**X/HEC TIME SERIES 2023 - Practical Session Report – Group 6**

**Group Members:** João Melo, Elizaveta Barysheva, Maria Susanne Stoelben

**Data Preprocessing**

First, we extracted time features such as day, month, year, weekday, weekend (bool), and season. Second, we created a high consumption indicator that alerts if the electricity consumption is above the 90%-percentile for the region. Third, we categorized the cloudiness. Fourth, we created lag-extended features. The dataset has data per 30 minutes, so we used a multiplication by 48 for each day to create lag intervals. Also, as we used the shift function, we had to make sure that for each 30-minute timestamp, there is a corresponding row – if not, the empty row should be added. Lastly, we applied min-max-standardization to numerical columns. For the following steps, we used the provided preprocessed data from Dropbox.

**Model Architecture and Training**

First, we ran an RNN (many-to-one) and LSTM (many-to-many). The RNN was constructed as a univariate single-layer RNN with one linear layer to generate the next hidden state and one linear layer to generate the output, which both take the stacked input and previous hidden state as input. Additionally, we used a ReLu activation on the new hidden state. For the LSTM, we built a univariate, multivariate, and region-agnostic version. The LSTMs all had 2 layers and a ReLu activation for the output before it was passed to a fully connected layer to generate the predictions. The input sequence was made up of the information of the past 24 hours (always selected to be only one region in one sequence) to predict either the next hour or all values for the next 24 hours. For not-region-agnostic models, Ile de France was taken as an exemplary region to subset the data. We min-max-standardized all numerical features. Due to runtime complexity, we used a subset of 20% of the data for the RNN model. The testing data contained 20% of the preprocessed data set. We ran each model for 10 epochs. For the RNN and LSTM, we used the MSE loss and Adam optimizer with a learning rate of 1e-4. Second, we also ran a TFT (many-to-many). We used the TFT of the Pytorch forecasting library, which is based on a specific time series forecasting dataset. For this dataset, we specified the region as a group, an encoder length of 12 to 24, a prediction length of 1 to 24, a softplus group normalizer, and different known / unknown reals and categoricals. We only used 40% of all data for this model due to runtime complexity. Furthermore, we used the quantile loss with the learning rate equal to 5e-2. The testing set used the last 180 days in the preprocessed data.

**Results and Comparison**

In the table below, we compared the different model architectures by listing the SMAPE and MAE on the test set and the run time for training. It is important to note that the models were trained on different data. Therefore, the only two models that were tested on the same data are the uni- and multivariate LSTM. Nonetheless, we believe that the results still give a good indication for key learnings.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | sMAPE | MAE | Run Time |
| RNN | 15% | 2543 | 11.38 minutes |
| LSTM (univariate) | 6% | 1022 | 50 seconds |
| LSTM (Multivariate) | 5% | 753 | 38 seconds |
| LSTM (AlL Regions) | 2% | 257 | 10.38 minutes |
| TFT | 5% | 436 | ~50 minutes |

The LSTM models were the fastest by runtime, especially considering that they were trained on all data. The RNN performance is the worst looking both at SMAPE and MAE. Moreover, we noticed that the predictions for the RNN were quite unstable, i.e., using the model to predict recursively leads to far-off results for hours closer to the 24 hours’ time horizon. It is interesting to see how the performance of the LSTM improves by switching from univariate to multivariate and finally to region-agnostic. The first improvement is relatable to classical machine learning since more variables might hold information about the target. For example, we would expect the electricity consumption to be higher in the morning and evening for households. The second performance improvement of the LSTM might be explained by two leavers. First, the model is fed more data – 12 times as many sequences to be exact. Second, it is possible that the model picked up on cross-region patterns that supported the learning. The TFT model showed a high runtime even when restricted to 40% of the data, and the performance was inferior to the comparable LSTM (region-agnostic). In conclusion, for this specific task, we deem the region-agnostic LSTM the most preferable algorithm of the tested models.

**Discussion**

The deep learning methods can handle larger amounts of data, are more straightforward for multivariate approaches, and are able to identify more complex patterns than classical machine learning models. Also, one can use them natively to predict a time sequence into the future, which is quite different from traditional time series models such as ARIMA, which always only predict the next point in time. However, one major challenge in implementing the deep learning models is the runtime due to model complexity. This makes experimenting with a model more difficult and means that the computational cost of maintaining the model over time is high. Specifically, transformers are known for their need for high amounts of data to perform well. The interpretability of deep learning models is another challenge compared to a more straightforward feature importance approach of classical machine learning models. However, techniques like attention mechanisms and layer-wise relevance propagation may offer some interpretability. Finally, if a company wants to do time series analysis, and the focus is obtaining very accurate results from a large dataset, deep learning methods might be a better and more versatile choice compared to their classical machine learning counterparts due to the various advantages already mentioned.

**Conclusion**

In conclusion, experimenting with different state-of-the-art models for deep learning in time series modeling was quite insightful. Especially with high amounts of data and for multivariate as well as grouped time series, deep learning models seem to be a reasonable choice. In future works, it would be interesting to generate coherent testing to be able to properly compare the different algorithms and select the best one for the task at hand. Additionally, more extensive feature engineering and selection might deliver some performance improvements.