Delhi Climate Forecasting

Liam Glennie, Clara Molins

May 18, 2023

Abstract

The goal of this work is to forecast the temperature in the city of Delhi, India. The analysis is based on a dataset found in Kaggle named *Daily Climate time series data*, available at this link. AR, SARIMA, ANN, and LSTMs are studied with ANNs proving to be best in time efficiency and minimizing forecasting error.

1 Introduction

This paper presents a detailed univariate time-series analysis of the temperature attribute of the *Daily Climate time series data* dataset provided in this Kaggle link which was collected from Weather Underground API. This sample has four attributes, namely *meantemp*, *humidity*, *wind_speed*, and *meanpressure*, with daily data from 1st January 2013 to 24th April 2017 from the city of Delhi.

First, before performing data splitting, exploratory data analysis was performed on all the attributes. In particular, univariate and bivariate analyses were performed. The univariate analysis mainly consisted of finding patterns through time while the bivariate analysis was performed to understand the relationships between attributes even though they were not considered for forecasting purposes.

Afterward, different univariate time-series models such as the Auto-Regressive algorithm, SARIMA, ANNs (Artificial Neural Networks), and LSTMs (Long-short Term Memory) were applied with the goal of forecasting the *meantemp* attribute. Different hyperparameters were studied for both ANNs and LSTMs through grid searches.

As this work was performed for learning purposes only the *meantemp* attribute was analyzed. However, it is planned to perform further multivariate time-series forecasting in a forthcoming paper.

2 The Dataset

As mentioned in the introduction, this work is based on a Kaggle dataset with daily data on temperature, humidity, wind, and pressure in Delhi, India between 1/01/2013 and 14/04/2017. A brief description of the attributes can be found below.

- Mean temperature (*meantemp*): This is the mean temperature averaged out from multiple 3-hour intervals in a day. The unit measure is degrees Celsius.
- Humidity (*humidty*): Relative humidity value of the day. It is measured in %.
- Wind Speed(*wind_speed*): This attribute indicates the average wind speed measured in km/h.
- Mean pressure (*meanpressure*): Pressure reading of weather. It is measured in mbar (millibars).

Note that the provided dataset was already divided into training and testing datasets. However, in time-series analysis, the whole dataset (including the evaluation and testing datasets) is processed and treated before performing analysis. For that reason, it was decided to join them, obtaining a complete unique sample. Note that this was also useful to later apply a three-way data split with the desired proportions.

3 Preprocessing

Both of the provided datasets contained a row corresponding to the day 1/01/2017, and it was decided to keep only the one in the test sample as the values of the training one seemed to be either filler values or unrealistic. For example, knowing that all the features are means, it is unlikely that they all be exact numbers or that the wind speed is zero, which was the case in the training sample.

Furthermore, the dataset did not contain any NAs nor any additional duplicates, therefore little preprocessing was applied. In fact, only some extreme outliers from the pressure attribute were removed, set to NA, and imputed using KNN (k-Nearest Neighbor).

Figure 1 visually exhibits the extreme values encountered for each attribute in the dataset. The first plot shows that the meantemp attribute did not have any extreme values, while the second, third, and fourth plot showed that the rest of the variables did. Particularly, wind_speed and meanpressure.

Regarding humidity and wind_speed it was decided to not remove such values as they are reasonable after all. As can be observed the maximum recorded wind speed is 42,22 km/h, which is not extreme and could be experienced in Delhi. In addition, the minimum humidity value in the sample is 13,42% which is just a bit lower than the minimum relative humidity according to this Indian climate website which states it is 14,9%. For this reason, it was preferred to keep the values.

The only attribute which unrealistic extreme values was *meanpressure*. For this attribute, only average values per month were found, instead of minimum and/or maximum values recorded. According to this website, the minimum average pressure per month is 996.9 mbar. The maximum pressure recorded is 1083.8 mbar according to Arizona State University. Given these values, it was decided to set to null all values out of the following range: 980-1084 mbar.

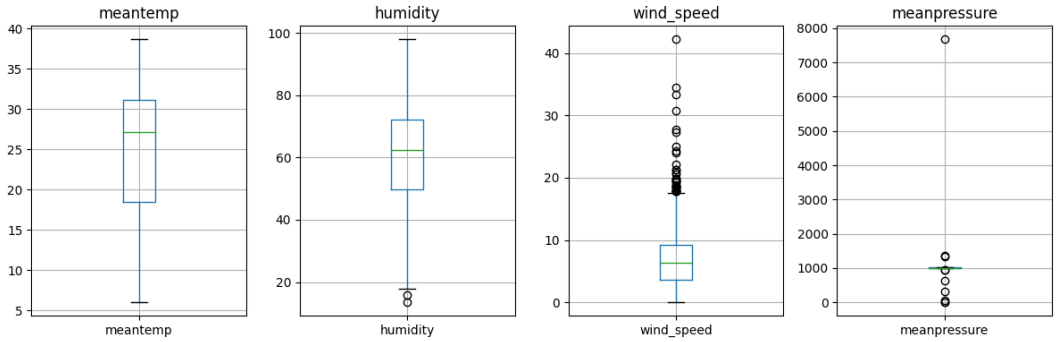

Figure 1: Boxplots of all attributes. From right to left: meantemp, humidity, wind_speed, and meanpressure.

After the values were set to null, as mentioned above, the KNN algorithm was used to impute them. This algorithm works by finding the most similar observations to the ones with missing values. This makes it a good candidate for imputing missing data in time series, where nearby observations are likely to be more similar than those that are far apart.

Interpolation was also considered as a possible imputation method, but in contrast to KNN, it cannot handle non-linear patterns. For such reason, KNN was chosen.

4 Exploratory Data Analysis

The following subsections present the results obtained from performing univariate and bivariate analysis.

4.1 Univariate analysis

First, temporal plots were created to visualize possible trends and patterns through time for each attribute. Note that these plots were all performed after the data was processed.

Figure 2 shows the evolution of mean temperature, humidity, wind speed, and pressure respectively.

This figure shows, as expected, that temperature has a yearly periodic pattern. In winter, the temperature drops, then it rises through spring, and peaks in summer. Afterward, it falls through autumn until it hits a minimum in winter, and the cycle begins again.

Interestingly, humidity values also seem to follow a pattern. In fact, an 'M' shape can be observed four times in the corresponding plot. In this case, the lowest values are usually recorded in April and the highest in July and February every year. Nonetheless, the variance is higher for this attribute than temperature.

Regarding, wind speed, some waves can also be observed. These waves seem to follow a similar behavior to the ones observed in temperature. Wind speed seems to be generally lower in winter, rising through spring, and decreasing after summer.

Finally, the last plot shows that pressure follows the opposite behavior of temperature, and in consequence, it also follows a yearly periodicity. This can be better observed in Figure 3 where both attributes have been normalized to be able to compare them.

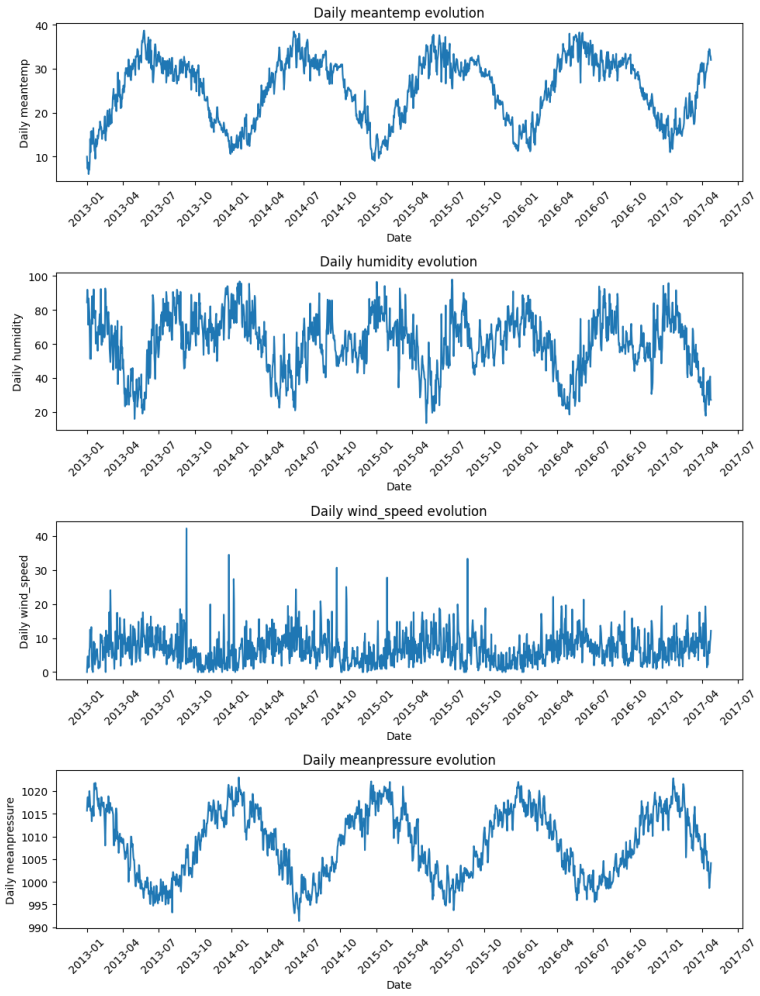

Figure 2: Temporal evolution of all attributes. From top to bottom: meantemp, humidity, wind_speed, and meanpressure.

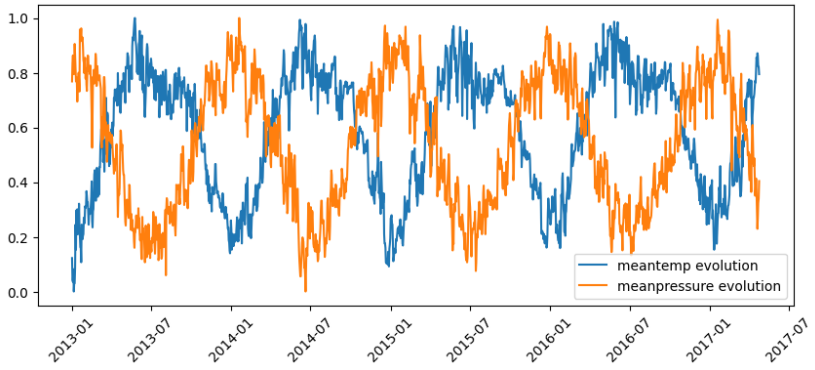

Figure 3: Comparison of meantemp and meanpressure

Other plots were also performed to visualize yearly differences within each attribute. In order to better observe the differences between years and avoid creating plots with a lot of overlapped lines, the plots in Figure 14 in the Annex show the evolution over time but for all the days in a week, the value is set to be the average of all the values in that week. This way, the plot is smoothed and the differences between years are more recognizable.

However, as the differences were still hard to distinguish it was decided to create the following Tables, which showcase, per each attribute and year, the minimum and maximum values, the average and standard deviation. When creating these tables, data from 2017 was not considered due to its incomplete (i.e. there was only data available from the first trimester). A stat that stands out is the increase in average and minimum temperature over the years which is likely due to climate change.

Year	2013	2014	2015	2016
Mean	24.79	25.01	25.11	27.10
STD	7.41	7.60	7.24	6.89
Min	6.00	9.00	9.63	11.19
Max	38.71	38.50	37.75	38.27

Table 1: Statistics of meantemp

Year	2013	2014	2015	2016
Mean	63.05	59.77	61.43	58.74
STD	18.19	16.22	15.69	16.51
Min	15.86	20.88	13.43	18.47
Max	94.00	96.86	98.00	94.30

Table 2: Statistics of humidity

Year	2013	2014	2015	2016
Mean	6.83	6.76	6.48	7.16
STD	4.83	4.66	4.70	3.99
Min	0.00	0.00	0.00	0.00
Max	42.22	30.69	33.33	22.10

Table 3: Statistics of wind_speed

Year	2013	2014	2015	2016
Mean	1007.64	1008.35	1008.83	1008.12
STD	7.75	7.64	7.43	6.86
Min	993.25	991.38	993.75	995.56
Max	1021.80	1023.00	1022.00	1021.14

Table 4: Statistics of meanpressure

4.2 Bivariate analysis

In order to understand the relationships between the different variables, a correlation matrix, scatter plots, and cross-correlation plots were performed between all the attributes.

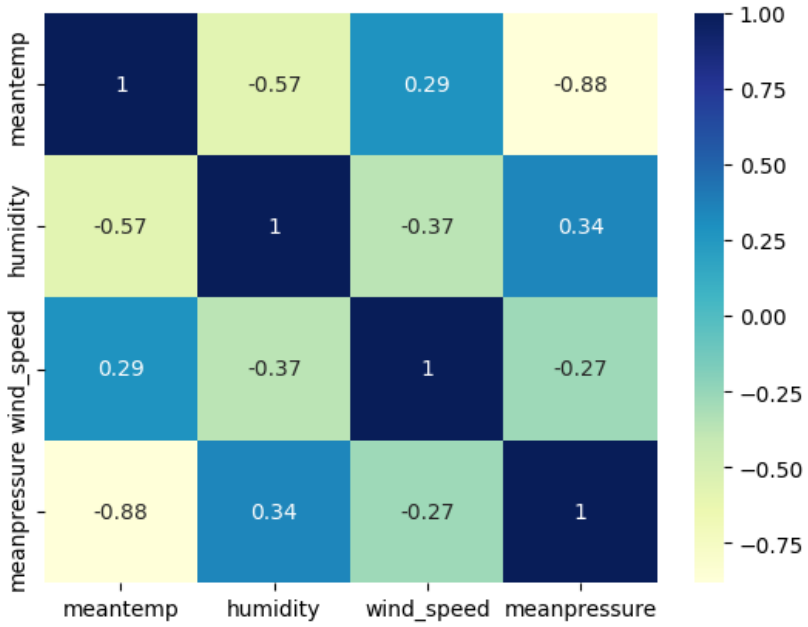

Figure 4: Correlation matrix

As previously observed when performing univariate analysis, Figure 4 shows a high negative correlation between *meantemp* and *meanpressure*. In fact, they have an 88% of correlation.

In addition, it also shows that there is a considerable negative autocorrelation between *meantemp* and *humidity*; 57% to be precise.

Between the rest of the attributes, the correlation is lower. In particular, the minimum correlation is 27%, between *meanpressure* and *wind_speed*, which is not negligible.

The scatterplots in Figure 5 visually show these correlations. However, it is also interesting to see that humidity values follow an almost normal distribution, and that wind speed follows a logarithmic distribution.

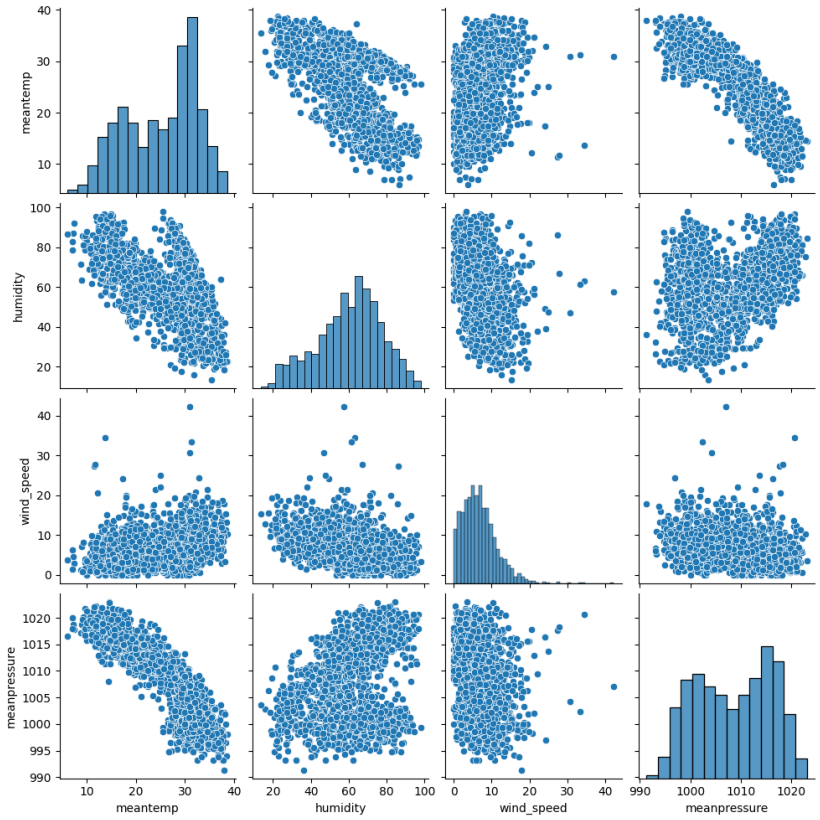

Figure 5: Scatter plots between all variables.

5 Data Split

In order to divide the data for training, evaluating, and testing the models, the three-way split was followed. In particular, the first 70% of the days were kept for training, the following 15% for evaluation, and the remaining 15% for testing.

6 Models

The following subsections explain the different algorithms applied. As mentioned earlier, this study solely focuses on univariate time-series models, therefore only considers the mean temperature (i.e. *meantemp*). Future work will tackle the challenge of multivariate time-series models.

6.1 Auto-Regression

The Auto-Regressive (AR) algorithm is a popular time-series algorithm that works on the basis of using previous data to predict future values of the forecast. It models a time series as a linear combination of p previous values, where p represents the lag order. An AR model can be expressed as:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t \tag{1}$$

where y_t is the value at time t, c is a constant term, ϕ_1 through ϕ_p are the coefficients for the p previous values of y, and ε_t is the error term at time t.

The are four aspects that the AR algorithm assumes about the data being fed to it. The first is stationarity, meaning that the mean, variance, and autocorrelation structure of the data do not change over time. An Augmented Dickey-Fuller (ADF) test was applied to check the stationarity of the data.

The null hypothesis of an ADF test is that the data is not stationary. Thus, a p-value below 0.05, assuming a 95% confidence level, would reject the null hypothesis, meaning that the data can be considered to be stationary. After performing this test, a p-value of 0.27 was obtained, meaning that the null hypothesis fails to be rejected. Therefore, the data is not stationary.

The second assumption is no autocorrelation, meaning that the different values of the time series should not show correlation. Figure 6 illustrates that there is autocorrelation in the data, given that all points can be found outside the blue shadowed area.

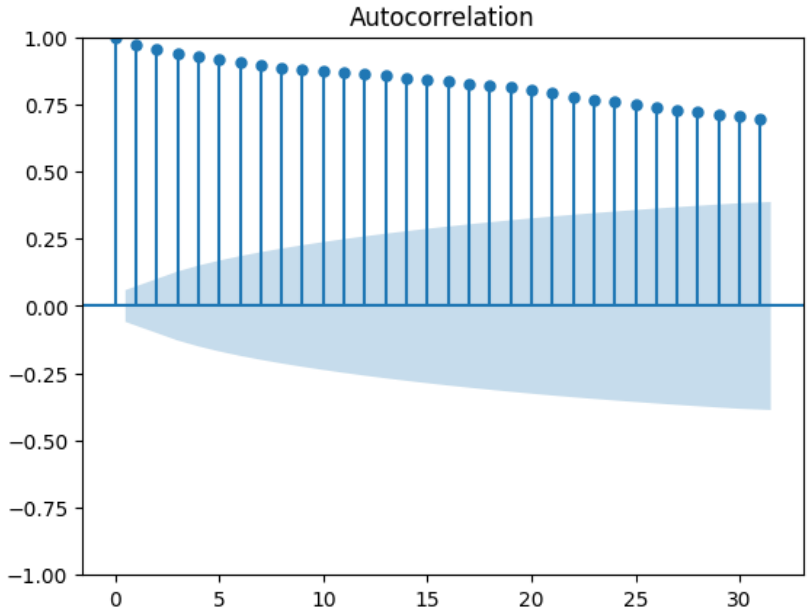

Figure 6: Autocorrelation plot.

Next, the AR algorithm assumes that there is no seasonality in the data, meaning that the mean, variance, and autocorrelation structure over time do not change. Figure 7 displays that the data shows a clear periodic pattern where the mean and variance change over time.

The final assumption of the data is no multicollinearity which cannot occur in a univariate dataset.

To solve all three of the failed assumptions, differencing has been applied to the data, a technique that calculates the difference between consecutive observations. Figure 7 displays the changes differencing has made to the data.

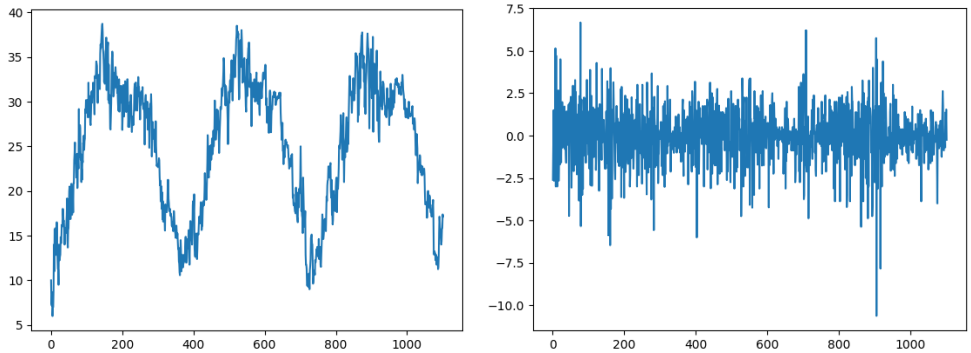

Figure 7: Mean temperature before and after applying differencing.

Note that since differencing has been applied, the AR models trained will predict the difference in

mean temperature of the day before. Thus, metrics like MSE and RMSE will have a lower scale than if the actual mean temperature were predicted.

As previously mentioned, the AR algorithm fits a linear model to predict the next time step using p time steps. Therefore, the model is initially fitted with the training data. Then, during validation, once a prediction is made of a time step, its true value is added to the training set to fit the model to predict the next time step. The upcoming subsections explain how different time step sizes were used and their outcome.

6.1.1 Daily

In the case of daily predictions, the model uses historical data to predict the difference in mean temperature of the next day. Once it has been predicted, the true value of the difference is added to the historical data to predict the next day. Thus, if the validation set has n days, the model will be fitted $n - 1$ times.

A grid search was applied to study the behavior of the AR algorithm for different lag orders. Table 5 showcases Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) obtained. Lag order 15 yields the best results, although the variance between different lags is negligible. Again, it should be noted that the predictions represent the difference in temperature between days and not the actual mean temperature. This results in the scale of the errors being reduced.

Lag Order	1	3	7	15	30	60
MSE	3.32	3.16	3.09	3.01	3.06	3.05
RMSE	1.82	1.78	1.76	1.74	1.75	1.75

Table 5: Results for different lag orders.

Figure 8 visually shows the predictive abilities of the AR model trained with lag order 15. The prediction is colored in orange while the real values are in blue. As can be noticed, the model is able to follow the trends, however, it has difficulties predicting extreme variations.

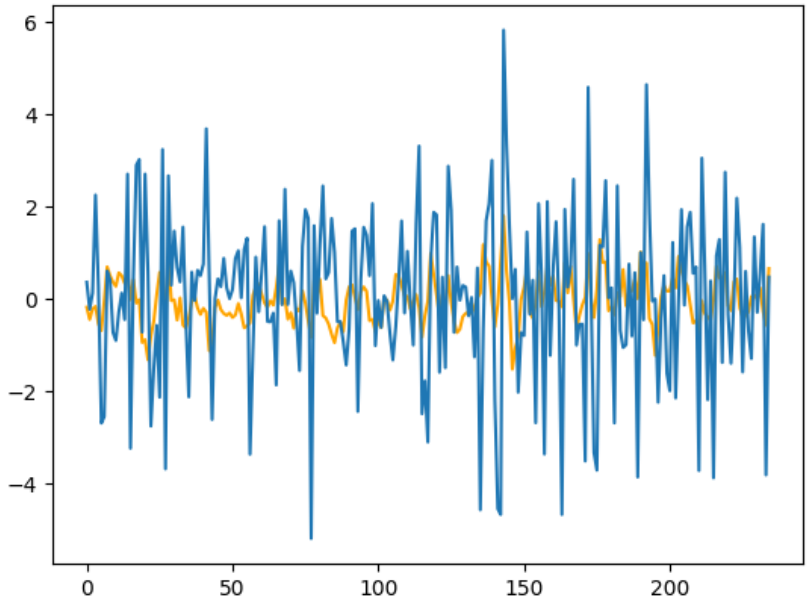

Figure 8: Mean temperature predictions with AR trained daily.

6.1.2 Weekly

When it comes to the weekly version, the model predicts a seven-day forecast. So, like the daily model, it uses historical data to produce said forecast, and then the true values of these seven days are added to the history to predict the following week. Like, the daily models, the data is the difference in mean temperature between days. Table 6 shows the results in MSE and RMSE of different lag orders where the difference between lags is minimal. Note that the optimal results are obtained, again, with log order 15.

In addition, it is worth noting that the results obtained with this model were worse than with the daily trained model.

Lag Order	1	3	7	15	30	60
MSE	3.29	3.18	3.20	3.16	3.23	3.24
RMSE	1.81	1.78	1.79	1.78	1.80	1.80

Table 6: Results for different lag orders.

Figure 9 shows the predictive abilities of the weekly AR model trained with lag order 15. Similarly to what was shown in the previous section, the model is able to follow the trends but it has difficulties predicting extreme variations.

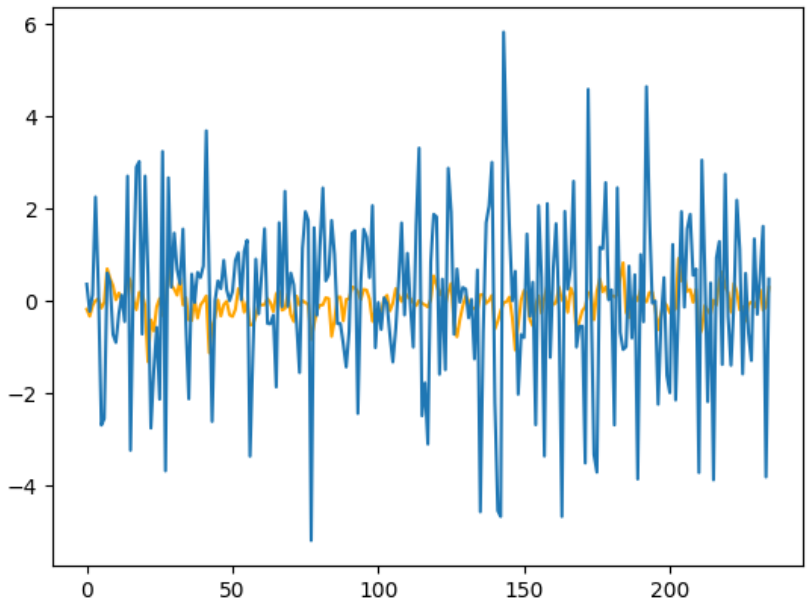

Figure 9: Mean temperature predictions with AR trained weekly with log order 15.

6.2 SARIMA

SARIMA (Seasonal Autoregressive Integrated Moving Average) is another typical statistical model used for time series analysis and forecasting. It is an extension of the ARIMA (Autoregressive Integrated Moving Average) model, which can account for seasonality in the data. In fact, the SARIMA model consists of the following main components:

- Seasonality (S): A repeating pattern that occurs at regular intervals, such as monthly or quarterly.
- Autoregression (AR): A model that predicts future values based on past values of the same variable.
- Integration (I): A process that is used to make a non-stationary time series stationary by taking differences between consecutive observations.

- Moving Average (MA): The model predicts future values based on past forecast errors.

The SARIMA time series model is expressed in a general multiplicative form, denoted as $SARIMA(p,d,q)x(P,D,Q)s$. The model is divided into two parts, where the first part represents the order of the non-seasonal parameters, and the second part represents the order of the seasonal parameters.

In the first part, the lowercase p, indicates the order of non-seasonal autoregression. The lowercase d represents the number of regular differencing, while q represents the order of non-seasonal moving average.

Moving on to the second part, uppercase P denotes the order of seasonal autoregression. The uppercase D signifies the number of seasonal differencing, and Q indicates the order of seasonal moving average. The variable s represents the seasonal length, which varies depending on the frequency of the data. For example, if the data is observed monthly, s would be 12; for weekly data, s would be 52, and for daily data, s would be 365.

It was decided to directly use SARIMA instead of ARIMA because, as previously shown in Section 4.1, the *meantemp* attribute we are trying to predict has an annual seasonality (i.e. periodicity).

Like AR, SARIMA also assumes stationarity within the data. Therefore, for training the model, d was set to one for each of the types of models mentioned below.

6.2.1 Daily

Similarly to the AR section, the SARIMA model is fitted multiple times during validation as it needs to use previous data to predict the next time step. Given that it needs to be trained multiple times and SARIMA is far more complex than AR, only one set of lag order and season duration was applied. No further values for these hyperparameters were investigated as the training time for SARIMA was much longer than ANN, which will be seen further in the study, and it did not bring any performance benefits.

Nonetheless, Table 7 and Figure 10 showcase that SARIMA outperforms AR despite the error being very similar, the scale of the data is different. The error of the SARIMA model is relatively much smaller than the AR models.

Lag Order	Season Duration	MSE	RMSE
1	15	3.37	1.84

Table 7: Results for different lag orders and season durations.

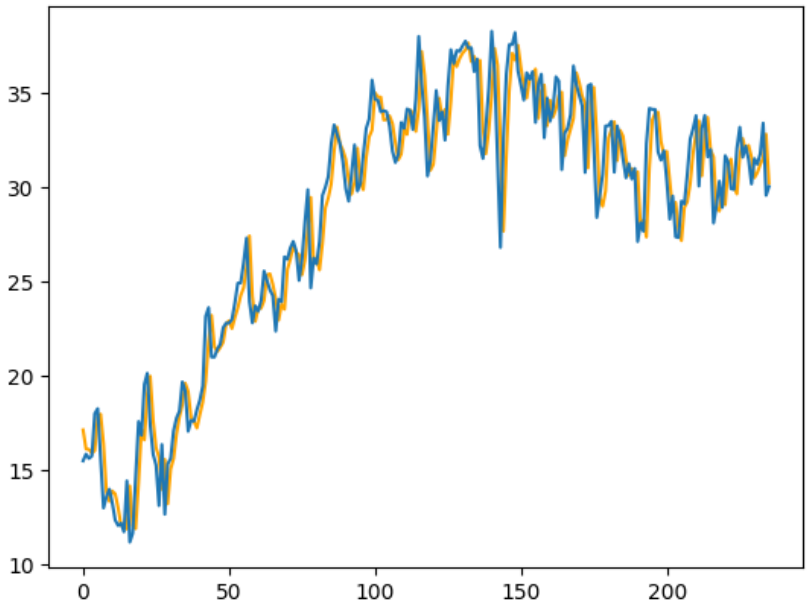

Figure 10: Mean temperature predictions with daily SARIMA model. True data (blue). Prediction (orange)

6.2.2 Weekly

A weekly forecasting version of SARIMA was studied with the goal of reducing training time and potentially finding a trade-off between time and performance. Since the training time is reduced, multiple hyperparameters' values could be tested.

As shown in Table 8, changing the lag order or the season duration did not improve nor worsen the model. In fact, the results are so similar that it is hard to tell that one is better than the rest. Both models trained with season duration 15 and lag order 1 and 7 show the exact same abilities (MSE of 5.32, and RMSE of 2.31). These numbers show that the model is rather good, but the daily one, as mentioned above, is better, especially when it comes to identifying extreme peaks and drops in temperature.

Lag Order	Season Duration	MSE	RMSE
1	15	5.32	2.31
1	30	5.38	2.32
7	15	5.32	2.31
7	30	5.35	2.31

Table 8: Results for different lag orders and season durations.

Figure 11 shows the results obtained by training the model with lag order 1 and season duration 15. As observed, the model is able of forecasting the general trend, however, it fails with large spikes and drops. This is due to the fact that, for all days within a week, it predicts very similar values. Note that the prediction is pictured in orange while the real values are colored in blue.

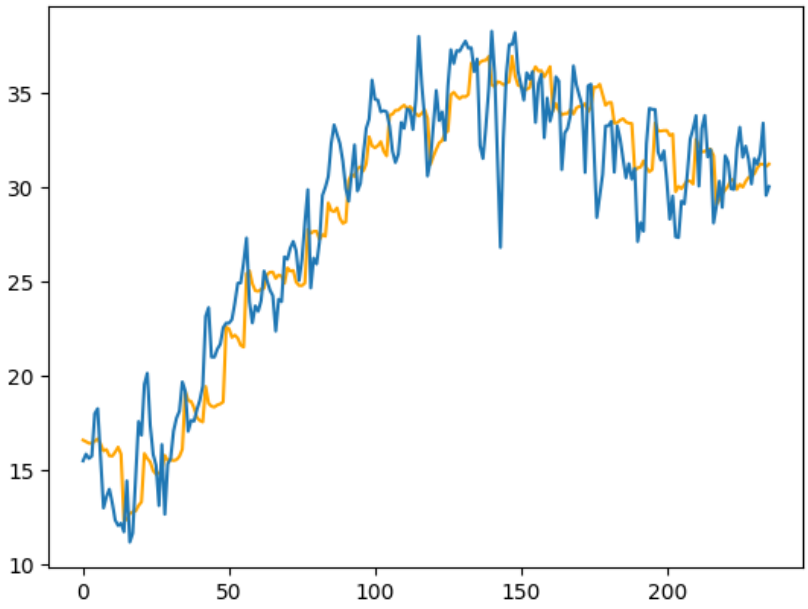

Figure 11: Mean temperature predictions with weekly SARIMA model.

6.3 ANN

Artificial Neural networks (ANN) are architectures that represent how the human brain learns and are used in many applications to automate processes. The typical structure of an ANN is:

- **Node:** A simple data structure that can connect to other nodes via edges.
- **Neuron:** A node in a neural network. A neuron receives one or more weighted inputs and sums them using an activation function to produce an output.
- **Edge:** Link between two neurons that has a weight associated with it.
- **Weight:** A real value that represents the influence one neuron has on the output of another neuron.
- **Activation function:** Calculates the output of each neuron by applying a nonlinear function.

Artificial neural networks consist of three types of layers formed by neurons connected by edges. The input layer is the first layer, where each neuron represents the data fed into the network, such as words, pixels, or numerical values. The hidden layers follow the input layer, and each component in a hidden layer is considered a neuron that calculates an output from its weighted inputs and activation function. There can be as many hidden layers as needed. Deep networks with more layers are preferred over tall networks with more neurons, as deep networks can achieve the same performance using fewer parameters and are more efficient in terms of computation. Also, they can model more complex representations and are better at finding underlying patterns in the data. The output layer is the final layer, which represents the prediction, either probability of each class when doing classification or the predicted value in the case of regression.

ANNs are useful for time series because they can capture complex patterns in the data that traditional statistical methods may struggle with. Additionally, ANNs do not have the constraints of data assumptions like the previously studied AR and SARIMA models.

Unlike AR and SARIMA, ANNs do not natively use the previous time steps to predict the following one. Therefore, a sliding window method was developed to solve this. It accepts the parameter *lookback* which regulates the number of previous time steps taken into account for each input. So, for example, if $lookback = 2$, to predict the value for day_n, the model would receive $[\text{day}_{n-2}, \text{day}_{n-1}]$. For the first three models, a sliding window of three was chosen.

6.3.1 Model Architectures

First, two different simple architectures were evaluated. The first one (Model 1 from now on), consists of two dense layers. The first layer is defined to have 8 neurons and uses the $ReLU$ activation function. Differently, the second dense layer has a single output neuron and no activation function is defined for it, meaning it will output a linear combination of its input values.

In contrast, the second model (Model 2 from now on) is more complex. This model instead of having 2 dense layers, has 4 of them. The first three use the $ReLU$ activation function and use 64, 16, and 4 neurons respectively. Lastly, the fourth layer is equivalent to the last layer of Model 1.

Table 9 clearly shows that Model 1 makes better predictions than Model 2. However, observing the MSE for the training and testing sets, it is also perceived that both models are a little overfitted.

In the following subsection, hyperparameter tuning for Model 1 is studied.

Model	Train RMSE	Validation RMSE
Model 1	1.64	1.83
Model 2	1.87	2.15

Table 9: Results table of Model 1 vs Model 2.

6.3.2 Hyperparameter Tuning

There are multiple hyperparameters that can be tuned for an ANN. A simple model like Model 1 with only two dense layers was considered to perform a grid search on. The hyperparameters tuned were:

- **Batch size**: [1, 2, 4, 8, 32]
- **Epochs**: [5, 10, 20]
- **Activation function**: ['relu', 'sigmoid', 'selu']
- **Number of neurons in first layer:** [4, 8, 16]

The optimal model according to the grid search was a model with a dense layer of 8 neurons, SeLU activation function trained for 20 epochs with a batch size of 1.

Table 10 describes the performance of the model with pretty equal results when it comes to the training and validation scores. The results obtained are very similar to Model 1 which is why different sliding window sizes are considered for both Model 1 and 3.

Model	Train RMSE	Validation RMSE
Model 3	1.64	1.81

Table 10: Results table of Model 3.

6.3.3 Sliding Window Size

Modifications in the sliding window size have shown very small changes in the model's predictive abilities. As Models 1 and 3 were equivalently good, different window sizes were applied to both.

From Table 11 it seems clear that the optimal forecasts are obtained by Model 1 with a sliding window size of 5 days.

Model	Window size	Train RMSE	Validation RMSE
Model 3	3	1.64	1.83
Model 3	5	1.62	1.79
Model 3	7	1.63	1.81
Model 1	3	1.64	1.81
Model 1	5	1.63	1.80
Model 1	7	1.63	1.79

Table 11: Results of Model 1 and Model 3 applying different sliding window sizes.

Figure 12 shows that Model 3 with a sliding window of 5 days is not only capable of correctly forecasting general trends but also both high and low temperature peaks, making it the best model exposed so far in this paper.

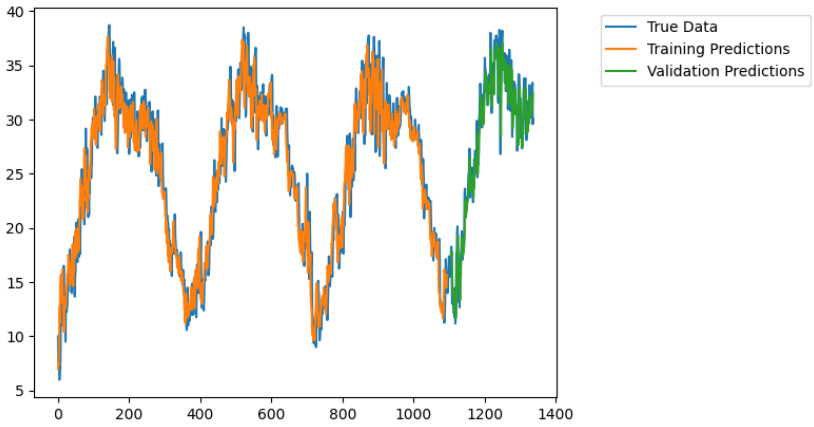

Figure 12: Forecasting results of Model 3 with a sliding window of 5 days.

6.4 LSTM

LSTMs (Long Short-Term Memory) are a type of Recurrent Neural Networks (RNNs). Now, ANNs process each input independently without maintaining a notion of time. On the other hand, LSTMs use recurrent connections to preserve information from previous time steps to use to predict current or future steps.

LSTMs use memory cells that can selectively retain or forget information. These memory cells are controlled by gate mechanisms, which are neural networks that regulate the flow of information to the memory cell. These gates include the input gate, output gate, and forget gate, which are used to control the information flow into the cell, the information flow out of the cell, and the information that is retained or forgotten in the cell, respectively.

This ability to retain past information makes LSTMs suited for time series problems.

6.4.1 Models

This section describes the different models evaluated. Both models have a similar architecture, having only one LSTM layer and a dense output layer. The difference can be found in the number of units in the LSTM layer. Model 4 has 4 LSTM units whereas Model 5 has 16. The rest of the hyperparameters were equal for both models: 30 epochs, a batch size of 1, and a look-back window of 1. Table 12 shows the performance of these two models. Clearly, their ability to correctly forecast temperature is worse than Model 1 with a sliding window size of 5 days. Therefore, there will be no further study on LSTMs for this report.

Model	Train RMSE	Validation RMSE
Model 4	3.05	3.40
Model 5	3.02	3.38

Table 12: Results of Model 4 and Model 5.

7 Model Testing

Considering all the results that have been exposed in the previous sections, it was concluded that in general, Artificial Neural Networks performed the best, with Model 3 with a window size of 5 days, being optimal for this case.

Table 13 and Figure 13 showcase the forecasting abilities of this model where the test RMSE of the model is lower than the training and validation errors. Whilst this indicates that the model generalizes well to new data, the error may be lower because the period it was tasked to predict is less erratic. Figure 13 illustrates that in periods with high temperatures, the behavior is a little more chaotic than in lower temperature periods.

Model	Train RMSE	Validation RMSE	Test RMSE
Model 3	1.62	1.83	1.46

Table 13: Results of Model 3 on testing data.

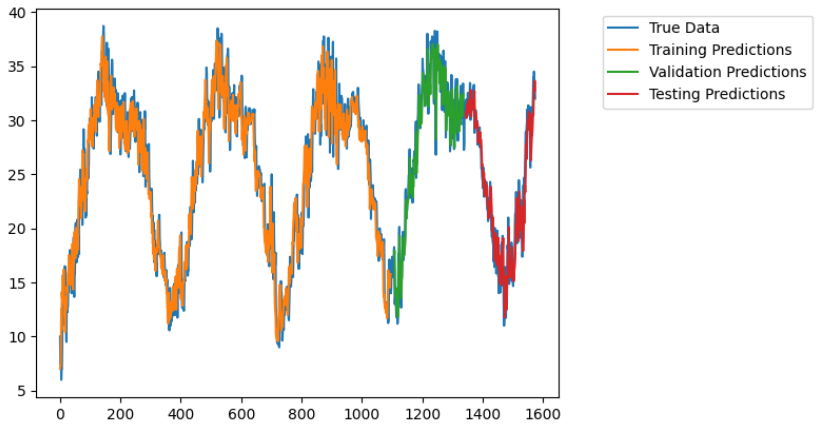

Figure 13: Forecasting results of Model 3 with a sliding window of 5 days.

8 Conclusions

Univariate analysis revealed distinct patterns in the examined attributes:

- The daily mean temperature exhibited a noticeable yearly periodic pattern.
- Humidity values displayed an intriguing 'M' shaped pattern.
- Wind speed followed a similar pattern to temperature, albeit with less prominence.
- Pressure exhibited an inverse behavior compared to temperature, displaying a yearly periodicity.

Bivariate analysis provided further valuable insights:

- The correlation matrix highlighted a significant negative correlation (88%) between *meantemp* and *meanpressure*.
- The cross-correlation plot revealed a considerable negative autocorrelation (57%) between *meantemp* and *humidity*.

- Among the attributes, the minimum correlation (27%) was observed between *meanpressure* and *wind_speed*, which is not negligible.

Regarding temperature forecasting, deep learning methods, like ANN, exhibited superior accuracy and efficiency compared to traditional statistical models such as AR and SARIMA.

Future research could encompass multivariate time-series analysis and exploration of alternative models for forecasting various attributes, including *meantemp*, *humidity*, *wind_speed*, and *meanpressure*.

Given the insights derived from the bivariate analysis, incorporating additional explanatory variables into the models is anticipated to enhance their predictive capabilities.

9 Annex

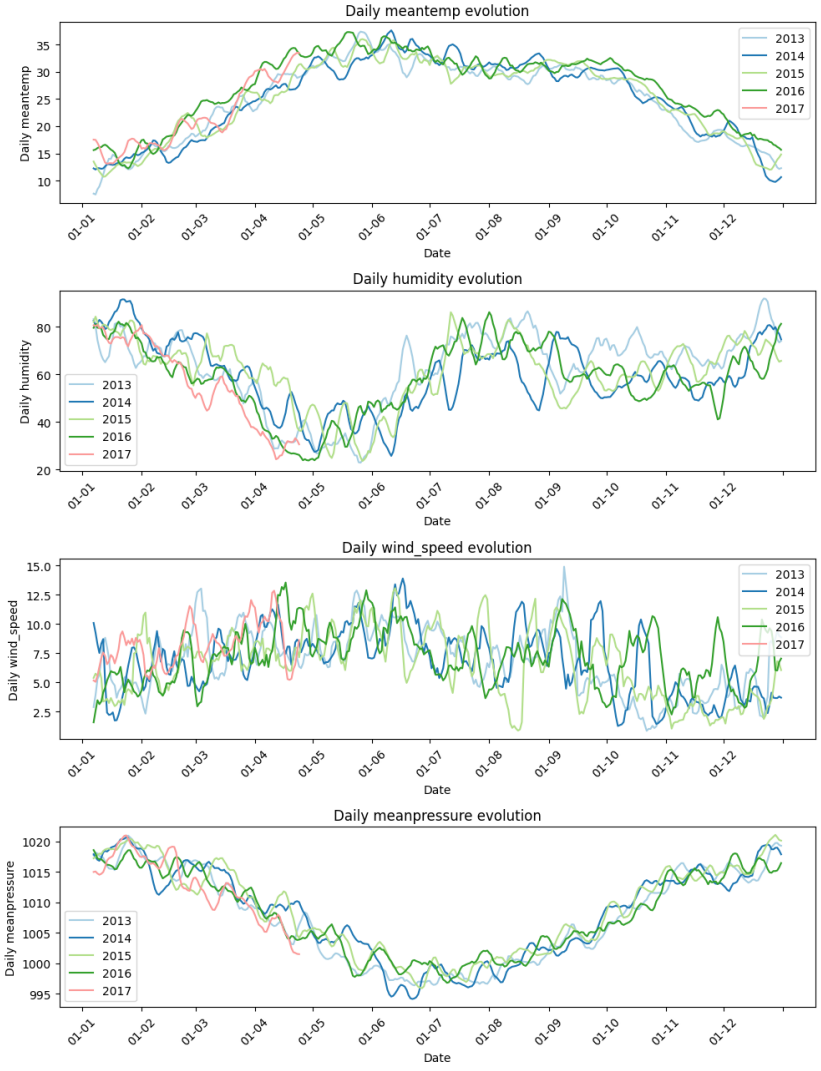

Figure 14: Temporal evolution of all attributes. From top to bottom: meantemp, humidity, wind_speed, and meanpressure.