the C1-hum, C2-hum and C3-hum models the improvement was minor. The auto4 autoregressive terms seemed appropriate when applied to the reference and hum models, but for later tried models autoregressive term from the auto1 worked better.
Also we tested if shorter rolling windows do not behave better during stable weather, because if a model did similar errors for longer time, the new observation could influence the model behaviour more when the window is shorter. Though the approach only worsen the results, so we rejected it.

Chapter 6

Results for all stations

In this chapter we present final results that we got after evaluating the ensemble2 model using data for all stations we had, except the station with ID 11894 because it contained too much invalid records. To remind, ensemble2 is the ensemble model which averages results of the C1, C2 and C3 models with humidity included as the feature in all of these separate models. We also used autoregressive version of this model where each model in the ensemble also consists of the autoregressive term from the auto1 model. The autoregressive term was included only if weather was declared stable using the s1 function. In this chapter we refer to those models as to *final* and *final** (the model with autoregressive term).
We compared the final and final* models to Aladin predictions. All the models were linear regression models with rolling window length of 120 and the base of the weight function was 0.97. We also used the same weight base for all the models we compare in this chapter.
We conclude that using the autoregressive term did help for the final model. When using this term improvement for MAE occurred in 19 cases and in 10 cases the results were worse. However, the improvement was very minor. In case of MSE improvement occurred for 15 stations and results were worse for 13 stations. We therefore do not recommend using the autoregressive term for this model as it only introduces more complexity and very minor improvement. On the other hand, we recommend using it for simpler models such as the reference or hum.
To sum up, we could improve the results for all the stations compared to Aladin predictions. The greatest improvement was gained for the stations located around 2000 meters above sea level. Those are called *Lomnický štít* and *Chopok*. For the final model the improvement in MAE ranged from 18.95% to 73.61% and the median improvement was 44.08%. The improvement for MSE ranged from 27.60% to 89.44% and the median improvement was 58.34%.

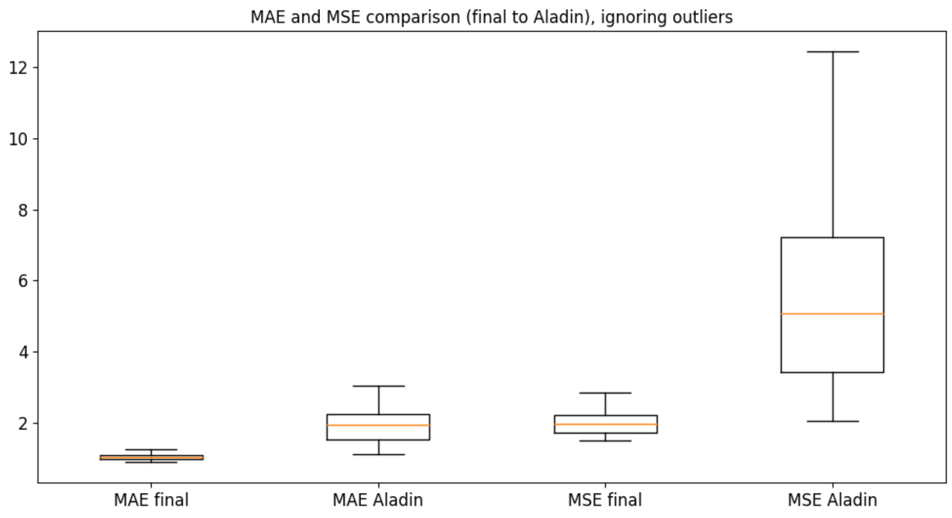


Figure 6.1: Boxplots showing ranges of MAE and MSE for Aladin and the final model. The outliers were removed for better readability.

It is difficult to directly compare to the Hajdin's results as the data that were used for evaluation differ. Also we could not replicate the results for the final model that was used in the Hajdin's thesis. Nevertheless, the author claimed that the improvement in MAE was from 13% to 57%, thus we consider our results to introduce significant improvement.

To better visualize the improvement we attach histograms of MAE and MSE improvement gained with the final model in Figures 6.2 and 6.3. The different ranges of MAE and MSE for the final model and Aladin predictions are also visualized using boxplots in Figure 6.1.

To explore the improvement that the final model introduced to our baseline model, we compared it to the reference model. The improvement ranged between 2.15% up to 5.65% for MAE and between 1.67% up to 9.68% for MSE. The final model improved those metrics for all the stations compared to the reference model. Median of the improvement was 3.38% for MAE and 5.29% for MSE. Histograms in Figures 6.5 and 6.6 show distribution of the improvement. We also attach boxplots of MAE and MSE values in Figure 6.4.

To investigate whether adding humidity influenced lower errors, we decided to compare the reference and hum models. The hum model errors were lower for all the stations and the improvement ranged between 0.91% up to 3.73% for MAE and between 0.34% up to 4.92% for MSE. Median of the improvement was 1.91% for MAE and 2.56% for MSE. The improvement for MAE is shown in Figure 6.7.

The final results for the reference, final and final* models are captured in Tables 6.1, 6.2 for MAE and MSE respectively.

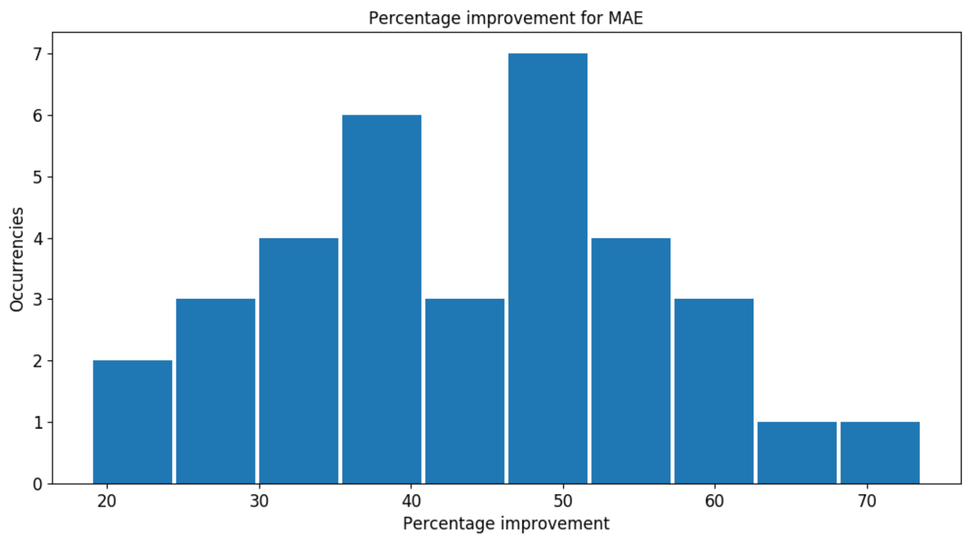


Figure 6.2: MAE percentage improvement gained with the final model compared to Aladin predictions.

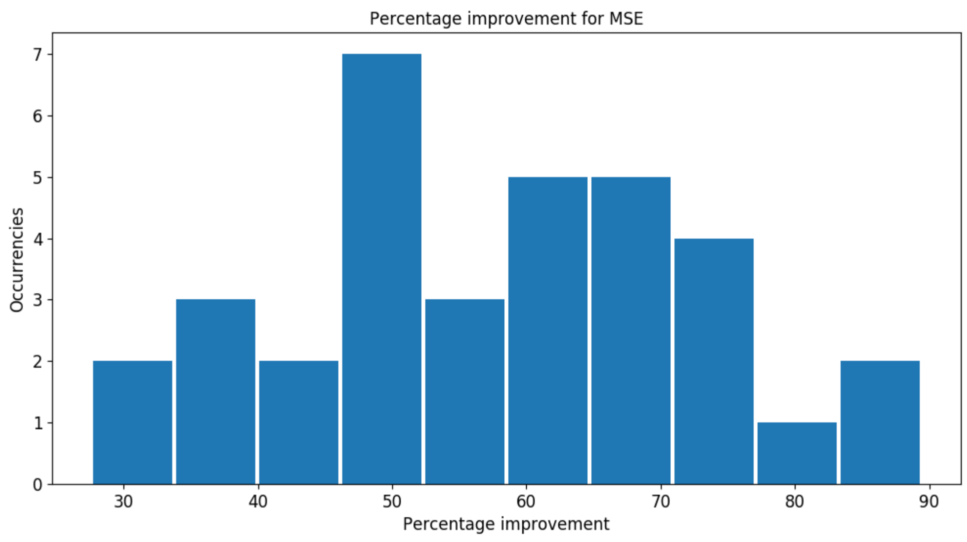


Figure 6.3: MSE percentage improvement gained with the final model compared to Aladin predictions.

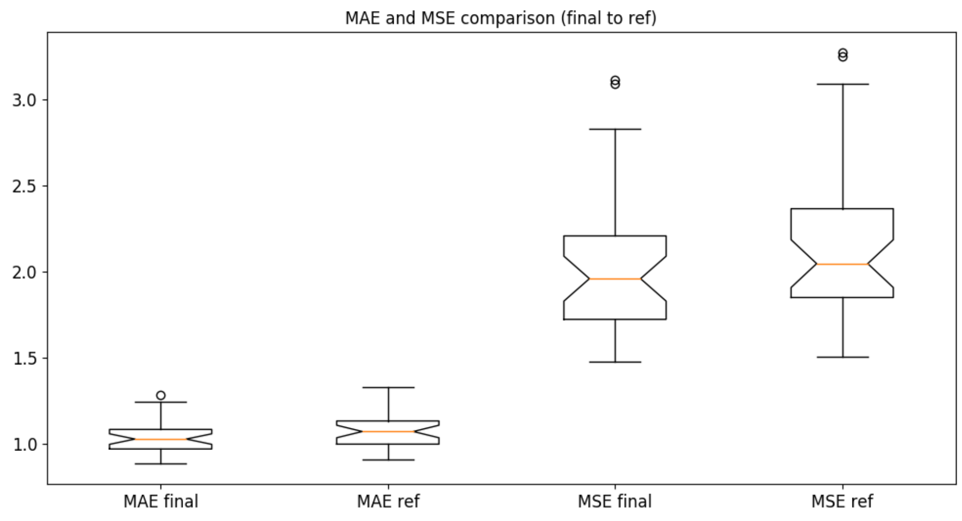


Figure 6.4: Boxplots showing ranges of MAE and MSE for the reference and final models.

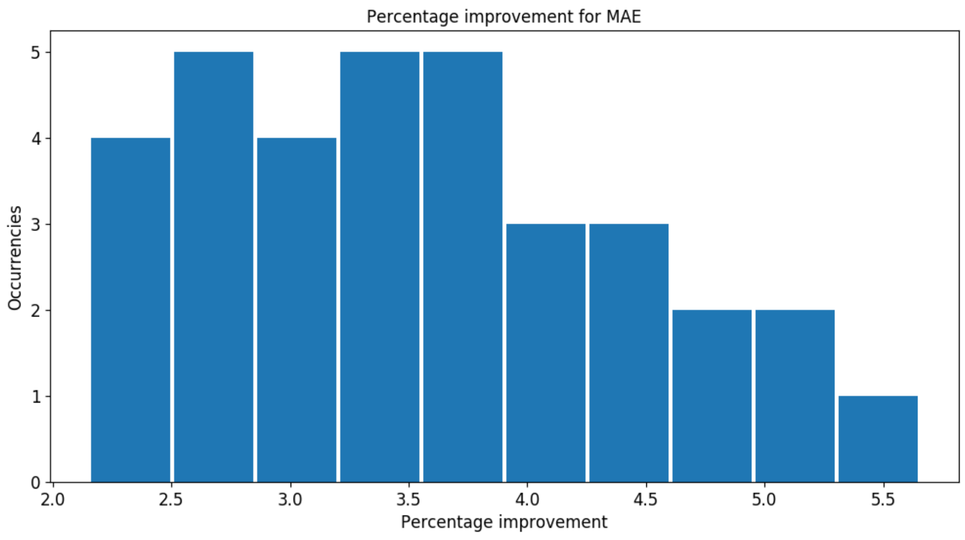


Figure 6.5: MAE percentage improvement gained with the final model compared to the reference model.

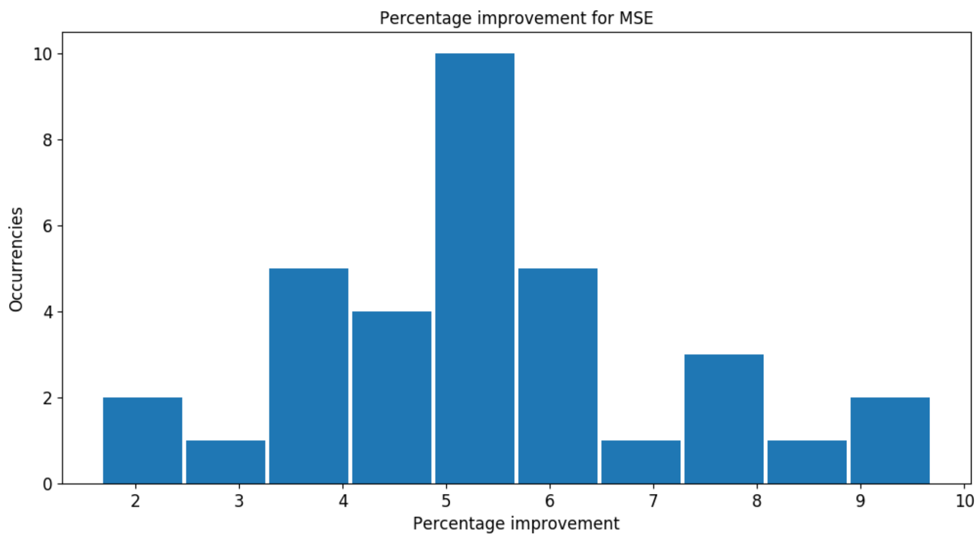


Figure 6.6: MSE percentage improvement gained with the final model compared to the reference model.

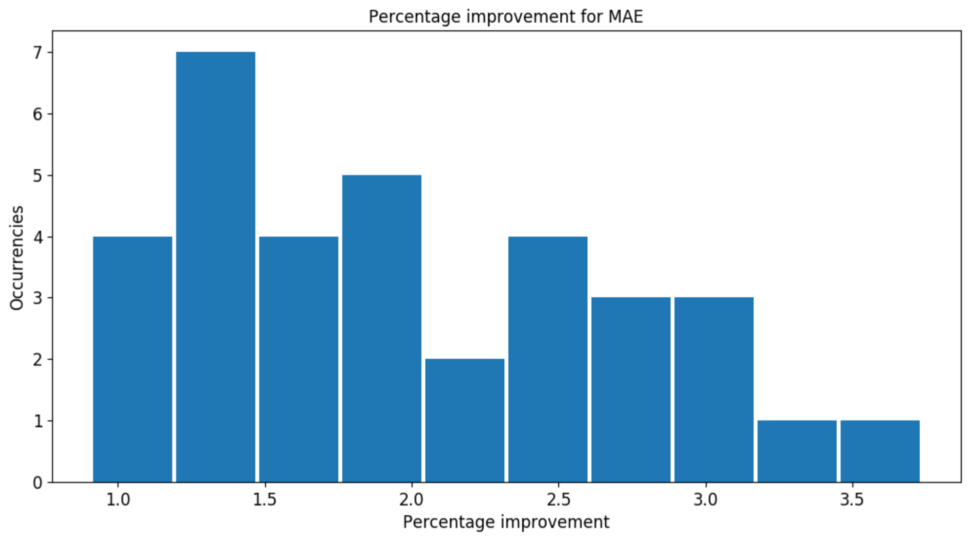


Figure 6.7: MAE percentage improvement gained with the hum model compared to the reference model.

Table 6.1: MAE for all stations.

Station	*Records*	*ref*	*final*	*final**	*Aladin*
Kuchyňa	16050	1.1524	1.1225	1.1224	1.7636
Trenčín	8913	1.0358	1.0093	1.0094	1.5941
Senica	11225	0.9549	0.9304	0.9299	1.4937
Malý Javorník	5966	0.9782	0.9229	0.9229	1.5937
Bratislava-Koliba	23099	0.9068	0.8863	0.8865	1.0935
Bratislava-Letisko	23099	0.9438	0.9048	0.9047	1.2525
Jaslovské Bohunice	23099	0.9779	0.9415	0.9413	1.3080
Piešťany	17225	1.0719	1.0371	1.0372	1.5626
Veľké Janíkovce	19842	1.0090	0.9728	0.9728	1.4916
Mochovce	22760	0.9250	0.8944	0.8939	1.2120
Turzovka	18462	1.1341	1.0866	1.0867	2.1727
Hurbanovo	23099	0.9685	0.9248	0.9253	1.4007
Prievidza	18423	1.0795	1.0532	1.0529	2.5166
Liptovský Mikuláš	16899	1.3065	1.2424	1.2414	2.0274
Dudince	21112	1.0755	1.0268	1.0268	1.8313
Žiar nad Hronom	18822	1.0715	1.0283	1.0281	2.2467
Sliač	21972	1.0830	1.0359	1.0353	2.0135
Chopok	16846	0.9949	0.9700	0.9696	3.0407
Brezno	8060	1.2442	1.1823	1.1831	2.2238
Liesek	20135	1.0785	1.0448	1.0439	1.3369
Oravice	12390	1.3324	1.2865	1.2861	2.6875
Lučenec	20105	1.0077	0.9700	0.9695	1.5608
Lomnický štít	23099	1.1301	1.0776	1.0769	4.0834
Štrbské pleso	14817	1.1384	1.1031	1.1034	2.7813
Poprad	8064	1.2254	1.1990	1.1990	2.1499
Telgárt	23099	1.0431	1.0149	1.0153	2.4320
Gánovce	23099	1.1188	1.0824	1.0826	2.1235
Kojsovská hoľa	5872	1.1146	1.0842	1.0842	2.4577
Bardejov	8885	1.1388	1.0806	1.0799	2.0567
Jakubovany	10673	1.0161	0.9854	0.9845	1.9434
Košice-letisko	19469	0.9695	0.9399	0.9396	1.5184
Tisinec	20727	1.0618	1.0222	1.0223	1.9237
Trebišov-Milhostov	19966	1.0674	1.0271	1.0269	1.5589
Kamenica nad Cirochou	22716	1.1986	1.1592	1.1581	2.5500

Table 6.2: MSE for all stations.

Station	*Records*	*ref*	*final*	*final**	*Aladin*
Kuchyňa	16050	2.4026	2.3212	2.3213	4.3580
Trenčín	8913	1.8732	1.8195	1.8206	3.8031
Senica	11225	1.6244	1.5932	1.5924	3.2649
Malý Javorník	5966	1.8674	1.7189	1.7189	3.9574
Bratislava-Koliba	23099	1.5039	1.4788	1.4799	2.0425
Bratislava-Letisko	23099	1.5930	1.5134	1.5131	2.4843
Jaslovské Bohunice	23099	1.6758	1.6000	1.6000	2.8711
Piešťany	17225	2.0497	1.9787	1.9815	3.6042
Veľké Janíkovce	19842	1.7921	1.6978	1.6978	3.3844
Mochovce	22760	1.5464	1.4827	1.4818	2.4307
Turzovka	18462	2.3403	2.2157	2.2188	6.6629
Hurbanovo	23099	1.6688	1.5728	1.5750	3.1241
Prievidza	18423	2.0275	1.9574	1.9586	8.4315
Liptovský Mikuláš	16899	3.0926	2.8320	2.8304	6.1616
Dudince	21112	2.0703	1.9205	1.9219	4.8315
Žiar nad Hronom	18822	2.0162	1.9088	1.9088	7.1754
Sliač	21972	2.1261	2.0108	2.0094	5.9388
Chopok	16846	2.0140	1.8615	1.8612	12.4447
Brezno	8060	2.8933	2.6131	2.6132	7.2302
Liesek	20135	2.0748	1.9812	1.9781	3.0161
Oravice	12390	3.2751	3.1144	3.1125	9.8747
Lučenec	20105	1.9495	1.8514	1.8500	3.8088
Lomnický štít	23099	2.4156	2.1879	2.1861	20.7126
Štrbské pleso	14817	2.3729	2.2204	2.2219	10.4482
Poprad	8064	2.7276	2.5729	2.5729	6.4523
Telgárt	23099	1.9977	1.8720	1.8737	7.8438
Gánovce	23099	2.2452	2.1235	2.1238	2.9957
Kojsovská hoľa	5872	3.2513	3.0903	3.0903	10.1062
Bardejov	8885	2.3491	2.1736	2.1723	5.8907
Jakubovany	10673	1.8450	1.7379	1.7369	5.3926
Košice-letisko	19469	1.6718	1.5984	1.5977	3.4764
Tisinec	20727	2.0456	1.9631	1.9659	5.3345
Trebišov-Milhostov	19966	2.0975	1.9690	1.9670	3.8804
Kamenica nad Cirochou	22716	2.5933	2.4935	2.4913	9.0021

Conclusion

In our thesis we focused on improving accuracy of temperature forecasts created by model Aladin. We restricted for twelve hour predictions using the rolling window approach.

Compared to the Hajdin's thesis we changed the rolling window period to 24 hours and showed that using this period it is reasonable to use longer rolling window lengths. We used only features related to temperature and humidity. Using new features created from temperature values, we could further decrease models errors. Improvement graphs were incorporated in the process of choosing the right ensemble model and for analysis of improvement. We created three linear regression models using different features and averaged their predictions. We preserved weighting of predictions, to make newer predictions more significant, same as done in the Hajdin's thesis.

Experimenting with different learning algorithms did not bring expected improvement. Choosing the right hyperparameters was problem for more complex algorithms where we suffered from underfitting or overfitting and could not reach better results than with linear regression. We showed that when linear regression and GRU recurrent networks were averaged, the results were slightly better compared to the linear model. However, the learning of recurrent neural networks is much more time consuming.

We focused on conditions when temperature was stable and tried the models which used their past errors during stable conditions. Stable weather was detected based on the difference between predicted temperatures within two days. Past error from 24 hours before and mean of past errors improved results for simpler models. For models with multiple features the improvement was minor or none. We therefore propose to use past error terms only for simpler models.

MAE improvement across all stations varied between 19% to 74% that is higher than improvement achieved in the Hajdin's thesis which was between 13% to 57%. We constructed the final model using only data from one station to have better assumptions about generalization of the improvement. In practise, we would expect that constructing a separate model for each station might introduce further improvement.

Bibliography

[1] Keras documentation. Available from https://keras.io/.

[2] Scikit-learn documentation. Available from http://scikit-learn.org/stable/.

[3] Understanding lstm networks. Available from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[4] Bernhard Scholkopf Alex J. Smola. A tutorial on support vector regression, September 2003.

[5] Mohamed Aly. Survey on multiclass classification methods, November 2005.

[6] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[7] A.H. Nury et al. Time series analysis and forecasting of temperatures in the sylhet division of bangladesh. In *4th International Conference on Environmental Aspects of Bangladesh*, August 2013.

[8] Ales Farda et al. Model aladin as regional climate model for central and eastern europe. *Studia Geophysica et Geodaetica*, 2:i313–i323, May 2010.

[9] Chih-Wei Hsu et al. A practical guide to support vector classification, 2003.

[10] F. Chevalier et al. Support vector regression with reduced training sets for air temperature prediction: a comparison with artificial neural networks. *Computers and Electronics in Agriculture*, 68:i52–i61, August 2009.

[11] Junyoung Chung et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. Workshop on Deep Learning, 2014.

[12] Kira Feldmann et al. Spatial postprocessing of ensemble forecasts for temperature using nonhomogeneous gaussian regression, March 2015. Available at http://journals.ametsoc.org.

[13] Quoc V. Le et al. A simple way to initialize recurrent networks of rectified linear units, 2015.

[14] Kira Feldmann. Statistical postprocessing of ensemble forecasts for temperature: The importance of spatial modeling. Master's thesis, Ruprecht-Karls-Universitat Heidelberg, Fakultat fur Mathematik und Informatik, 2012.

[15] Hoogenboom G. The georgia automated environmental monitoring network, 1993.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[17] Michal Hajdin. Numerical weather prediction postprocessing. Master's thesis, Faculty of Mathematics, Physics and Informatics of Comenius University, 2016.

[18] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):i1735–i1780, 1997.

[19] John H. McDonald. *Handbook of biological statistics, Third Edition.* Sparky house publishing, 2014.

[20] Andrew Ng. Slides to Machine learning course, referred February 23, 2018, available from `https://github.com/worldveil/coursera-ml/blob/master/linear-regression/Lecture4.pdf`.

[21] Andrew Ng. Support vector machines. Notes for Standford machine learning course, referred January 29, 2017, available from `http://cs229.stanford.edu/notes/cs229-notes3.pdf`.

[22] Eberly College of Science. Multicollinearity and other regression pitfalls. Online course available from https://onlinecourses.science.psu.edu/stat501/node/343.

[23] Srinath Perera. Rolling window regression: a simple approach for time series next value predictions. Available from https://medium.com/making-sense-of-data/time-series-next-value-prediction-using-regression-over-a-rolling-window-228f0acae363.

[24] Sebastian Raschka. About feature scaling and normalization. Available from http://sebastianraschka.com/Articles/2014_about_feature_scaling.html.

[25] Gesine Reinert. Time series. Course material.

[26] Sanford Weisberg. *Applied Linear Regression, Third Edition.* WILEY, 2005.

[27] Shie-Yui Liong Xinying Yu. Forecasting of hydrologic time series with ridge regression in feature space. *Journal of Hydrology*, 332:i290–i302, 2007.

[28] M.Shashi Y.Radhika. Atmospheric temperature prediction using support vector machines. *International Journal of Computer Theory and Engineering*, 1, April 2009.

[29] Alessandro Rinaldo Yuval Nardi. Autoregressive process modeling via the lasso procedure, 2011. Carnegie Mellon University.