Long Term Time Series Forecasting for Cluj-Napoca Weather Prediction

1. Introduction

Time Series forecasting has long been a field where people are always looking for advancements and better accuracy. The reason why people are so preoccupied by this field is because a large part of data has direct applications within the field. From traffic analysis, weather prediction to exchange or stock data, we can see in the world that a lot of the world today works on patterns that can be seasonal, and may be daily, weekly or yearly.

Therefore, research groups and scientists were heavily indulged in solving this type of problem as accurately as possible. From linear regression to ARIMA[1] models and more recently Long-Short Term Memories[2], time series have seen a lot of development over the years.

Nowadays, the Machine Learning world seems to be ruled by a new architecture: Transformers, from the infamous paper ***Attention is all you need[3]***. What started as a translation model, now has many applications such as Named Entity Recognition, Classification, text generation, AI Assistants and many more in Natural Language Processing and Image generation, image description and classification for Image data. Furthermore, it even combines the two of them resulting in Multimodal models that can do a little bit of almost everything. But, there has been one particular type of data that has models where Neural Networks have a hard time with: Tabular Data, where algorithms such as XGBoost[4] are still heavily used and still bring the best results. This being said, there is a type of tabular data where neural networks saw usefulness: time-series data. Until recently Long-Short Term Memories or Convolutional Neural Networks have dominated this type of datasets. Nowadays a group from Tsinghua University have discovered that transformer architectures or models influenced heavily by them work really well for these datasets. From the first model, the Informer[5], where they changed the multi-head attention layer in order to have a better space complexity until inverting all the layers (iTransformer[6]) or integrating only parts of them (TimesNet)[7], in the last 2 years and a half since its inception, there was a lot of creativity and science involved in making these type of architectures as good as possible.

In this experimental report we will compare one model that is being used extensively at the moment in practice for time series forecasting,Extreme Gradient Boosting (XGBoost). and we will compare them with two transformer models, the Autoformer, which was the first transformer model claimed to have achieved state-of-the art and iTransformers, the latest model that claims to have achieved SOTA in their ICLR paper from late December 2023. We will compare the results but also their performance and assess which model can be suitable and for how many purposes.

2. Methods

XGBoost are ensemble models that combine multiple decision trees using the bootstrapping algorithm in order to build classifiers or regressors. In time series forecasting it requires that the dataset be transformed into a supervised learning problem by using sliding-window representations. We are using this algorithm as the weakest model, because we want to construct a baseline from where to start[2].

Autoformer[8] is a type of transformer that replaces the self-attention mechanism with an AutoCorrelation mechanism. It was proposed in order to take into consideration period based dependencies that we encounter in all Time Series Forecasting datasets. Time delay similarities are calculated by using a Fast Fourier Transform (FFT), because it transforms time to a frequency domain, which is exactly what we need for finding these periods. The Autoformer algorithm was used for weather prediction at the 2022 Winter Olympics in China, so we can say that this was the first real-time use of a transformer model for Time Series Forecasting.

iTransformer is a type of transformer that adopts an encoder-only architecture. They found out that making independent time-series as tokens captures correlations in the self-attention layer and as a consequence, it learns better than it did in the models before. So instead of modifying the components of the architecture it adopts the components on the inverted dimensions with an altered architecture. The alteration can be described in the following way: we can take a multivariate time series, take all the features recorded at a time, make them as a token, embed them and pass them through the self-attention layer to find similarities with other time-series that went through the same process.

3. Exploratory Data Analysis

Dataset: a weather dataset of Cluj Napoca. It was scraped from the Open Weather Map using their Weather API. The data collected was from January 1st 2008 until May 2023 at an hourly rate. This resulted in 139394 collected data points. In our initial dataset we had 28 features as presented in figure 3.1.

Parameter	Description
city_name	City name
lat	Geographical coordinates of the location (latitude)
lon	Geographical coordinates of the location (longitude)
main.temp	Temperature
main.temp_min	Minimum temperature at the moment (optional)
main.temp_max	Maximum temperature at the moment (optional)
main.feels_like	This temperature parameter accounts for the human perception of weather
main.pressure	Atmospheric pressure (on the sea level), hPa
main.humidity	Humidity, %
main.dew_point	Atmospheric temperature (varying according to pressure and humidity) below which water droplets begin to condense and dew can form
wind.speed	Wind speed
wind.deg	Wind direction, degrees (meteorological)
wind.gust	Wind gust
clouds.all	Cloudiness, %
rain.1h	Rain volume for the last hour, mm
rain.3h	Rain volume for the last 3 hours, mm
snow.1h	Snow volume for the last hour, mm (in liquid state)
snow.3h	Snow volume for the last 3 hours, mm (in liquid state)
weather.id	Weather condition id
weather.main	Group of weather parameters (Rain, Snow, Extreme etc.)
weather.description	Weather condition within the group
weather.icon	Weather icon id
visibility	Average visibility, metres
dt	Time of data calculation, unix, UTC
dt_iso	Date and time in UTC format
timezone	Shift in seconds from UTC

Figure 3.1. Parameter-Description table of all data taken from the API

From these resulting attributes, we selected only the ones that were useful for us. So in the first iteration we cut down the categorical values and we selected one timedate value of the ones which were extracted from the API. The final values can be seen in figure 3.2.

Name	count	mean	std	min	25%	50%	75%	max
T(degC)	139391.0	9.834	9.46	-23.02	2.24	9.7	16.91	37.61
Vis(m)	139391.0	8687.608	2751.263	49.0	9999.0	10000.0	10000.0	10000.0
Tdew(degC)	139391.0	5.085	7.569	-24.39	-0.45	5.15	11.35	25.19
Tfeel(degC)	139391.0	8.745	10.259	-29.27	0.69	8.8	16.7	37.74
Tmin(degC)	139391.0	8.543	9.542	-26.18	0.84	8.45	15.75	37.0
Tmax(degC)	139391.0	12.123	9.636	-21.25	4.7	12.0	19.4	42.7
p(mbar)	139391.0	1016.13	7.613	978.0	1012.0	1016.0	1021.0	1044.0
h(%)	139391.0	75.659	19.814	7.0	62.0	81.0	93.0	100.0
ws(m/s)	139391.0	2.319	1.694	0.0	1.03	2.0	3.1	18.0
wd(deg)	139391.0	154.977	117.502	0.0	60.0	130.0	270.0	360.0
rain_1h(mm)	139391.0	0.083	0.385	0.0	0.0	0.0	0.0	18.0
rain_3h(mm)	139391.0	0.025	0.409	0.0	0.0	0.0	0.0	27.0
snow_1h(mm)	139391.0	0.012	0.084	0.0	0.0	0.0	0.0	2.28
snow_3h(mm)	139391.0	0.005	0.11	0.0	0.0	0.0	0.0	9.0
clouds(%)	139391.0	38.215	38.417	0.0	0.0	20.0	75.0	100.0

Figure 3.2 Statistics of each of the final features of the dataset

From the get-go, we can see that a lot of temperature values may have the same meaning. Also both the two snow and rain attributes may be strongly correlated. For making sure of this, we made a correlation matrix as can be seen in figure 3.3.

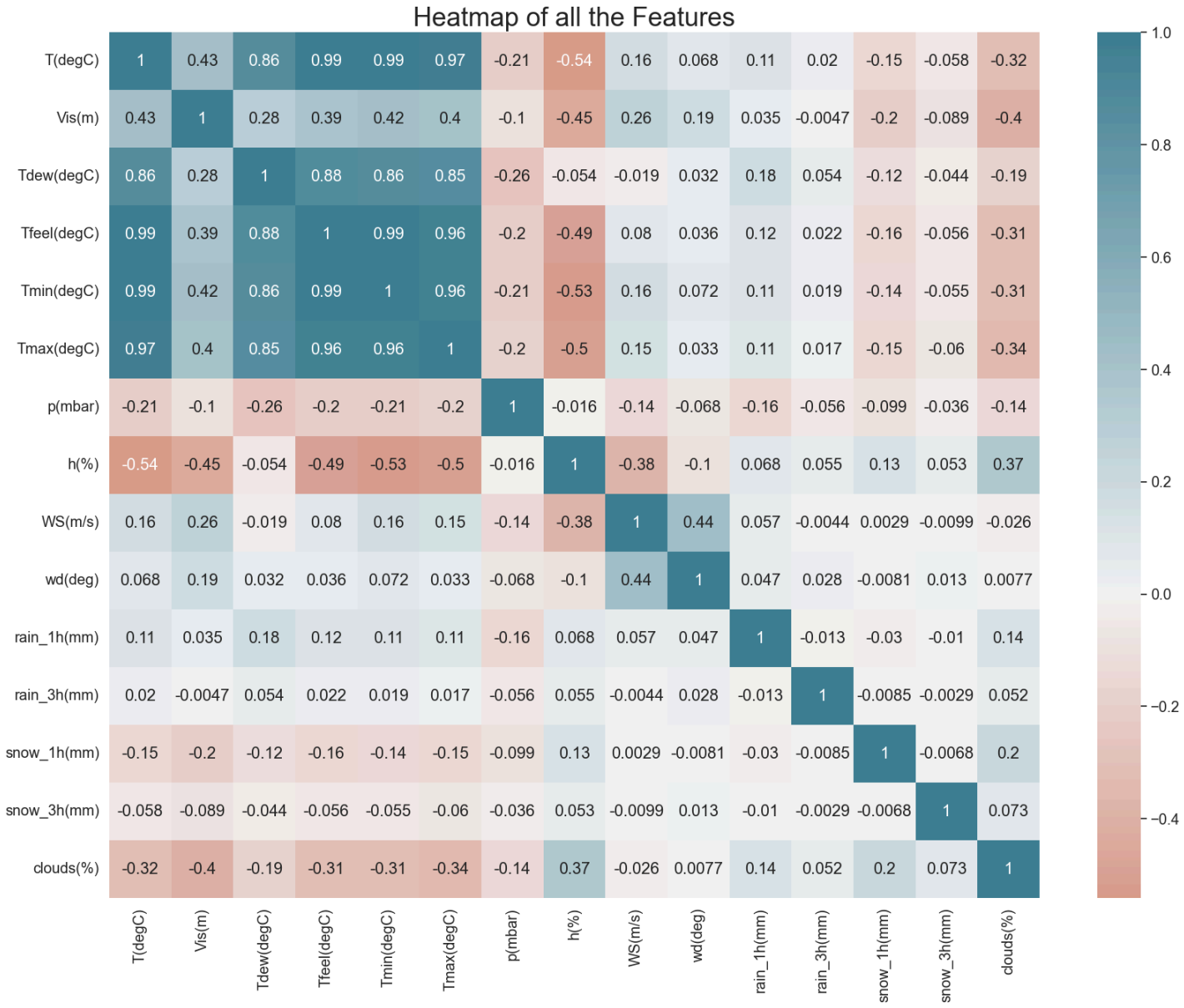

Figure 3.3 Correlation Matrix of the weather dataset.

As we suspected, the temperatures are strongly correlated with one another. Also we can see some other insights such as the humidity and pressure having an inverse correlation to the temperature. Then, we wanted to plot all the data to see how it looks. Also, we made a density plot to see the distribution of each feature. Finally, we wanted to see what the average temperature is in each month spanning from 2008 until 2023.

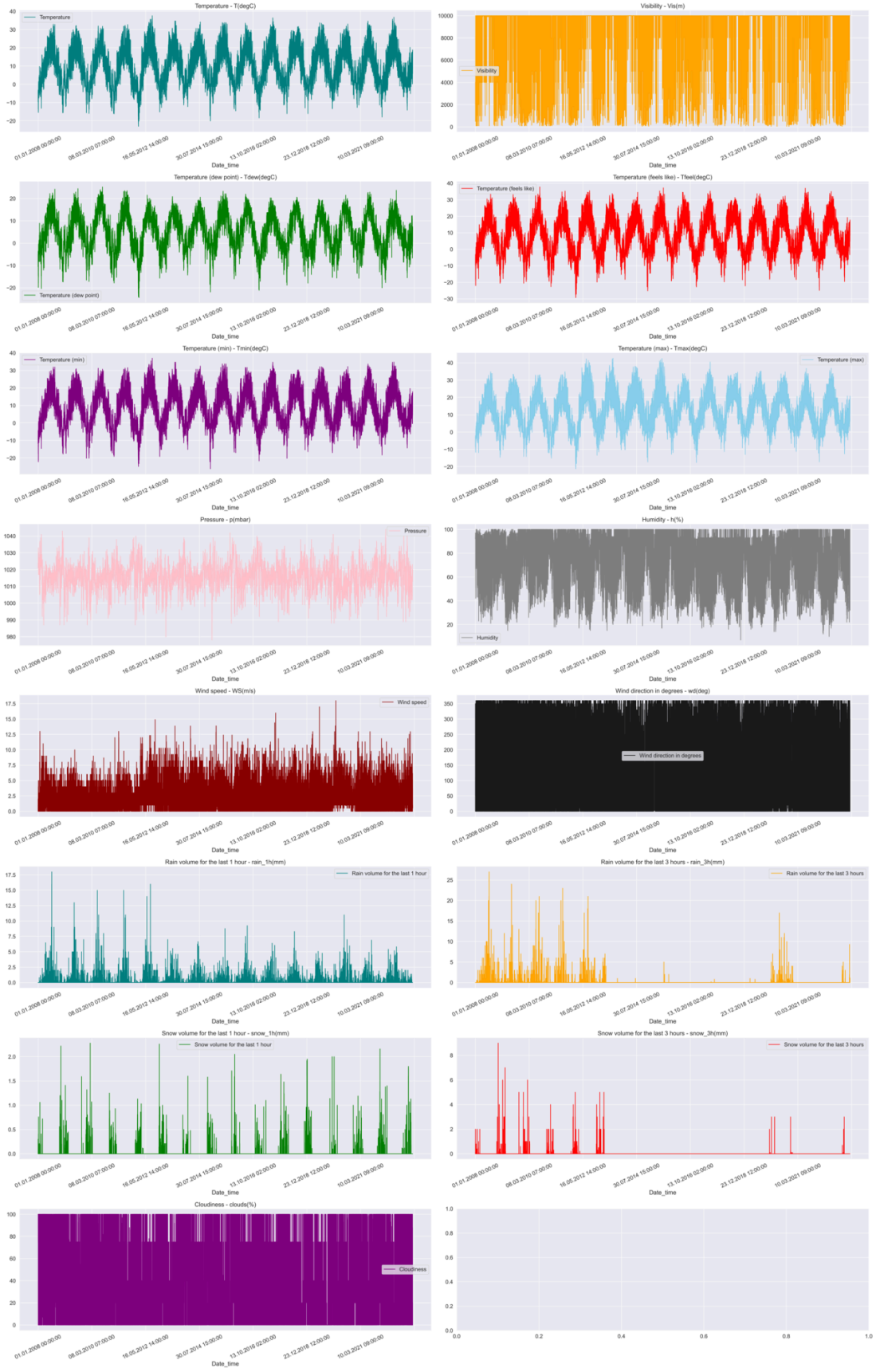

Figure 3.4. All time-series plots of the dataset

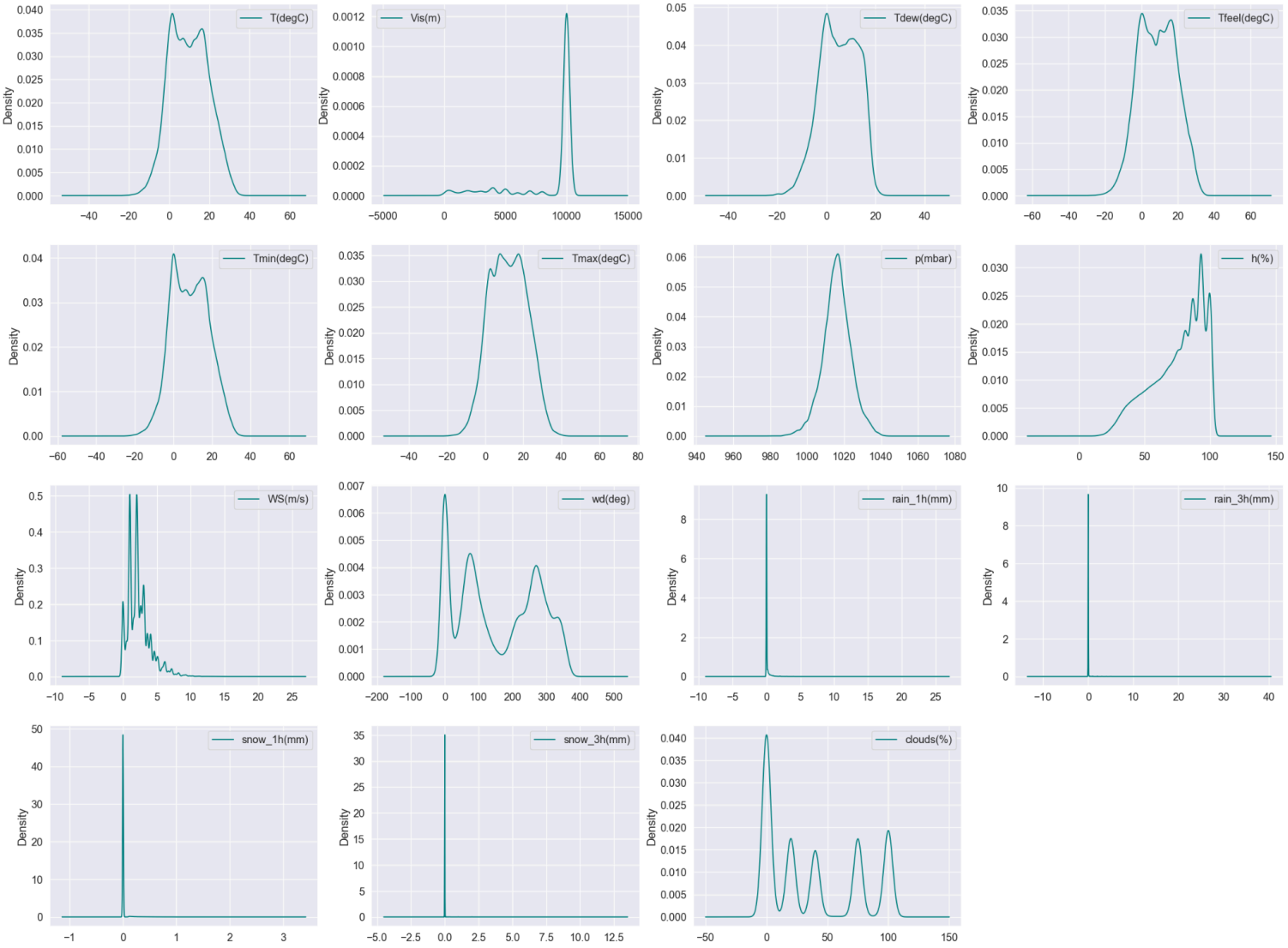

Figure 3.5. Density plot of all available attributes in the dataset.

	January	February	March	April	May	June	July	August	September	October	November	December
2008	-1.92	1.5	5.44	10.01	14.67	18.79	18.93	19.75	13.48	9.83	3.71	1.09
2009	-2.07	-0.82	3.56	12.51	15.39	18.13	20.46	20.12	16.62	9.5	5.92	-0.13
2010	-2.72	1.01	4.28	10.14	14.96	18.61	20.36	20.33	13.81	6.62	6.7	-2.09
2011	-3.31	-2.7	4.72	10.47	14.71	18.63	19.62	20.14	17.28	7.63	0.02	1.07
2012	-2.99	-7.15	4.07	11.21	15.42	19.89	23.2	21.42	17.59	10.31	4.58	-2.42
2013	-2.13	2.0	3.01	11.43	16.16	18.96	20.33	21.38	13.49	9.73	6.73	-2.06
2014	0.34	3.12	8.26	11.51	15.1	18.22	20.44	19.74	16.12	10.51	4.62	1.34
2015	-0.55	0.26	5.46	9.18	15.54	19.13	21.81	21.92	17.19	9.46	5.74	1.51
2016	-2.51	5.14	6.09	12.48	14.22	19.96	20.56	19.38	16.55	8.52	2.84	-2.77
2017	-6.52	1.73	8.32	9.46	15.56	20.24	20.8	22.19	15.55	9.84	4.96	1.49
2018	0.24	-0.17	3.32	14.94	18.16	19.35	20.28	21.98	16.3	12.03	5.37	-0.25
2019	-1.77	1.72	6.92	11.6	14.09	21.36	20.35	21.97	16.57	10.6	8.8	0.69
2020	-2.66	2.31	5.96	9.76	13.35	19.09	19.86	21.24	17.3	11.21	3.36	3.25
2021	-0.39	1.46	3.15	7.88	13.8	19.38	21.99	19.04	14.15	7.72	3.94	1.26
2022	-1.44	1.66	3.2	8.44	15.34	20.15	21.88	21.67	14.07	10.3	5.61	1.93
2023	3.08	0.53	5.64	8.05	12.75	-	-	-	-	-	-	-

Figure 3.6 Average monthly temperatures from January 2008 until May 2023

From the average temperature we can see that there has been a temperature increase in December over the years and a temperature decrease in April and even March.

4. Results and Dicussions

For Training the Transformer based models we cloned the Time-Series Library repository. We trained each of the transformers on an input sequence length of 96 hours to predict an output of length 96, 192, 336 and 720 hours.For each prediction we used the same dropout of 0.1, the same encoder and decoder input size of 15, due to the 15 features that we have, the same output size of 15. The type of embedding used was timeF, the activation function was GeLU, The dimension of the model was 512, as was the feedforward dimension. We had 3 encoding layers, 1 decoding layer and a 25 window layer of moving average. All these hyperparameters were the default ones. Also we used

The only thing we changed was in regards to batch size due to Memory usage limitations. While for a batch of 64 the iTransformer memory usage was approximately 1.5 GB for all prediction lengths, for the Autoformer it was about 3 GB for a prediction length of 96, 192 for a batch size of 32,about 4 GB for a batch size of 24 for a 336 hours prediction length and for a batch size of 16 and a 720 hours prediction length. We also used epoch delay in order to not continue the training if the error on the validation data doesn't decrease. The target feature was the temperature.

We used 96758 data points for training, 13221 data points for validation and 27159 data points for testing. So the train/validation/test split was 70/10/20%. Also we can see the difference in the size of the models. The iTransformer has a 0.02 MB model while the Autoformer one has 0.04 MB, which is more than double. This doesn't make much difference for today's GPUs and CPU's inference time, but it is significant information if we would like to train bigger models.

Also we can see this difference regarding training time. For one epoch the Autoformer spent 1000 seconds so if we had a batch of 64 we guess it would be around 10 minutes, while the transformer has a 100 seconds/epoch training time. From this we can conclude that the Autoformer takes approximately 5 times longer to train. Due to the fact that we had to make the batch size smaller, the 720 prediction length Autoformer with validation took 5 hours and 45 minutes for 4 epochs and a validation pass. In total the iTransformer took for all experiments approximately 3 hours and 40 minutes whereas the Autoformer experiments took a total of approximately 10 hours to complete.

The baseline model, XGBoost, is widely used and still considered SOTA for tabular data. It is also a very performant model for time-series forecasting, if we tune carefully each hyperparameter. In this case, we took the approach of not tuning any, because we did not tune the transformers, although for this particular model it represents a big disadvantage. Also, this was not made because of time and lack of memory purposes, as the training itself took 3 hours to complete, while using 3.8 GB of GPU. The only thing changed was splitting time into multiple features such as hour, minute, second, month and year because we needed this in order to transform it from a time-series to a regressor. As we can see in table 4.1 XGBoost performed significantly worse then the transformers architectures, but this is to be taken with a grain of salt because of the reasons mentioned above.

Models	No. of	iTransformer		Autoformer		XGBoost	
Metric	Hours	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.565	0.412	0.599	0.467	4.472	1.794
	192	0.628	0.449	0.636	0.489	4.488	1.799
	336	0.676	0.478	0.658	0.500	4.499	1.801
	720	0.727	0.505	0.689	0.516	4.514	1.805
Average		0.649	0.461	0.646	0.493	4.493	1.800

Table 4.1. Full results for long-term Forecasting of the Cluj Napoca Weather Dataset

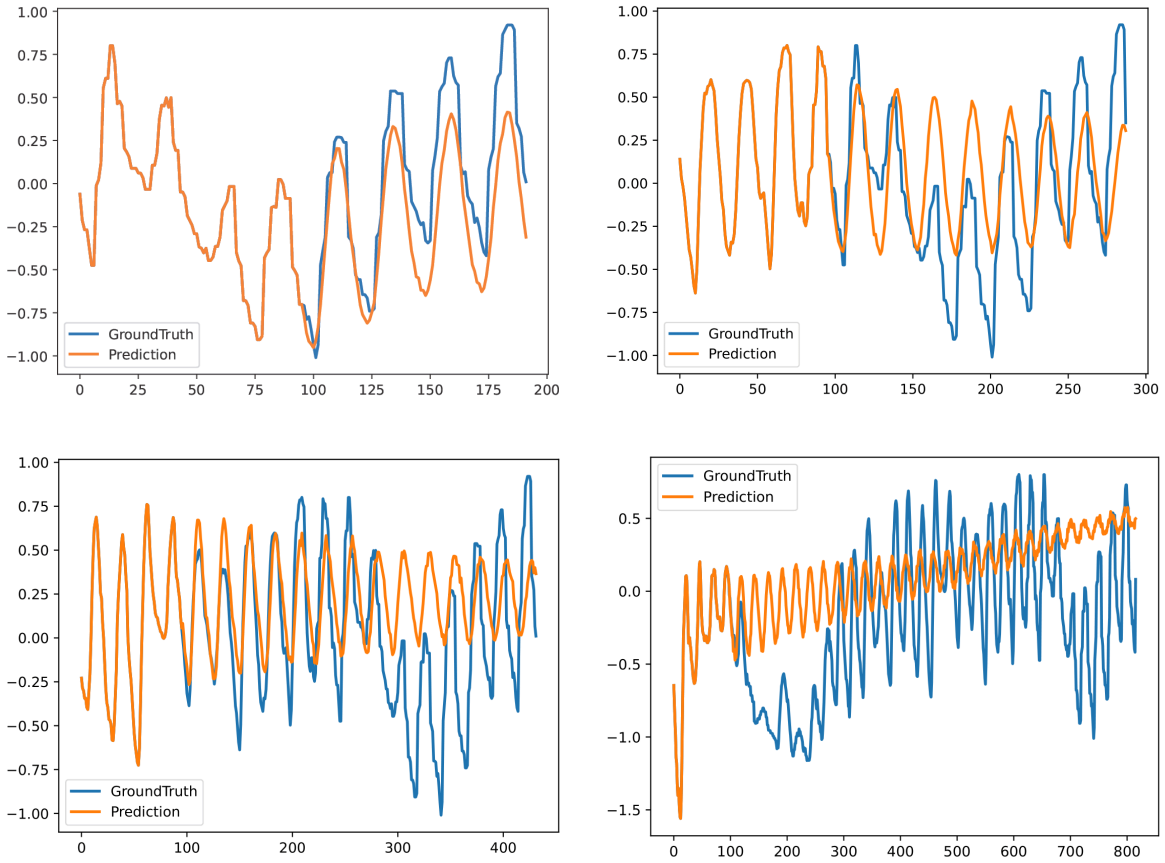

Figure 4.1. Ground Truth vs Prediction Plots for an input sequence of 96 hours for the iTransformer

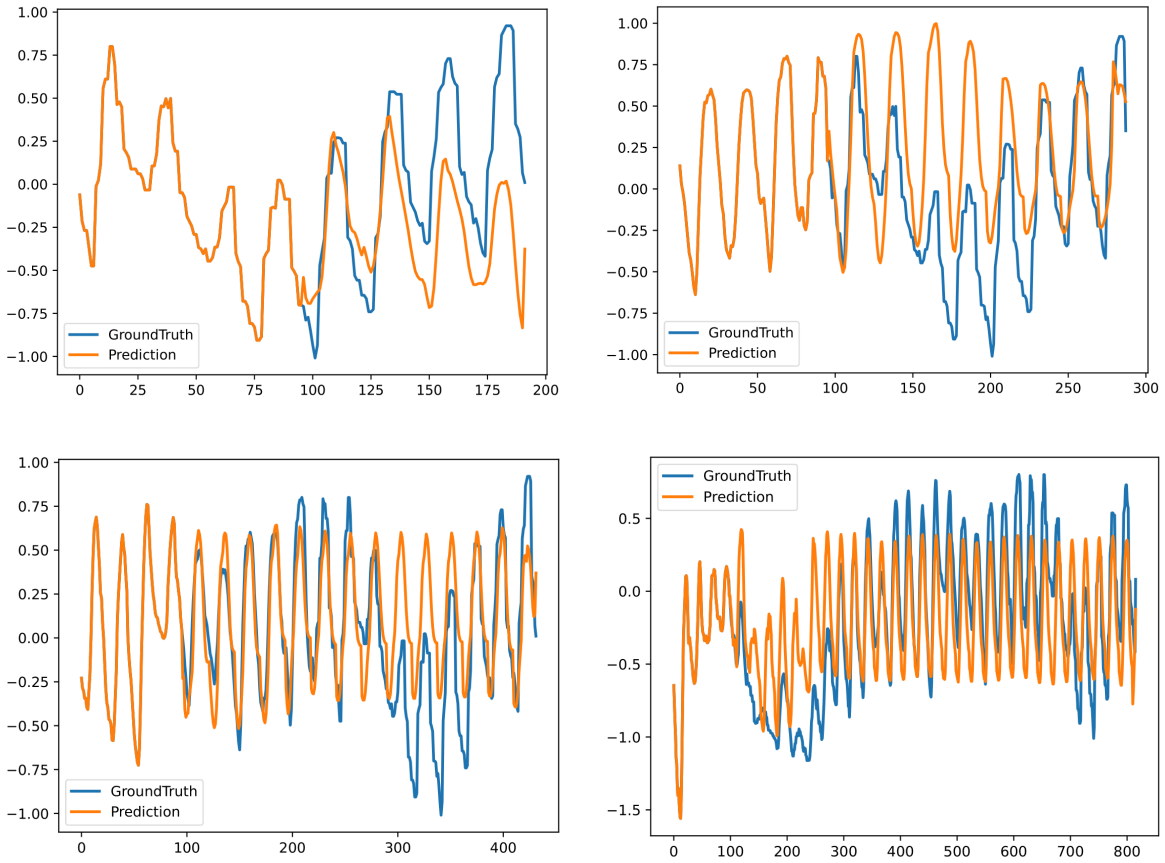

Figure 4.2. Ground Truth vs Prediction Plots for an input sequence of 96 hours for the Autoformer

The metrics used for **long term forecasting** were Mean Squared Error (MSE) and Mean Absolute Error (MAE). MSE is calculated by taking the average of the squares of the differences between the predicted and actual values. The formula is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2$$, where n = length of the dataset

Y_i = actual value

$\overline{Y}_i$ = predicted value.

One disadvantage of MSE is that the units are the squared units of the target value so for unit correctness a better approach would be to use RMSE that is the square root of MSE.

MAE measures the accuracy of the prediction model and is calculated as the average of the absolute differences between the predicted and actual values.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \widehat{Y}_i|$$

As we can see the main advantage of MAE stands in its interpretability. Also it doesn't penalize large errors like MSE which can be both an advantage and a disadvantage.

From the figures and the table above we can draw the following observations: while for a 96 and 192 hours the iTransformer is a clear winner above the Autoformer, for the other two forecasting outputs it seems that the Autoformer can generalize better and the iTransformer makes bigger errors that are penalized by MSE. But because of its speed iTransformer may have uses even in Online Learning, as the weather data changes rapidly and the models can be fine-tuned accordingly.

5. Conclusion

At the end, we can say that we have made a necessary experiment to see if these transformer models can back up what the authors stated in their various papers. We found out that it is partly true because the iTransformer did not generalize so well when dealing with longer prediction sequences. Also, we must confess that this study is not complete, as we can make the case that an iAutoformer may perform better than the original model and the inverted transformer that we tried. Furthermore, we compared them with a baseline model, XGBoost, and found out that if no algorithm is hyperparameter tuned, the Transformer ones generalize better, and in the case of the iTransformer, can be trained much faster.

6. Bibliography

[1] Box, G., & Pierce, D. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. Journal of the American statistical Association, 65(332), 1509–1526.

[2] Sepp Hochreiter, & Jürgen Schmidhuber (1997). Long Short-Term Memory. *Neural Computation, 9(8), 1735–1780.*

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, & Illia Polosukhin. (2023). Attention Is All You Need.

[4] Tianqi Chen, & Carlos Guestrin (2016). XGBoost: A Scalable Tree Boosting System. CoRR, abs/1603.02754.

[5] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, & Wancai Zhang. (2021). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting.

[6]Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, & Mingsheng Long. (2023). iTransformer: Inverted Transformers Are Effective for Time Series Forecasting.

[7] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, & Mingsheng Long. (2023). TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis.

[8] Haixu Wu, Jiehui Xu, Jianmin Wang, & Mingsheng Long. (2022). Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting.