,

UTRECHT UNIVERSITY

FACULTY OF SCIENCE

Applied Data Science

Final Thesis Project

Using LSTM and XGBoost for streamflow prediction based on meteorological time series data

Maryam Afshari Hemmatalikeykha

Advisors

Dr. Edwin Sutanudjaja
Prof. dr. Derek Karssenberg
Youchen Shen

July,2022

Abstract

Streamflow prediction, as one of the most critical issues in hydrological studies, plays a crucial role in water resources management namely reservoir operation, water allocation, and flood control. In this study, daily streamflow prediction (obs) for Basel and Lobith stations at Rhine Basin for the years 1996-2000 were derived from three Machine Learning (ML) models based on meteorological time series such as precipitation(p), temperature (t), and reference potential evapotranspiration (et) covering 1981-1996. A Recurrent Neural Network, Long-Short Term Memory (LSTM) as a time series model along with two non-time-series models; Extreme Gradient Boosting (XGBoost) and Multiple Linear Regression (MLR), have been developed to carry out. The latter was used as the ML benchmark of the former models. Moreover, a hydrological model, PCR-GLOBWB was applied as the second benchmark. It is found that ML models could fail in predicting streamflow from only meteorological variables in the absence of past values of streamflow due to the lagged relationships between streamflow and meteorological variables. To overcome this problem, we investigated the optimal time-lag value between predictors and streamflow combining the statistical method cross-correlation with the implementation of the LSTM model. The optimal time-lag value indicated the time window parameter in the LSTM model and the number of lagged variables to be included in the non-time-series models. Consequently, two possible input scenarios were considered in developing the non-time-series models (i) using solely meteorological predictors and (ii) using lagged predictors equal to the optimal time-lag value in addition to the meteorological predictors. The results demonstrated that overall, ML models can achieve satisfactory results in predicting streamflow based on meteorological features. However, using only meteorological variables non-time-series models cannot provide high accuracy. Only when prior records of meteorological parameters were combined did their accuracy noticeably increase. Among ML models, the LSTM model outperforms other models in the Lobith station and shares a similar performance with the lagged version of the MLR model in the Basel station. Regarding the performance consistency in streamflow prediction, the LSTM is the superior model in both locations.

Contents

Chapter 1

Introduction

Streamflow prediction is an undeniably important agent for water resources management [1, 2]. As climate change is predicted to cause frequent extreme events in the future, the accuracy and skill of streamflow prediction models are of not only scientific value but also great societal importance.

The two common approaches in streamflow prediction are hydrological models and machine learning (ML) models [3]. As the former approach has limitations such as high computational demand and vulnerability to the model structure errors, recently, ML models are increasingly being employed for streamflow prediction. Rather than directly simulating physical processes, ML models imitate physical rules from historical data to form a functional relationship between inputs and outputs. They are generally faster to train and can work with any predictors [4]. Several studies have demonstrated that ML models outperform conventional methods based on hydrological models [5, 6, 7, 4]. Yaseen et al. [8] reviewed papers from 2000 to 2015 on the application of ML models for streamflow prediction and found that they had made significant progress in this area.

Although there are many ML prediction models, not all can be directly employed for long-term streamflow prediction [9]. The Autoregressive (AR) models, for example, perform well in time series forecasting, but they heavily rely on observed streamflow from the previous time steps and thus grant accurate estimates for forecasting with a relatively short lead-time [10]. To be able to predict streamflow over longer time spans, and/or overtime spans that lack observed streamflow data, these methods cannot be used. In these cases and to enhance accuracy, non-linear methods that explain the nonlinear interaction between the input variables and the target are recommended [11].

Moreover, the architecture of the model should be capable of capturing the temporal features that are vital for time series prediction. Among ML models, Recurrent Neural Networks (RNNs) have the ability to take into account the sequential nature of time series explicitly and learn more efficiently. Therefore, they are frequently used or included as components of the deep learning frameworks of time series models [12]. In streamflow prediction, for example, RNNs are being increasingly employed to recognize time-dependent features [13, 14]. According to Duan et al. (2020) among RNNs, the most commonly used network is the Long-Short Term Memory (LSTM) [9]. Compared to other types of RNNs, LSTMs do not have a problem with exploding and/or vanishing gradients and can learn long-term dependencies between input and output features [4]. Numerous studies applying LSTM approve its capability in streamflow prediction [15, 9, 4, 16, 13, 17]. Therefore, LSTM appears to be an eligible model for streamflow prediction.

On the other hand, although LSTMs typically produce passable results in prediction accuracy; computational time of prediction and a high degree of data demand are important aspects to be considered. Given that discharge data are often limited in many parts of the world [17], we looked for another ML method, a less data-hungry algorithm. Compared to other ML methods Decision Tree (DT) is computationally cheaper. Also, DT-based models offer the advantage of being easy to interpret and visualize [18]. More importantly, DT-based models perform well when provided with limited training data in certain domains[19]. Lately, a DT-based model, Extreme Gradient Boosting (XGBoost), introduced by Chen and Guestrin (2016)[20], has become popular in ML competitions since this fast, efficient, scalable model yields promising results in many domains [21, 22, 23]. Gauch et al. (2021) aimed to evaluate XGBoost and LSTM-based models to limited training data in predicting streamflow. Their results show that XGBoost and LSTM-based models provide similarly accurate predictions on small datasets, while LSTM is superior when more training data are available [17]. Given that for many regions, sufficient meteorological records are generally not available and/or not feasible to record [24, 17], we think XGBoost is an attractive but less known ML model in streamflow prediction.

Accordingly, this study documents the development of two different ML methods in predicting streamflow based on meteorological predictors in Basel and Lobith stations at Rhine Basin; LSTM and XGBoost. While LSTM as a time series model takes into account the temporal dependencies in the data, XGBoost is a more computationally efficient model that works well when provided with small data. Although it is not explicitly designed for time-series prediction, it is proven to work well in time-series problems [25, 26, 27, 28]. To realize if the use of these ML models is advantageous, we benchmarked them to two other models: (i) Multiple Linear Regression (MLR) as an ML benchmark (ii)and PCR-GLOBWB, a global hydrology model, as the hydrological benchmark [29].

It is acclaimed that ML models could fail in predicting streamflows from only meteorological variables in the absence of past values of streamflow. The main reason for this could be low and lagged relationships between streamflow and meteorological variables[30]. To overcome this problem, we investigated the lagged relationship between predictors and streamflow combining the cross-correlation statistical method with the implementation of LSTM. The optimal time-lag value indicates the time window parameter in the LSTM model and the number of lagged variables to be included in the non-time-series models of XGBoost and MLR. To check the effect of the time-lag value we will consider different input scenarios in developing the non-time-series models (i) using only meteorological predictors and (ii) using lagged predictors equal to the optimal time-lag value in addition to the meteorological predictors.

Shen et al. (2022) conducted a study on streamflow prediction of Rhine Basin in which to improve hydrological streamflow predictions, an updating procedure was implemented[10]. In order to correct predictions of the PCR-GLOBWB, researchers included simulated streamflow in addition to the meteorological data as predictors of a Random Forest (RF)-based model. Interestingly, their results show that using input meteorological variables along with their lag time of 10 days can elicit results that are equivalent to error-correction procedures in Basel stations and slightly lower in Lobith and Cochem stations.

Although the performance of the RF model is noticeably better than the PCR-GLOBWB model, their approach has some limitations that this study aims to address: (i) by using simulation variables, their approach still relies on a hydrological model. We aim to predict streamflow in an efficient manner by excluding the simulation variables, taking into account that physically based models typically require significant computational expense. (ii) they employed a limited time lag. Considering the memory of catchment, discharge may take up until several months to even several years [31] and therefore this time is too short for streamflow prediction. Thus, higher lag times were examined in our analysis. (iii) they did not investigate the optimal time-lag value. We deliberately sought to determine the best time-lag value for each station. (iv) finally, LSTM and XGBoost appear to be superior candidates to Random Forest employed by Shen et al. As previously noted, LSTM is a time series model which takes into account the temporal dependencies in the data. On the other hand, we believe XGBoost would be a good fit here because the time period of data available for this investigation is constrained.

In a nutshell, this study aims to predict streamflow exclusively from meteorological data using LSTM and XGBoost. The outcomes will be compared with those of the error-correction procedure described in Shen et al paper. Consequently, the following is the main research question and its sub-questions:

- Using meteorological variables, how do other ML methods such as LSTM, XGBoost, and MLR perform compared to the error-correction procedure described in Shen et al. paper?

    1. Until which lag, past information of each predictor is correlated with streamflow in each station? In other words, what is an appropriate time-lag value for each station?
    2. How does the performance of LSTM differ from the performance of XGBoost in predicting streamflow?
    3. How does including lagged variables affect the performance of XGBoost and MLR models?

To accomplish the described objectives, the cross-correlation method was used to obtain insight into the correlation period between each predictor and streamflow. We then trained many LSTM models using the same structure but different time windows within the correlation period to gain the ideal time-lag value. The prediction accuracy of all methods is validated using Nash-Sutcliffe efficiency (NSE) and Original Kling-Gupta Efficiency (KGE). To take into account the stochastic nature of this algorithm and increase the accuracy of the results, We ran the LSTM model ten times on each time window and presented the average validation metrics. The time window of the LSTM model with the highest KGE value is chosen as the optimal time-lag value. Regarding XGBoost and MLR models, different sets of input variables were tested to inspect the effect of including lagged variables. Finally, we analyzed how well these models performed in comparison to the benchmarks.

Chapter 2

Literature Review

1 Streamflow prediction using meteorological data

Several studies employed the idea of predicting streamflow from solely meteorological data using ML methods in different parts of the world. Adnan et al. (2019) investigated different ML models in modeling monthly streamflow using precipitation and temperature inputs at Swat River Basin in Pakistan. Their results show that monthly streamflows of Kalam Station can be predicted using only temperature data. Moreover, only precipitation inputs also provide good accuracy for Kalam Station while they produce inaccurate predictions for the Chakdara Station [24]. Tongal and Booij (2018) compared ML methods such as MLPNN, SVM, and RF using precipitation, temperature, and evapotranspiration, in streamflow modeling using data from four rivers in USA [30]. Their results confirm that employing meteorological data can lead to accurate predictions of streamflow.

2 LSTM

LSTM is a ML model for time-series prediction [32]. It takes a time series as input and updates internal memory at each time step to calculate the output value. The way output value is calculated is driven by a set of parameters, or weights, that are learned during training (hydrologists would refer to this training process as calibration).

LSTMs are being increasingly employed for streamflow prediction [9, 16, 4, 14, 13]. According to Duan et al. (2020) among such models, LSTM is the most commonly used RNN in streamflow prediction [9]. Duan et al. (2020) document the development of an ML-based modeling system for estimating future daily streamflow in California. Comparing Temporal Convolutional Neural Network (TCNN) model with other commonly used ML models, their study indicates that there are some important temporal features that ANNs struggle to capture, contrary to TCNNs and other RNNs such as LSTMs and GRUs. Moreover, they conclude that overall, the TCNN model performs better in the high-flow regime, whereas the LSTM model performs better in the low-flow regime[9]. Kratzert et al. (2019) used LSTM and Catchment Attributes to predict streamflow over CONUS. They benchmarked the LSTM model against several high-quality existing hydrological models. Their results demonstrate that LSTM with catchment attribute outperformed multiple locally- and regionally-calibrated benchmark models by a large margin. Interestingly, the results show LSTM without static catchment attributes (only trained on meteorological forcing data) outperformed both regionally calibrated models consistently as a single model and even more so as an ensemble [4]. Feng et al. (2020) added the lagged observations among inputs as data integration, which elevated the prediction accuracy of the LSTM model. They also employed a convolution data integration method in which a time-series segment of recent observation passes through a CNN unit, however, the resulting model did not outperform feeding observations directly into the LSTM model. Intriguingly, their study points without any data integration, the LSTM model already provided predictions that were competitive with other models. Yan et al. (2019) used LSTM to forecast streamflow in a small watershed. Their results make evident that LSTM can effectively focus on hidden flood factors from previous hydrological sequences [13]. Ma et al. (2019) investigated the competence of LSTM to predict water levels up to 10 days ahead in Oestrich and Cologne gauges of Rhine Basin using historical data from 2008 to 2015. They ran several tests afterward to test the model between July 2018 to December 2018. The results show LSTM capability in forecasting water level[15].

3 XGBoost

While LSTM as a time-series prediction model is an appropriate fit here, XGBoost might seem like a less suitable option due to the flat input vectors condition. Over the last few years, however, researchers have employed XGBoost with significant success in numerous time-series prediction tasks [25, 26, 27, 28].

Regarding streamflow prediction, Gauch et al. (2019a) found the XGBoost model to perform more accurately than LSTMs [19]. However, what makes XGBoost advantageous is computational efficiency and capability to perform well on limited data. Gauch et al. (2021) aimed to evaluate XGBoost and LSTM-based models to limited training data, both in terms of geographic diversity and different time spans in predicting streamflow. They trained both models on meteorological observations of the CAMELS dataset while individually restricting the training period length, the number of training basins, and input sequence length. Their results show that XGBoost and LSTM-based models provide similarly accurate predictions on small datasets, while LSTM is superior given more training data [17]. In another study, Gauch et al. (2019b) compared a physically-based model VIC-GRU with ML methods such as XGBoost and Ridge in predicting the streamflow of a gauging station given five years of meteorological data. Their results indicate that XGBoost provides the most accurate predictions of the three examined models on such a small training data [7]. Additionally, XGBoost is said to exploit out-of-core computation and enables the researcher to process hundreds of millions of examples on a desktop[20, 22, 23]. Phankokkruad and Wacharawichanant (2019) compared XGBoost with a `CNN_LSTM` model for service demand forecasting. Their study shows that XGBoost operates faster, takes a lower-performing time also has lower CPU consumption compared to the `CNN-LSTM` model[33].

4 Time-lag value

In the literature, most of the methods suggest choosing the optimal time-lags throughout the experiments, which may not always be sufficient in real-world problems [34, 35, 36]. Some papers, however, proposed methods other than an experiment in the literature to explore the proper selection of time-lag value. Surakhi et al. (2021) applied a comparative study between three methods to investigate the effect of selecting an optimal time-lag value on the performance of the prediction model. These methods include a statistical autocorrelation function, an LSTM model along with a heuristic algorithm to optimize the choice of time-lag value, and a parallel implementation of LSTM that dynamically chooses the best prediction based on the optimal time-lag value. Their study shows that the statistical method is precise and gives a good indication of the behavior of data correlation and it can be used to suggest an appropriate time-lag value that will be used later as a parameter of the deep learning prediction model [37]. In another study, Yaseen et al. (2016) used auto-correlation for monthly streamflow prediction, ultimately choosing a time lag of 5 months [38].

As mentioned above, ML models could fail in predicting streamflows from only meteorological variables in the absence of historical values of streamflow possibly due to lagged relationships between streamflow and meteorological variables[30]. There have been studies that overcame this challenge by investigating the proper time-lag value. Adnan et al. (2019) used different ML methods in modeling monthly streamflow using precipitation and temperature inputs at the Swat River Basin in Pakistan. To find the lagged relationship, the cross-correlation method was applied. They chose a time lag of 5 and 2 months for precipitation and temperature input, respectively [24]. Tongal and Booij (2018) compared ML methods using precipitation, temperature, and evapotranspiration, in streamflow modeling using data. They used a simulation framework by coupling a base flow separation method to the ML methods to account for the lagged relationship between the meteorological input variable and streamflow[30].

Chapter 3

Data and Methods

1 Data

Daily streamflow data from Basel and Lobith stations from 1981 to 2000 in the Rhine Basin obtained from GitHub repository related to Shen et al. paper [10]. The dataset contains daily streamflow with three meteorological variables: precipitation (p), temperature (t), and reference potential evapotranspiration (et). These are values for the upstream area (i.e. average over the catchment of the streamflow measurement location). These features are commonly used for streamflow prediction in the literature [9, 39] The data set includes 7305 records of 4 columns (three inputs and one output). The target variable is the streamflow and the input variables are the remaining meteorological variables. The data has high quality, and there is no noise nor are there missing values. Figures 1 and 2 in the Appendix section show time series of observed streamflow and the predictors.

1.1 Study area

Basel station's streamflow, located in the Alpine region of the upper Rhine, has a nival regime that is dominated by snow and glacier melt in spring and summer. Whereas Lobith station's streamflow, located in the lower Rhine, has a pluvial regime characterized by high streamflow in winter and minimum in late summer.

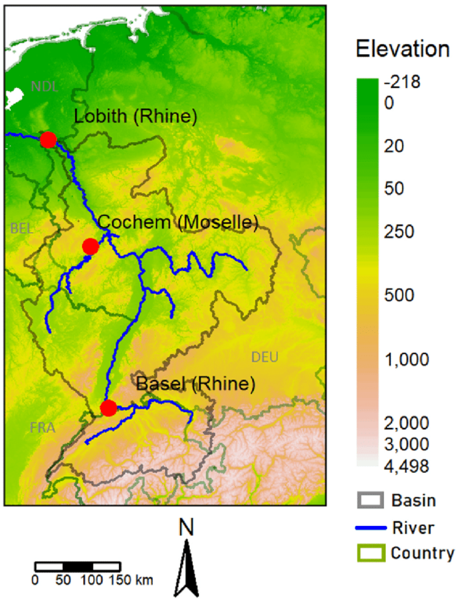

Figure 3.1: Digital elevation map from SRTM 90m digital elevation database (http://srtm.csi.cgiar.org/) of the Rhine Basin with Lobith and Basel stations indicated by red dots.

1.2 Scaling and transformation

The data transformation is influential in some ML models' training. It ensures that variables that are measured at different scales contribute equally to the model fitting. In this study, data was transformed for the LSTM and MLR models using the MinMaxScaler normalization (defined in Eq. 3.1) provided by Sklearn Preprocessing library [40].

$$\texttt{x\_scaled} = \texttt{x\_value} * (max - min) + min, \tag{3.1}$$

where `x_scaled` is the new scaled value, `x_value` is the original value, max is the maximum value of a given variable in the dataset observation and min is the minimum value in the dataset.

Regarding the XGBoost model, input variables do not require any normalization [41]. It is because the XGBoost model is not sensitive to monotonic transformations of its input variables. The model only needs to pick "cut points" on input variables to split a node. i.e. defining a split on one scale has a corresponding split on the transformed scale [42]. As a result, we skipped this step for the XGBoost model.

2 Methods

2.1 PCR-GLOBWB

The result of this study is compared to the PCR-GLOBWB model. Therefore, we briefly describe this model in this section. The PCR-GLOBWB model is a global hydrology and water resources model that can predict streamflow as well as state variables used in Shen et al. paper. It has a spatial resolution of 30 arcsec (about 1 km at the equator) and operates based on daily water balance [29]. PCR-GLOBWB model weigh not only natural processes but also human water use [43, 44, 45]. This model provides satisfactory streamflow predictions throughout the world [43]. In some cases, however, the PCR-GLOBWB model fails to simulate the streamflow as a result of an error in model structure, drivers, and parameters[10]. In this study, uncalibrated version of this model was used.

2.2 Streamflow prediction

In this study, we have a time series model, LSTM, and two other ML models, XGBoost and MLR which are not specifically designed for time series modeling. We started by exploring the optimal time-lag value. This serves two purposes:

- Regarding the LSTM model, the time window (also known as input batch size/ input sequence length) is one of the most critical hyperparameter optimizations in designing an LSTM model. Hence, by exploring optimal time-lag values we ensured the development of optimal configurations in the LSTM model.

- Regarding XGBoost and MLR (non-time series model) lagged variables equal to time-lag value were created to incorporate time-series information into them.

To see if the lagged relationship between variables increases the performance of the MLR and XGBoost models, we tested different sets of input variables. First, we trained these models only with meteorological input variables. We then trained them with meteorological input variables plus the lagged variables equal to the optimal time-lag value. Additionally, all time steps between the furthest time backwards were employed in generating lagged variables.

In order to have comparable results to Shen et al. (2022) paper, we fed all the variables in all ML models mentioned, and no method regarding variable selection was employed.

In the following sections, we will explore the dataset splitting, the time-lag value, the structure of models used, and the evaluation metrics.

2.2.1 Dataset splitting

In training all models, the dataset is split into two parts before the training process: the training set and the testing set. The training set is used to build the model and fit it to the available data with known inputs and outputs. The testing set is used to estimate the model performance on unseen data (data not used to train the model). 75 percent of the dataset is allocated for the training set and 25 percent of the dataset is allocated for the testing set.

As we have time-series data, we performed a time-series split. We kept the first 15 years as the training phase and the last 5 years as the testing set.

2.2.2 Time-lag value

As mentioned earlier, a proper time window parameter in the LSTM model is essential to create a set of training examples to be used for the next prediction which is exceptionally critical for model performance. On the other hand, feeding the optimal time-lag value to the XGBoost and MLR models, we introduce time-series information to these models.

To explore the optimal time-lag value, the cross-correlation function (defined in Eq. 3.2) provided by Scipy library[46] is used to get an indication of the correlation between the input and output variables.

$$z[k] = (x * y)(k - N + 1) = \sum_{l=0}^{||x||-1} x_{\mathrm{l}} y^{*}_{\mathrm{l-k+N-1}}, \tag{3.2}$$

for $k = 0, 1, ..., ||x|| + ||y|| - 2$

where $||x||$ is the length of x, $N = max(||x||, ||y||)$.

We then plotted the cross-correlation of different predictors and streamflow for each station to get an idea about the correlation period between the predictors and streamflow. Following that, many LSTM models were trained with the same structure but different time windows within the correlation period to gain the ideal time-lag value. To take into account the stochastic nature of the LSTM algorithm and increase the accuracy of the results, we ran the model 10 times on each time window. The time window of the LSTM model with the highest KGE value is chosen as the optimal time-lag value.

In determining optimal time-lag two points were considered: model complexity and computational power needed. Regarding model complexity, the value of time-lag should be large enough to account for all historical information relevant to predicting streamflow, however large time-lag can yield increased model complexity and thus reduce model performance [9]. Therefore in determining time-lag value, the trade-off of complexity and performance should be taken into account. Besides the model complexity, we were concerned with our limited computational power and time. Therefore we defined the scope of the search for optimal time-lag withing 0 to 205 lags.

2.2.3 Long-Short Term Memory (LSTM)

The LSTM architecture is a special type of RNN, developed to overcome the inability of the traditional RNN to learn long-term dependencies. Bengio et al. (1994) have proved that the traditional RNNs can rarely remember sequences length of over 10[47]. Considering the memory of catchment, discharge may take up until several months to even several years. Hence, this time is too short for streamflow prediction[31]. LSTM has three gates- the update gate u, the forget gate f, and the output gate o as depicted in figure 1.

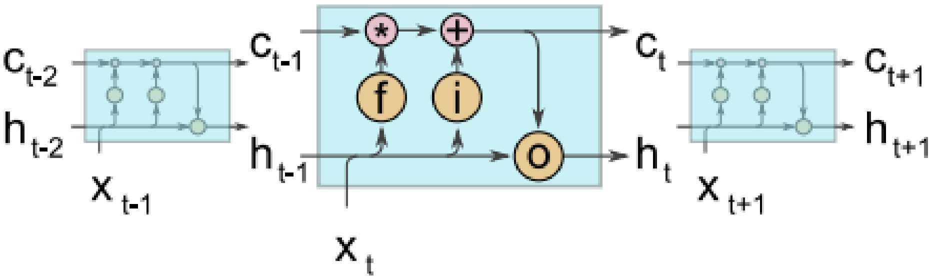

Figure 3.2: The interiors of an LSTM cell, where f stands for the forget gate (Eq. 3.2), i for the input gate (Eqs. 3.3–4), and o for the output gate (Eqs. 3.6–7). c_t denotes the cell state at time step t and h_t the hidden state.

Forget gate developed by Gers et al. (2000), controls which elements of the cell state vector C_{t-1} will be forgotten(to which degree):

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{3.3}$$

where f_t is a resulting vector with values in the range (0,1), $\sigma(.)$ denotes the logistic sigmoid function and W_f U_f and b_f stand for the set of learnable parameters for the forget gate, i.e. two versatile weight matrices and a bias vector. In the following step, an update vector for the cell state is calculated from the current input (x_t) and the last hidden state (h_{t-1}) illustrated in this equation:

$$f_t = \tanh(W_{\tilde{c}} x_\mathrm{t} + U_{\tilde{c}} h_{t-1} + b_{\tilde{c}}), \tag{3.4}$$

where $\tilde{c}$ is a vector with values in the range (-1, 1),tanh(.) is the hyperbolic tangent and $W_{\tilde{c}}$, $U_{\tilde{c}}$, $b_{\tilde{c}}$ are another set of learnable parameters. In addition, the second gate is compute, the input gate, depicting which (and to what extent) information of $\tilde{c}_t$ is passed down to update the cell state in the current time step:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{3.5}$$

where i_t is a vector with values in the range of (0,1) and W_i, U_i, b_i are a set of learnable parameters, defined for the input gate. With the results of Eqs. (2)–(4) the cell state c_t is updated by the subsequent equation:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{3.6}$$

where $\odot$ represents element-wise multiplication. Since the vectors f_t and i_t have both inputs in the range (0, 1), Eq. (5) can be inferred in the way that it determines, which information stored in c_{t-1} will be forgotten (values of f_t of approx. 0) and which will be stored (values of f_t of approx. 1). Likewise, i_t determines which new information stored in $\tilde{c}_t$ will be added to the cell state(values of i_t of approx. 1) and which will be disregard (values of i_t of approx. 0). Similar to the hidden state vector, the cell state is initialized by a vector of zeros in the first time step. Its length match up the length of the hidden state vector.

The third and last gate is the output gate, which inspect the information of the cell state c_t that runs into the new hidden state h_t. The output gate is computed by the next equation:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{3.7}$$

where o_t is a vector with values in the range (0,1), and W_o, U_o,b_o are are a set of learnable parameters, assigned for the output gate. From this vector, the new hidden state h_t is computed by combining the results of Eqs. (5) and (6):

$$h_t = \tanh(c_t) \odot o_t. \tag{3.8}$$

The cell state c_t in particular makes it possible to learn long-term dependencies effectively. Owing to its relatively simple linear interactions with the remaining LSTM cell, it can retain information unchanged over a long period of time steps. During training, this characteristic aids to avoid the problem of the exploding or vanishing gradients in the backpropagation step [32]. In addition, we can stack several layers on top of one other. The output from the final LSTM layer at the last time step (h_t) is connected through a dense layer to a single output neuron, which calculates the last streamflow prediction. The following equation gives the dense calculation:

$$y = W_d h_n + b_d, \tag{3.9}$$

where y represents the final streamflow, h_n denotes the output of the final LSTM layer in the last time step inferred from Eq. (3.7), W_d is the weight matrix of the dense layer, and b_d is the bias term. Finally, the pseudocode of the whole LSTM layer is given in the following algorithm:

$$h_t = g(Wx_t + Uh_{t-1} + b), \tag{3.10}$$

Where g(.) denotes the activation function, W and U are moveable weight matrices of the hidden state h and the input x, and b represents a bias vector. The inputs for the entire sequence of meteorological observations $x = [x_1, ..., x_n]$, where x_t is a vector comprising the meteorological inputs of time step t, is analyzed time step by time step and in each time step Eqs. (3.2)–(3.7) are repeated. When there are several stacked LSTM layers, the next layer gains the output $h = [h_1, ..., h_n]$ of the first layer as input. The final output, the streamflow, is then computed by Eq. (3.9), where h_n is the final output of the last LSTM layer.

2.2.3.1 The LSTM's parameters As tuning of the model parameters is very critical to ensure the optimal structure and thus highest prediction accuracy, therefore, before training this network, we first search for the optimal hyperparameters. While there are other methods to effectively search for plausible hyperparameters, the hyperband optimizer from the KerasTuner[48] has several advantages for our setting: it offers a higher speed up, it is a general-purpose technique that makes minimal assumptions contrary to prior configuration evaluation approaches [49]. Hyperband uses early-stopping and adaptive resource allocation to speed up the hyperparameter tuning process on a high-performing model. The algorithm trains a large number of models for a few epochs and carries forward only the top-performing half of the models to the next round. Hyperband determines the number of models to train in the next phase by computing $1 + log_{factor}(\texttt{max\_epoch})$ and rounding it up to the nearest integer [50, 49].

first, a tuner was defined to determine which hyperparameter combinations should be tested. We then trained 30 hyperband trials (one for each station) of optimizing the following parameters:

- number of layers between 1 to 4 layers
- input unit: the range is from 32 to 512 inclusive. When sampling from it, the minimum step for walking through the intervals is 32.
- hidden layers units: the range is from 32 to 512 inclusive. When sampling from it, the minimum step for walking through the intervals is 32.
- last layers neurons: the range is from 32 to 512 inclusive. When sampling from it, the minimum step for walking through the intervals is 32.
- dropout rate: the range is between 0 to 0.5 inclusive. When sampling from it, the minimum step for walking through the intervals is 0.1.
- dense layer's activation: to sample between Rectified Linear Unit (Relu) and Hyperbolic Tangent Function (Tanh) while default value is set to Relu.

For the remaining parameters, alternative but computationally efficient approaches were preferred. These parameters include optimizer and number of epochs. Adam optimizer was chosen based on previous literature on streamflow prediction [9], however learning rate was decided using the hyperparameter tuning approach as explained above. Regarding the number of epochs, in order to avoid overfitting and training the model with a high epoch, we used the Early Stopping method. We trained the model for 1000 epochs with the batch size set to 32. Once using early stopping, during training, the model is evaluated on a holdout validation dataset (the last 20% of the training set) after each epoch. If the performance of the model on the validation dataset decreases after a certain time (here specified to be 7 epochs), the training process is stopped.

2.2.3.2 Ensemble runs Neural networks use a gradient-based method to optimize the loss function. Since the networks allow for local minima, different initial weights can potentially lead to different models with different performances. In order to overcome this problem, we run the LSTM model 10 times and report an ensemble distribution of NSE and KGE values. In order to show the variance in the model performance, the Standard Deviation (SD) of NSE and KGE over 10 runs is reported. Hence, results and conclusions are based on the statistical distribution of model performance across the ensemble runs. Moreover, to ensure the accuracy of prediction, each of these 10 models makes a prediction and the mean of all predictions is reported as the final prediction.

2.2.4 Multiple Linear Regression

Multiple Linear Regression (MLR) introduced by Galton [51] and developed by Pearson [52] rule out nonlinear relationships between input and output variables. The equation of this algorithm is shown in Eq. 3.10.

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon. \tag{3.11}$$

Where y, is the predicted value of the dependent variable, n refers to the number of predictors, $\beta_1 x_1$ is the regression coefficient β_1 of the first independent variable x_1. $\beta_n x_n$ represents the regression coefficient β_n of the last independent variable x_n and ϵ is the model error.

In this study, we trained two MLRs with different input variables. In the first one, we refer to as `MLR_s`, we fed the meteorological variables of "t", "et", and "p" to the model as predictors and defined streamflow as the dependent variable. Wheres in the second model, we refer to it as `MLR_lag`, in addition to the meteorological variables, the lagged variables of them were included.

2.2.5 Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting is a tree-based algorithm that makes a prediction model in the form of a group of weak prediction models. The training procedure creates a tree structure, where every leaf node refers to a predicted value. Every path to a leaf reflects conjunction of input features that outcome the prediction of the leaf value. The final prediction is then calculated as the sum of each tree's prediction (depicted in Eq. 10) [20].

$$\hat{y}_i = \sum_{k=1}^{K} f_k(X_i), f_k \in F, \tag{3.12}$$

$F = f(x) = w_{q(x)}(q : R^m \rightarrow T, w \in R^T$ represents the space of regression trees and q stands for the structure of each tree that maps a leaf index with an example. T is how many leaves there are on the tree. Each f_k refers to an independent tree structure and leaf weights w.

Alike the LSTM model, before training this model, we first search for the optimal hyperparameters. To do so, the GridSearchCV function from the Scikit-learn library was used to optimize the following parameters [53]:

- learning rate: sampling from [.01, .05, .1]
- maximum tree depth per tree(`max_depth`) : sampling from [7, 9]
- the fraction of observations to be sampled for each tree (subsamples) :sampling from [0.8, 0.9]
- the fraction of columns to be sampled randomly for each tree (`colsample_bytree`) : sampling from [0.7, 0.9]
- the number of trees in our ensemble(`n_estimators`): sampling from [800, 1000]

A Grid Search is an exhaustive search over every combination of specified parameter values[53, 22]. For example, if 2 possible values are specified for one hyperparameter and 3 possible values are specified for another one, Grid Search will iterate over 6 possible combinations. That is to say, Grid Search creates several models, each with a different combination of hyper-parameters. Each of these combinations of parameters, which correspond to a single model, can be said to lie on a point of a "grid". The goal is then to train each of these models and evaluate them. To do so, Grid Search uses Cross-validation (CV) in the evaluating phase. The best-performing model in the evaluation phase determines the best combination. An advantage of Grid Search is that by brute-forcing all possible combinations, it ensures finding the best combination of hyperparameters.

In our case, we defined a 5-fold cross-validation. i.e. we had 5 folds for each of 48 candidates which resulted in totaling 240 fits (one for each station). As for `min_child_weight` parameter, its value has been defined manually. `min_child_weight` refers to the minimum sum of instance weight needed in a child. By default, its value is set to one. If the tree splitting results in a leaf node with the sum of instance weight less than `min_child_weight`, then the tree forming process will continue. As a result, a larger value assigned to this parameter makes the algorithm more conservative[54]. This parameter can be used to control over-fitting. Higher values restrict a model from learning relations that might be very specific to the particular sample selected for a tree. In our setting to prevent over-fitting, we specified it to be 4.

Similar to the MLR model, we trained two XGBoost models with different input variables. In the first one, referred to as `XGB_s`, we fed meteorological variables of ”t”, ”et”, and ”p” to the model as predictors and streamflow is the target variable. Wheres in the second model, referred to as `XGB_lag`, in addition to these meteorological variables, the lagged variables of them were included. Noteworthy, to have comparable results to Shen et al. paper, we fed all the variables in the model and no method regarding variable selection was performed.

2.2.6 Model setup and evaluation

As mentioned above, two different setting of input variables were explored in non-time-series models. Different models and their configurations are explained in table 3.

Abbreviations of models name	Concurrent meteorological variables	Lagged meteorological variables
LSTM	Yes	Yes
XGB_s	Yes	No
XGB_lag	Yes	Yes
MLR_s	Yes	No
MLR_lag	Yes	Yes

Note: LSTM model takes a sequence of input data and this is denoted by lagged variables for this specific model here

Table 3.1: Overview of models and their different setups

Model performance is quantified by the Nash-Sutcliffe model efficiency (NSE) and the Kling-Gupta efficiency (KGE). Both metrics are widely used in the hydrology domain but the latter offers a better indication of streamflow seasonality [10]. Hence, in determining the time-lag value, we considered the KGE value, however, both values were reported.

Equations (11) and (12) show how the coefficients are calculated,

$$
\begin{aligned}
NSE = 1 - \frac{\sum_{t=1}^{T}(q_m^t - q_o^t)^2}{\sum_{t=1}^{T}(q_o^t - \hat{q}_o)^2} = 1 - \frac{MSE}{\sigma_o^2} \\
MSE = \frac{1}{n}\sum_{t=1}^{T}(q_m^t - q_o^t)^2 \\
\sigma_o^2 = \frac{1}{n}\sum_{t=1}^{T}(q_m^t - \hat{q}_o)^2
\end{aligned}
\tag{3.13}
$$

$$
\begin{aligned}
KGE = \sqrt{(r-1)^2 + (\alpha-1)^2 + (\beta-1)^2}, \\
r = cov\frac{(q_m, q_o)}{(\sigma_m.\sigma_o)} \\
\alpha = \frac{\sigma_m}{\sigma_o} \\
\beta = \frac{\mu_m}{\mu_o}
\end{aligned}
\tag{3.14}
$$

where q represents streamflow with a unit of $\frac{m^3}{s}$ in daily time step t, T refers to the total number of days, σ is the standard deviation and μ is the average with subscript o depicting observations and subscript m represents the predictions made by the model.

Chapter 4

Results

Here, we discuss the optimal time-lag value, the structure of the better-performing models discussed in the previous sections and, the performance of each model and its configuration for the test set.

1 Time-lag value

The figures below reflect the correlation between the input variables and streamflow in the Basel and Lobith stations up until lag 200.

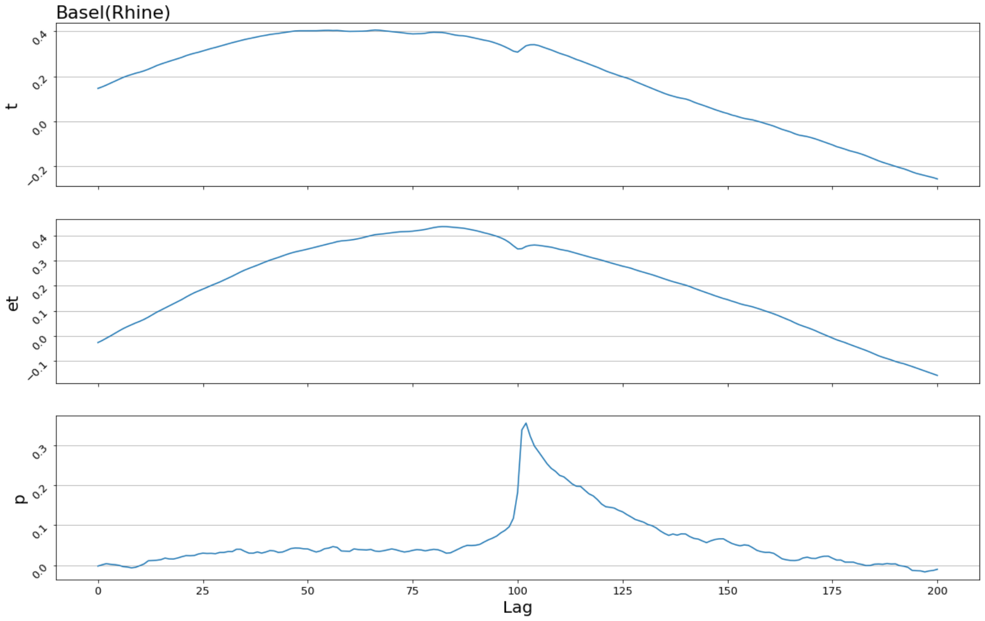

Figure 4.1: cross-correlation (CC) of the predictors and streamflow of Basel station

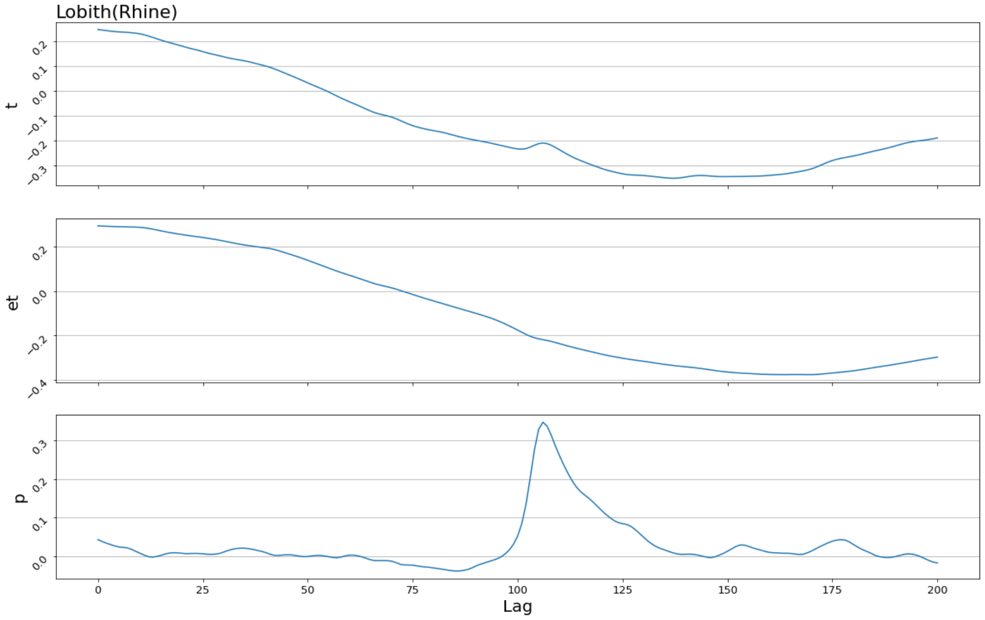

Figure 4.2: cross-correlation (CC) of the predictors and streamflow of Lobith station

We attained the highest correlation period, which we refer to as the correlation period, for each station by checking the figures manually. As for Basel station, between 50 to 110 lags were considered as the correlation period. All 3 predictors have a positive correlation with streamflow in this period, while the correlation of ”t” and ”et” variables with streamflow is at its highest values within this period. Whereas for the Lobith station, considering all predictors, the optimal time-lag value lies between 10 to 70 lags.

But a question soon arises, which lag within this period is the optimal time-lag value? To answer this question, we trained different LSTM models with the same structure but different time windows. As for Basel station, the models were trained with twelve-time windows including 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, and 205. As mentioned in the method section, we trained each model 10 times for each of these time windows and calculated the ensemble value of KGE and NSE. The standard deviation over 10 times running model can be seen in figure 4. Regarding Lobith station, the models were trained with fourteenth-time windows including 10, 15, 25, 35, 45, 55, 60, 65, 70, 75, 80, 90, 100, and 205. Although the correlation period was mentioned to guide us in selecting time windows, we looked at lower and higher values to fact-check our approach. Figure 3 shows the ensemble values of KGE and NSE resulting from examining different time windows.

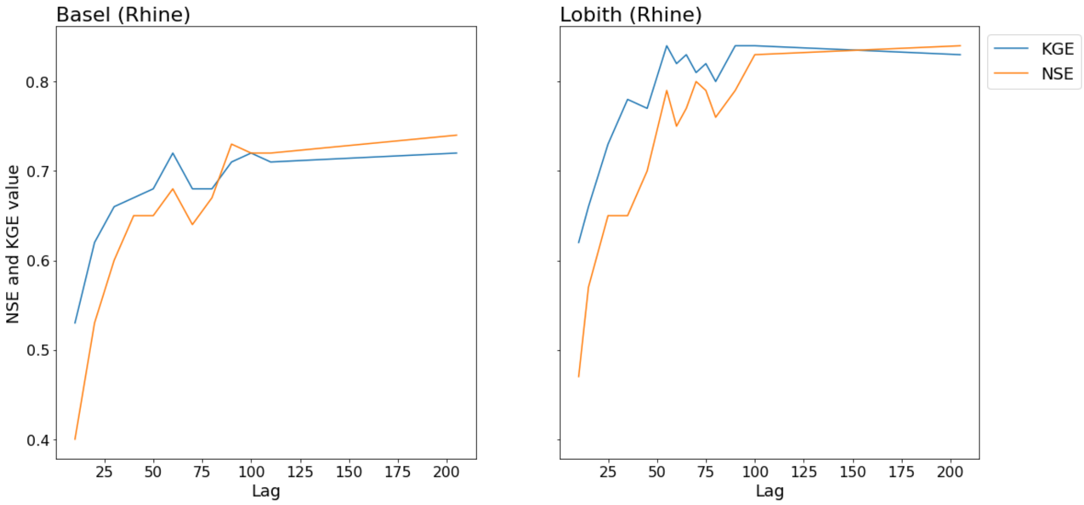

Figure 4.3: The ensemble values of KGE and NSE resulted from training the LSTM model with different time windows

As shown in figure 3, the highest KGE value for Basel was gained at lag 60, whereas for Lobith it was earned at lag 55. The same plot with standard deviation across a 10-time running model is shown in Figure 4.

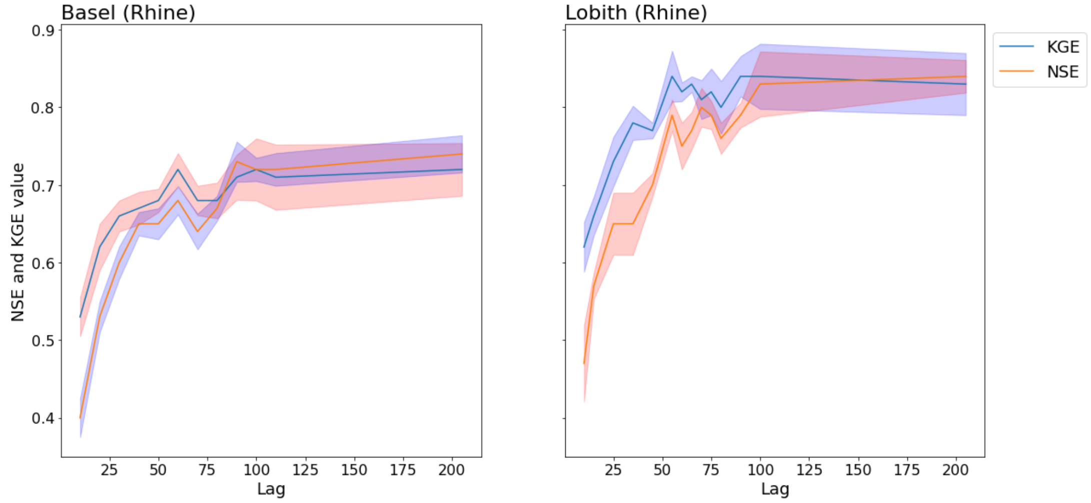

Figure 4.4: The ensemble values of KGE and NSE and their standard deviation (SD) resulted from training the LSTM model with different time windows. The SD is shown with a shadow around each line.

2 Architecture of the optimal LSTM network

The LSTM model was optimized using Keras Tuner for each station separately. The hyperparameters tuned and their corresponding values are listed in table 1 in the Appendix. To provide an overall view of the architecture of the optimal LSTM network (i.e.from the tuned hyperparameters and those that were chosen using other methods) tables (1, 2) were created.

Input	Previous 2-month daily meteorological variables of Basel station
Output	Daily streamflow at Basel station
Time period	From first January 1981 to 31st December 2000
Dataset distribution	Training set (75%), validation set (15%), testing set (10%)
Structure	A visible layer by 128 input unit + a hidden layer + second layer with 128 neurons + drop out layers at each layer with 0.1 rate + Dense layer with one output
Activation function	Rectified Linear Unit (Relu)
Learning Rate	0.001
Loss function	Mean Square Error (MSE)
Number of epochs	1000 but depending on the condition to cease training
Batch size	32

Table 4.1: Information regarding the structure of the better performing LSTM network in Basel station

Input	Previous 55 daily meteorological variables of Lobith station
Output	Daily streamflow at Lobith station
Time period	From first January 1981 to 31st December 2000
Dataset distribution	Training set (75%), validation set (15%), testing set (10%)
Structure	A visible layer by 64 input unit + two hidden layers + second layer with 480 neurons + drop out layers at each layer with 0.1 rate + Dense layer with one output
Activation function	Hyperbolic Tangent Function (tanh)
Learning Rate	0.001
Loss function	Mean Square Error (MSE)
Number of epochs	1000 but depending on the condition to cease training
Batch size	32

Table 4.2: Information regarding the structure of the better performing LSTM network in Lobith station

3 Architecture of the optimal XGBoost model

As explained in the method section, both configurations of the XGBoost model,XGB_s and XGB_lag were optimized using Grid Search for each station separately. The hyperparameters tuned for the XGB_s model and their corresponding values are listed in table 4 and 5.

XGB_s Parameter	Configuration
colsample_bytree	0.7
learning_rate	0.01
max_depth	7
min_child_weight	4
n_estimators	800
subsample	0.9

Table 4.3: Information regarding the optimal XGB_s hyperparameters of Basel

XGB_s	Configuration
colsample_bytree	0.7
learning_rate	0.01
max_depth	7
min_child_weight	4
n_estimators	800
subsample	0.8

Table 4.4: Information regarding the optimal XGB_s hyperparameters of Lobith

Similarly, we performed Grid Search for XGB_lag and the results are shown in the tables below. Interestingly, the optimized parameters of XGB_s and XGB_lag are identical in both stations. That is to say, adding lagged variable did not change the optimal hyperparameters for Basel and Lobith stations. Moreover, the set of hyperparameters for both stations are identical and the only difference is the *subsample* hyperparameter which is 0.9 for Basel and 0.8 for Lobith.

XGB_lag Parameter	Configuration
colsample_bytree	0.7
learning_rate	0.01
max_depth	7
min_child_weight	4
n_estimators	800
subsample	0.9

Table 4.5: Information regarding the optimal XGB_lag hyperparameters of Basel

XGB_lag	Configuration
colsample_bytree	0.7
learning_rate	0.01
max_depth	7
min_child_weight	4
n_estimators	800
subsample	0.8

Table 4.6: Information regarding the optimal XGB_lag hyperparameters of Lobith

4 Performance of the models

The performance of each model and its configuration are shown in figure 4. Among models discussed in this study, the LSTM outperformed other models for the Lobith station with an NSE value of 0.79 and KGE value of 0.84. Regarding Basel station, this network has a good performance but shares the first place with the MLR_lag model. Both models scored 0.68 in NSE and 0.72 in KGE. This is somewhat surprising, as the linear model said not to capture the non-linear relationship of the predictors and streamflow.

Interestingly, including lagged variables in the non-time-series model considerably improved their performance. The NSE value for Basel station improved from 0.12 to 0.68 and the KGE improved from 0.12 to 0.72. The same holds for Lobith station. The performance of MLR_lag is significantly higher than the performance of MLR_s. When including lagged variables NSE value improved from 0.037 to 0.67 and KGE improved from -0.047 to 0.76. Similarly, the XGBoost model performance improved when adding lagged input variables. Regarding Basel station, NSE improved from 0.15 to 0.61 and KGE improved from 0.21 to 0.60. In the case of Lobith station, NSE improved from 0.029 to 0.69 and KGE boosted from 0.094 to 0.67. However, in the end, the LSTM network trained on the same time window (55) outperformed non-time-series models in the case of the Lobith station, the change in the performance of non-time-series models shows that once time series information is introduced to them, they are capable of discharge prediction and in some cases, they may reach the RNNs performance such as LSTM.

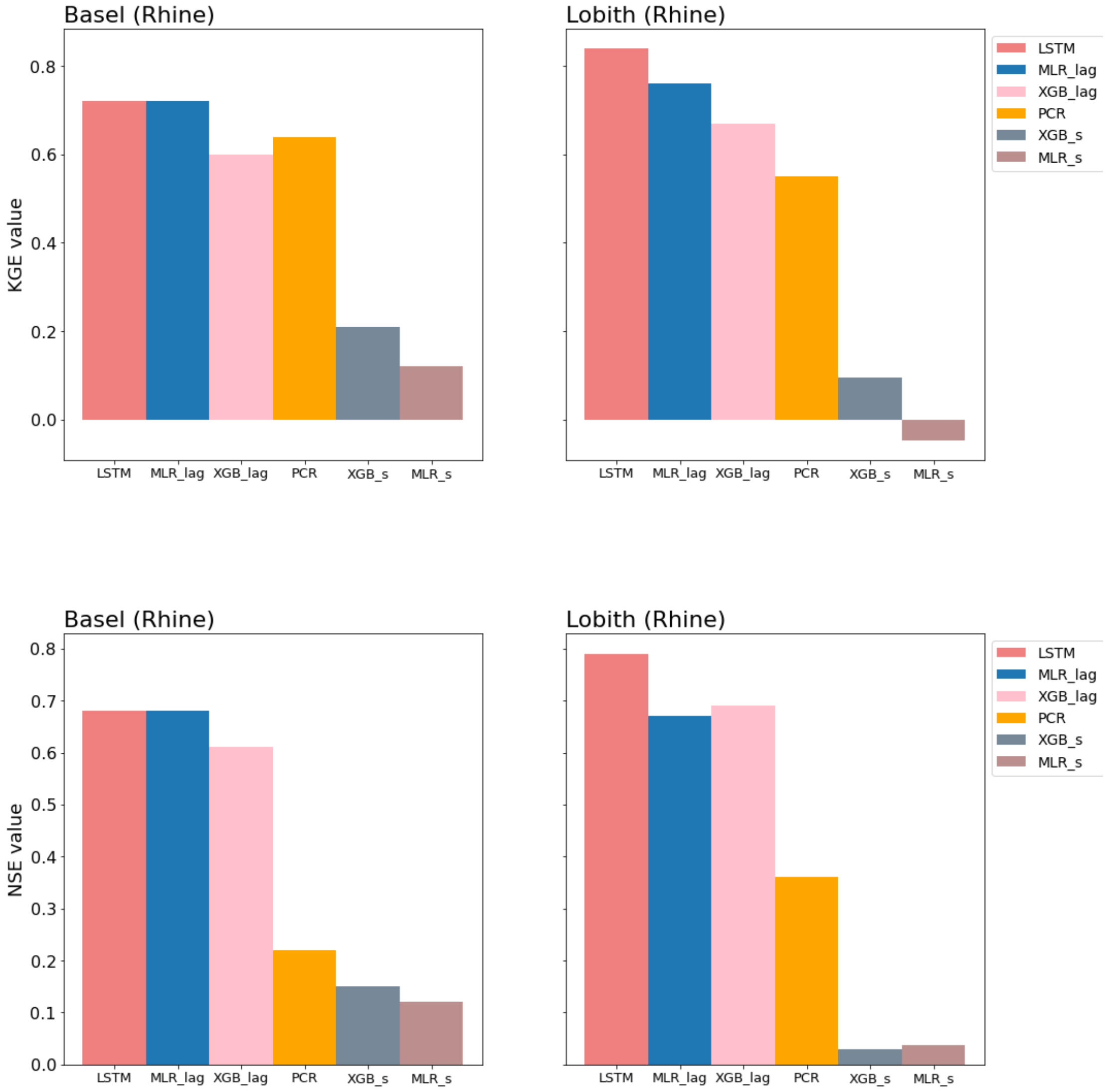

Figure 4.5: Model performance at different locations for the validation period (1996–2000)

In both stations, the LSTM, XGB_lag and MLR_lag models outperformed the PCR-GLOBWB model. Among these models, the LSTM outperformed other models for the Lobith station and performed equally well as the MLR_lag model in Basel station. Information regarding coefficients of MLR models is provided in the Appendix.

Finally, the standard deviation of ensemble values of NSE and KGE for the LSTM model at Lobith Station are 0.026 and 0.042, respectively. In the Basel station, the standard deviation of the NSE is 0.024, whereas KGE's is 0.035. It indicates the model is relatively stable.

Figures 5 and 6 show the time series of discharge and the LSTM model prediction along with PCR-GLOBWB streamflow prediction.

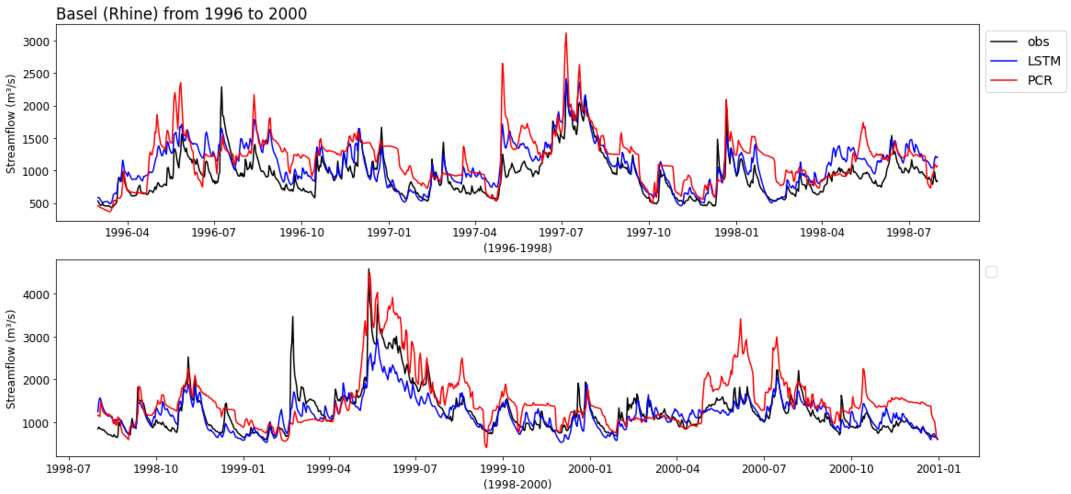

Figure 4.6: Performance of the LSTM model compared to PCR on test set in Basel station

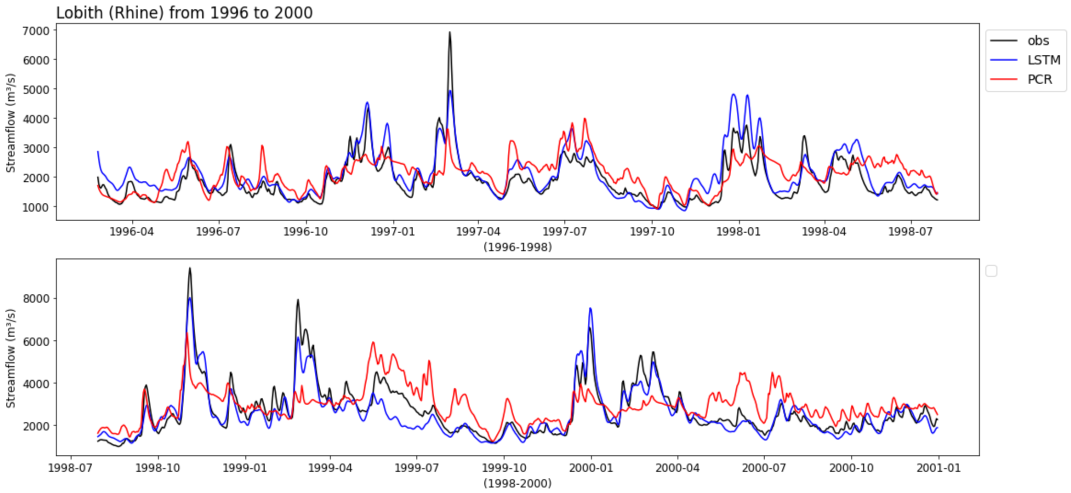

Figure 4.7: Performance of the LSTM model compared to PCR on test set in Lobith station

Figures 7 and 8 show the time series of discharge and the `XGB_lag` model prediction along with PCR-GLOBWB streamflow prediction.

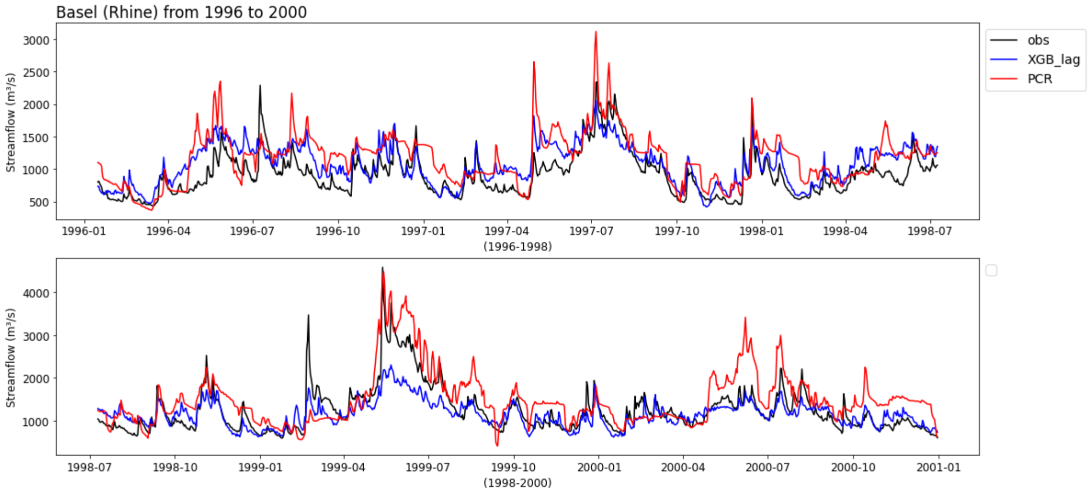

Figure 4.8: Performance of the `XGB_lag` model compared to PCR on test set in Basel station

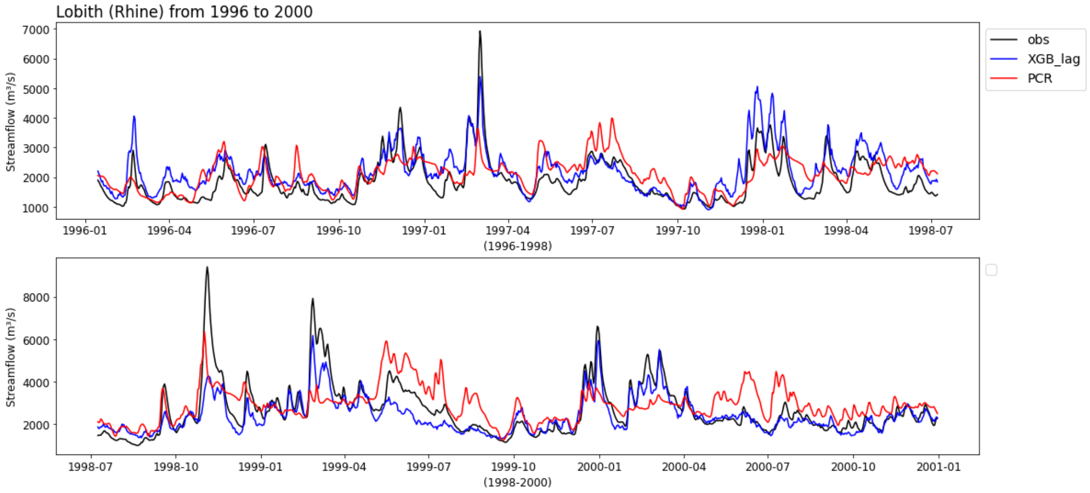

Figure 4.9: Performance of the `XGB_lag` model compared to PCR on test set in Lobith station

5 Performance consistency

In the context of the climate change, performance consistency is an important factor beside performance accuracy. A model that can predict well high and low streamflow can increase flood and drought control. To obtain insight into the consistency of each model's performance and compare them to the PCR model, their cumulative frequency curves are plotted against actual observations over the test period in figures 9 and 10. Regarding Basel station, the performance consistency of LSTM is higher than other ML models, and in this case, it outperforms the `MLR_lag` model for two reasons: first, it can predict streamflow values higher than 2200 up to 3000 m^3/s. Second, it can capture very low streamflow values, up to approximately 500 m^3/s. The only model that can capture streamflow higher than 3000 m^3/s values in the Basel is the PCR model. However, this model performs very well at low values of streamflow up to 500 m^3/s (the LSTM does well here as well) and high values but it fails to capture the trend in the normal range of streamflow (between 500 to 3200 m^3/s) where most of the data falls in.

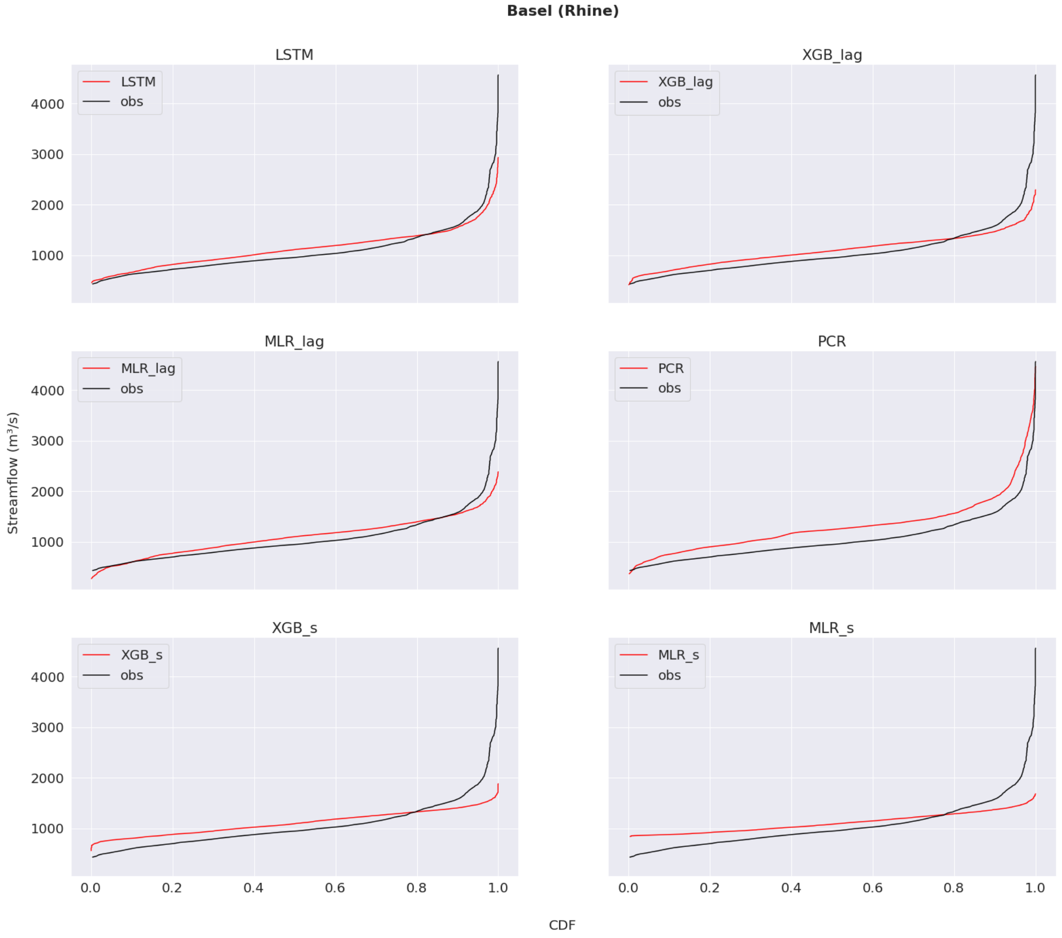

Figure 4.10: Cumulative Frequency Function (CDF) of the observations compared to the performance of ML and the PCR model in Basel station over test period (1996-2000).

Regarding Lobith station, the LSTM model outperforms other ML models in addition to the PCR model. The consistency of its performance in predicting streamflow in all low, high and normal streamflow periods is impressive. It is notable that in this station, the LSTM model is the only model to fit well during high streamflow. It is also interesting to see that in this case, `XGB_lag` performs more consistently than `MLR_lag` overall and especially in low streamflow.

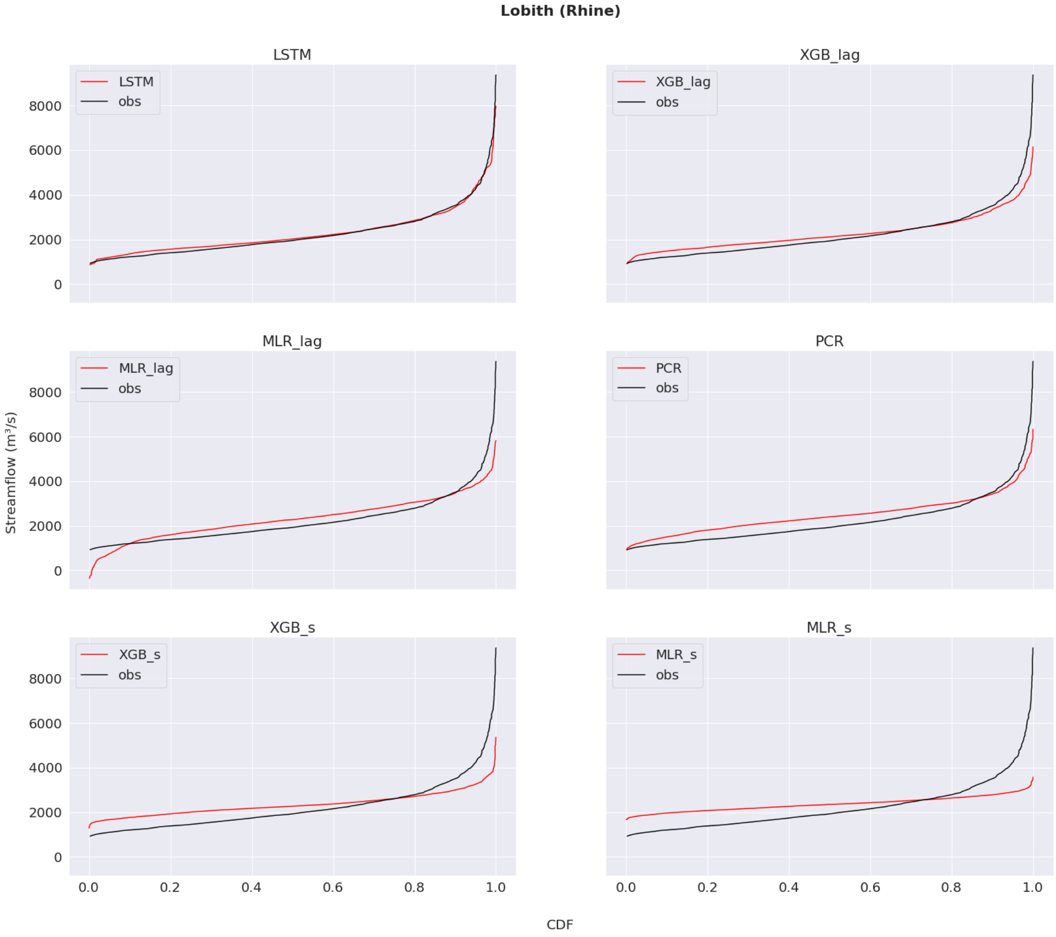

Figure 4.11: Cumulative Frequency Function (CDF) of the observations compared to the performance of ML and the PCR model in Lobith station over test period (1996-2000).

Overall, in both locations, the LSTM is the superior model ML model in performance consistency. Performance consistency is also visible in the observed against predicted streamflow plots provided in the Appendix section.

Chapter 5

Conclusion and Discussion

1 Discussion

Previous works employed ML models in the streamflow prediction but in most cases, it is based on the historical records of streamflow or simulation variables from a hydrological model. A few studies employed ML models in predicting streamflow based on solely meteorological variables. We, therefore, aimed to predict streamflow from three input meteorological variables using LSTM, and XGBoost models in Basel and Lobith stations. To determine if the use of these ML models is advantageous, we benchmarked them to two other models: (i) MLR as an ML benchmark (ii) and the PCR-GLOBWB as a hydrological benchmark.

We found that using input meteorological data, the LSTM model performs well in predicting streamflow. It outperforms other models in the Lobith station and shares a similar performance with the lagged version of the MLR model. The XGBoost model, on the contrary, works well only when lagged predictor variables are added to the model. Similarly, the `MLR_lag` performance increased significantly compared to the `MLR_s` in both locations such that it achieved a similar accuracy as the LSTM model in the Basel. The `MLR_s` and `XGB_s` models perform poorly and can not achieve KGE or NSE of greater than 0.2 in both stations. Interestingly, in the case of the Lobith station, non-time-series models advanced more when adding lagged predictor variables.

Regarding performance consistency, the LSTM is the superior model in both locations among ML models. In Basel station, for example, the LSTM and `MLR_lag` share the same accuracy but the LSTM performance is much better in low and high streamflow. It also constantly performs well in different values of streamflow in the Lobith. In the context of climate change, it is of vital importance to have a model that is capable of predicting high and low flow accurately to prevent extreme events such as floods and drought and the LSTM model proved to be a good fit in this case.

Surprisingly, including lagged variables in the MLR model results in better performance accuracy than the XGBoost model in both locations. However, in the case of Lobith station `XGB_lag` has a better performance consistency overall and in particular in low streamflow.

We conclude that although the `MLR_lag` performs rather well in terms of accuracy, it struggles in capturing high and low streamflows. A hydrological system responds to its drivers in a relatively simple manner. While not linear, it is also not overly chaotic. Thereby, including a high number of input lagged variables helps the `MLR_lag` model to pick up some variation but eventually it fails to predict high and low streamflow.

Shen et al. (2022) found a similar result using the same data. They show that by employing historical meteorological data the RF model can correct the predictions from a simulation model in the study area. As they only used 10 input lagged variables and assigned a smaller training set to the RF model than that of us, no definite conclusion can be drawn from comparing the results of these two studies. Nonetheless, the NSE and KGE values in the Lobith station improved from 0.72 to 0.79 and from 0.82 to 0.84 respectively as a result of this study. In Basel station, this was not the case. However, the NSE value improved from approx 0.53 to 0.68, KGE value gained by Shen et al. is higher (from 0.72 to 0.76). Also, as indicated earlier, we discovered that introducing the lagged predictor variables in the Lobith station improved non-time-series models further than the Basel. It might be due to the larger catchment of Lobith station compared to the Basel. Discharge from larger catchments responds slower to drivers as the travel time to the outlet of the catchment is longer for streamflow. Therefore, adding higher lagged predictor variables is more crucial here.

In comparison to the scenario of combining input simulation variables for the error correction procedure in Shen et al. paper, the NSE value in the Basel station is slightly higher and the KGE value is slightly lower than their results. Our results in the Lobith station are slightly lower than those in Shen et al. paper. Even though our results are less favorable, our approach demonstrates that adding optimal time-lag values as predictors may be an alternative to including the simulation predictor variables in ML models. Given the limitations of a hydrological model, the performance-efficiency trade-off should be taken into account when including input simulation variables. Future research is needed to further shed light on the application of hydrological models. The result of this study can serve as a benchmark for a scenario of adopting an updating procedure with additional simulation input variables using ML models with the same architecture to improve PCR-GLOBWB prediction.

Moreover, drawing on the success of streamflow predictions solely based on historical meteorological variables for gauged catchment, our approach has the potential for efficiently estimating streamflow at ungauged locations. In this potential application, the ML model such as LSTM or lagged version of MLR or XGBoost should be first trained by data from other stations with different characteristics so that the model can be generalized. This generalized model, then, can be used for streamflow prediction at ungauged stations.

2 Conclusions

Our research demonstrates that a time series RNN model, such as LSTM, performs convincingly in streamflow prediction given only input meteorological information. However, if a non-time-series ML model is used since it is typically more computationally efficient and less data-hungry, then adding the ideal quantity of lag variables dramatically improves prediction accuracy. The performance consistency may not be as good as when a time series RNN model is used, though. The most important thing is to determine the ideal time-lag value. This is crucial because including the optimal time-lag input predictors in ML models might be an alternative solution to including the simulation predictor variables in ML models from a hydrological model.

Additionally, the fact that the LSTM model outperforms the XGBoost in every way, demonstrates the number of data points present here (of the 7305 total data points, 75% were assigned to the training set) is enough for an LSTM model to produce decent results. Future research is required to evaluate the performance of these models with smaller data sets.

3 Limitations

In this study, we did not define the optimal time-lag value individually for each predictor, rather we selected a period in which all predictors seemed to have a positive correlation with streamflow. A predictor-specific time lag might decrease the complexity of the model and result in a better performance.

Moreover, in this study despite using Google Colab Pro to speed up the process, we were limited in three areas because of a lack of computational power and time.

1- We ran the LSTM models 10 times to take into account the stochastic nature of this model. It is while experts advise running a deep network between 30 to 100 times.

2- We were limited in hyperparameter tuning and a limited set of search space was chosen. A wholesome hyperparameter tuning will greatly affect the performance accuracy.

3- Investigating higher time-lag values required running an LSTM model 10 times with different time windows in this study. As the bigger the time window, the longer the training phase, it was not feasible for us to look for higher lags than 205 days. Investigating higher lags is highly recommended in future studies.

4 Code availability

the code used in this study, can be found at this repository on GitHub.

Chapter 6

Appendix

1 Time series plots

Figures 1 and 2 show time series of observed streamflow and the predictors.

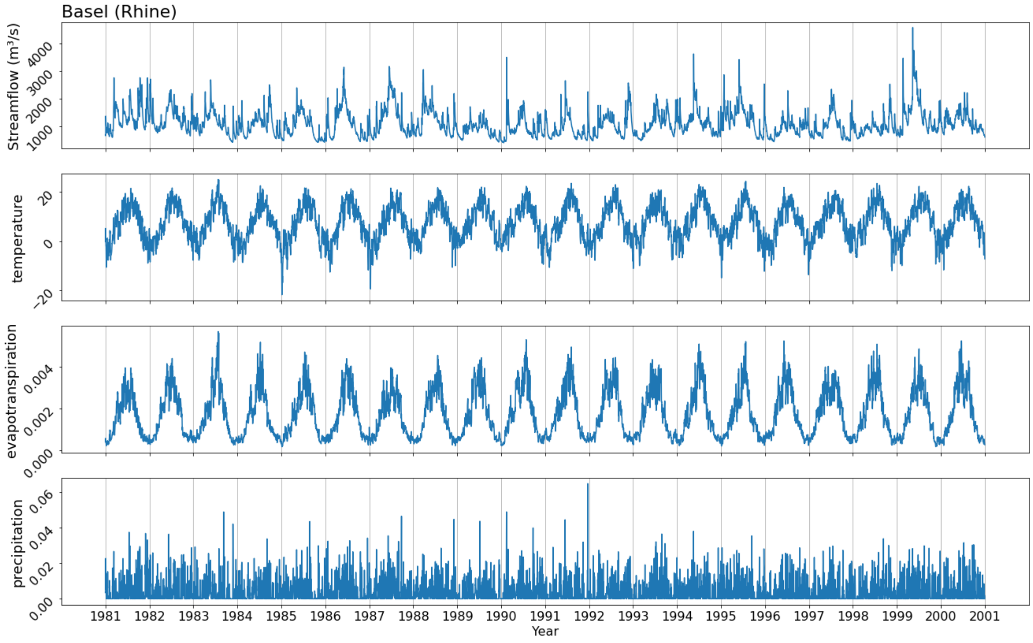

Figure 6.1: Time series of the observed streamflow and the predictors in Basel station between 1981 to 2000

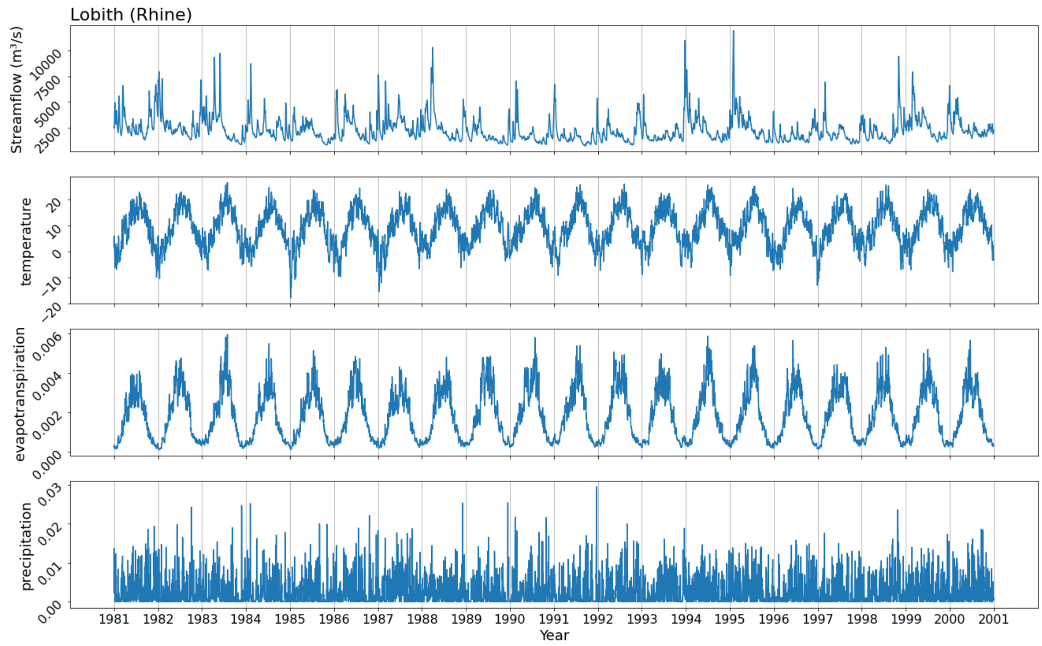

Figure 6.2: Time series of the observed streamflow and the predictors in Lobith station between 1981 to 2000

2 Hyperparametes tuning of the LSTM model

Table 1 shows hyperparametes tuning for the LSTM model in both locations.

Network Parameter	Configuration
Number of layers	1
Number of input units	128
Number of neurons in second layer	128
Number of units in layers	192, 128, 128, 352
Dropout rate	0.1
Learning rate	0.001
Activation function	relu

(a) Information regarding the optimal hyperparameters in Basel station

Network Parameter	Configuration
Number of layers	2
Number of input units	64
Number of neurons in second layer	480
Number of units in layers	192 , 448, 288, 352
Dropout rate	0.1
Learning rate	0.001
Activation function	tanh

(b) Information regarding the optimal hyperparameters in Lobith station

Table 6.1: The optimal hyperparameters of the LSTM

3 Model performance

Scatter plots of model performance are shown in the Figures 3 and 4.

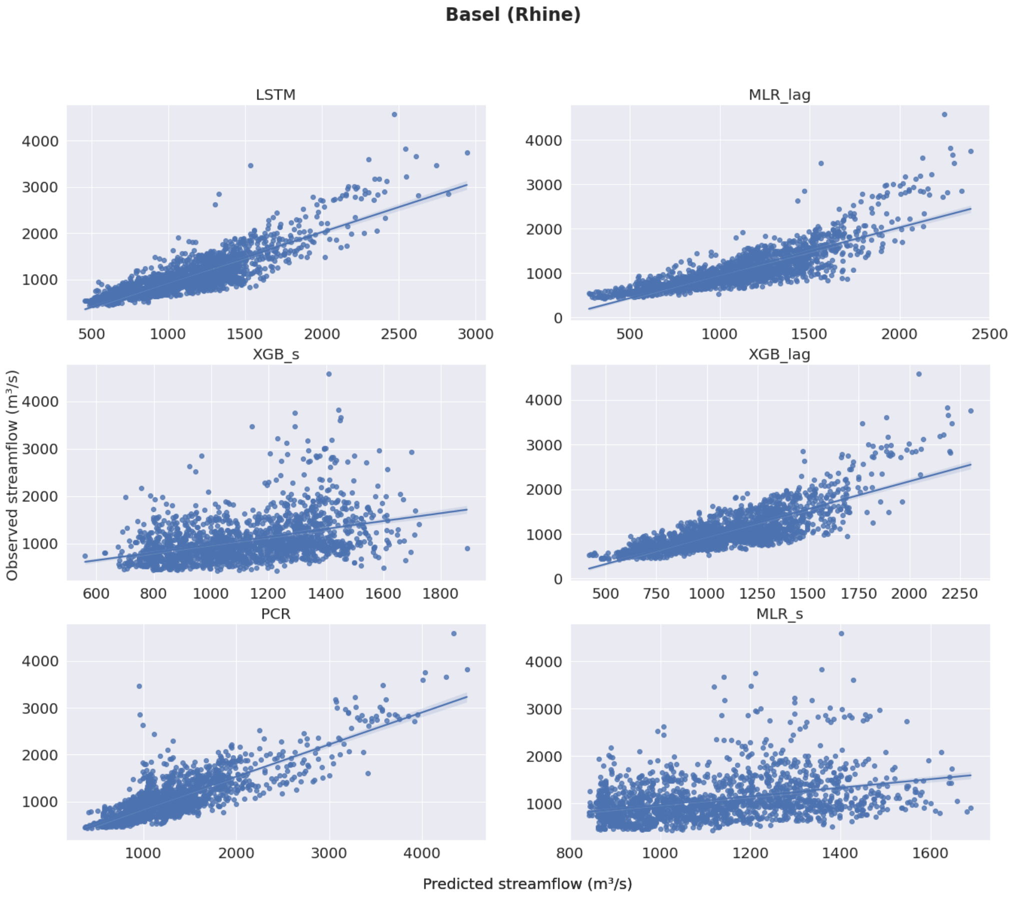

Figure 6.3: observed streamflow against predicted streamflow in different models in Basel station

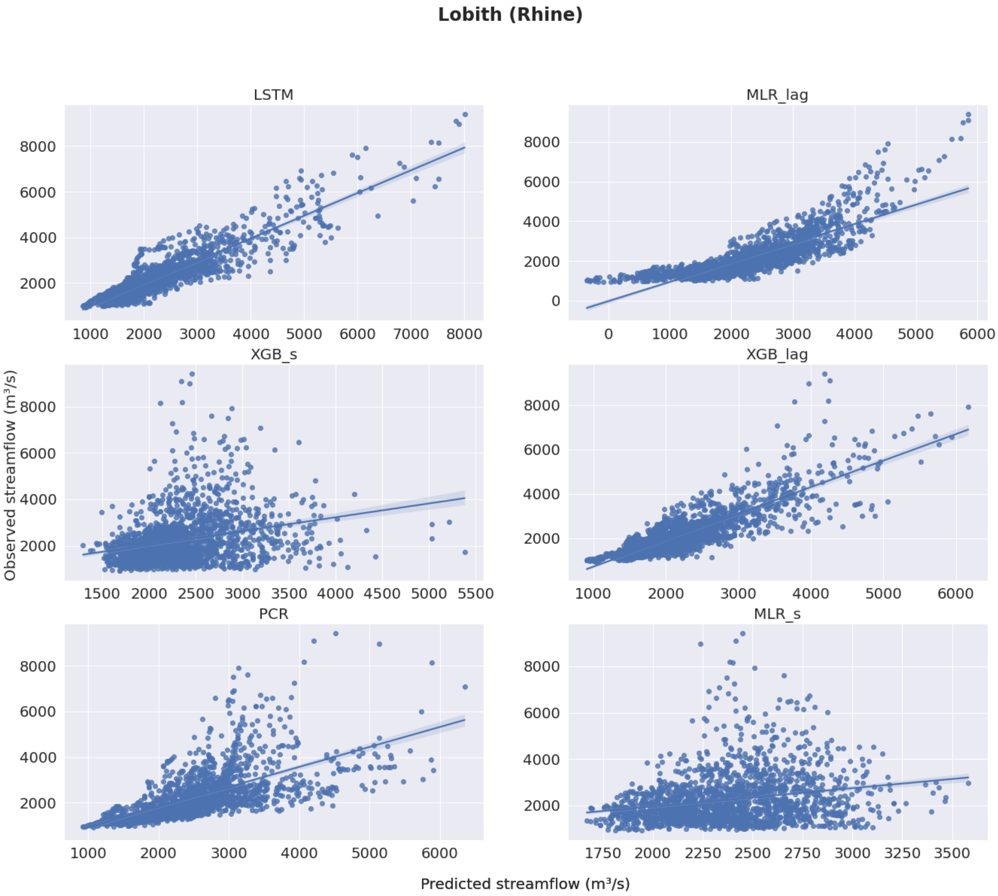

Figure 6.4: observed streamflow against predicted streamflow in different models in Lobith station

4 Feature importance

In this section, we will explore the coefficients of the MLR models to get an idea about feature importance. As seen in the previous section, the `MLR_lag` model has satisfactory results and we are specifically keen on knowing the coefficients of this model to further investigate the time-lag values. To do so, the coefficients of predictors for both models are plotted in Figures 11 and 13.

In both stations, t variable provides a negative coefficient for the `MLR_s` whereas et and p features have positive coefficients and are equally important in streamflow prediction.

Regarding the `MLR_lag` model, in Figures 12 and 14, the predictors with negative coefficients in both stations are shown. In Basel station, up until `t_lag_6` lag, the lagged variables of t feature, lags have positive coefficients. Whereas in Lobith station, up until `t_lag_12`, lags have positive coefficients in predicting streamflow, and lags above these values provide negative coefficients. In both stations, variable p and its lagged values have positive coefficients in predicting streamflow. Regarding lagged variables of et feature, the majority of its lagged variables have negative coefficients in both stations. These findings are against the results gained from the correlation plots above where lags of the variables t and et have high positive correlation and lags of the variable p have low correlation within the correlation periods mentioned.

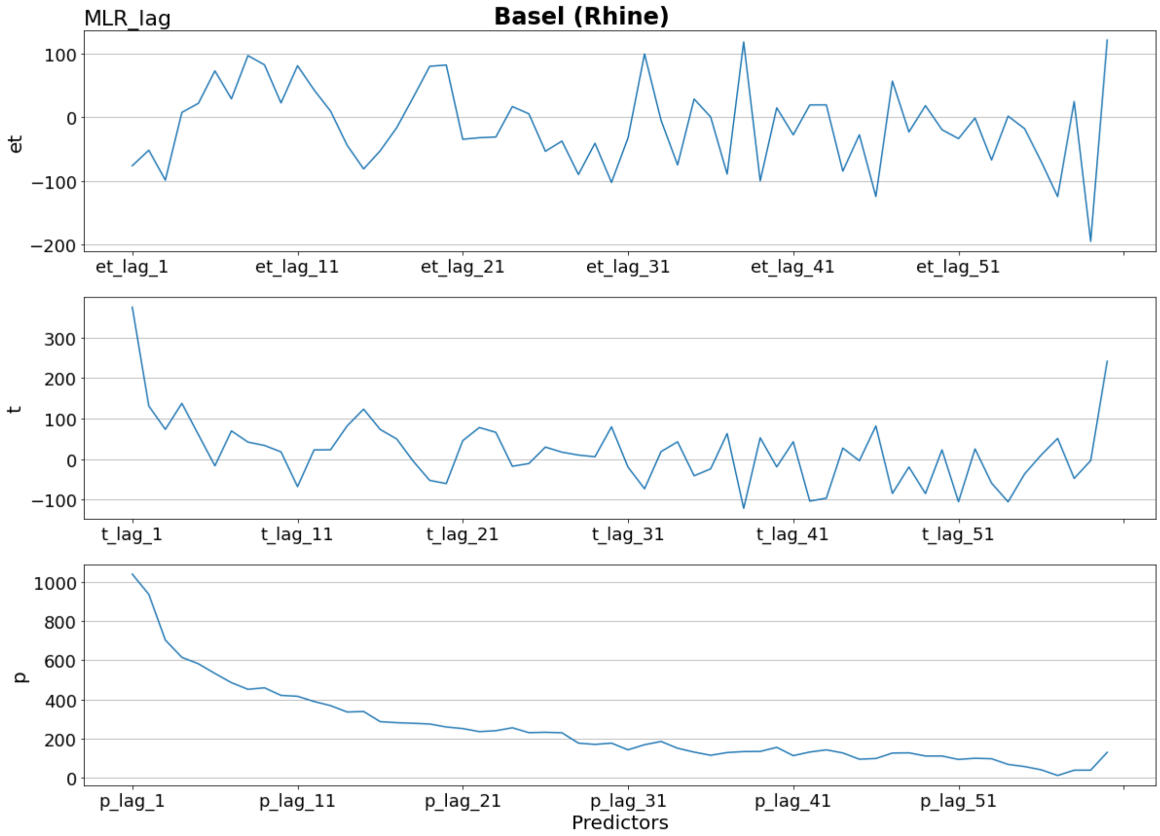

Figure 6.5: coefficients of the MLR models in Basel station

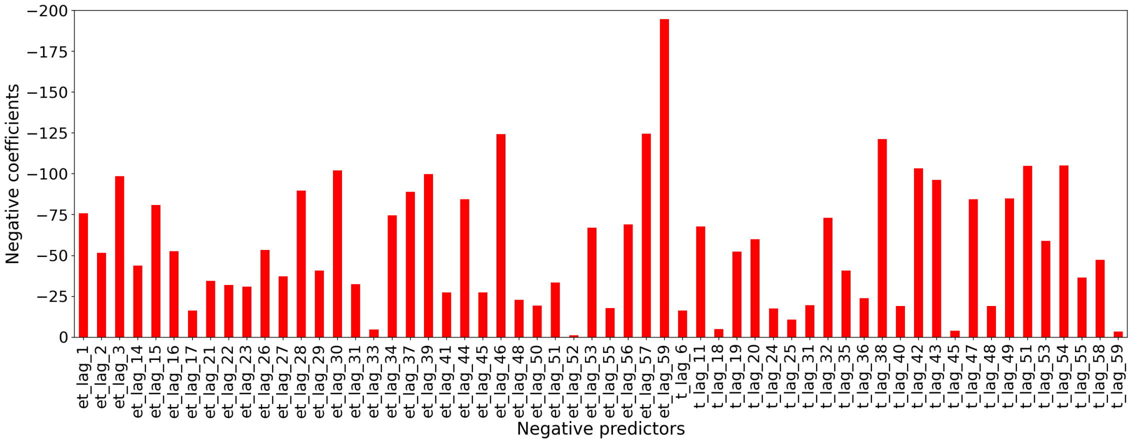

Figure 6.6: negative coefficients of the MLR_lag model in Basel station

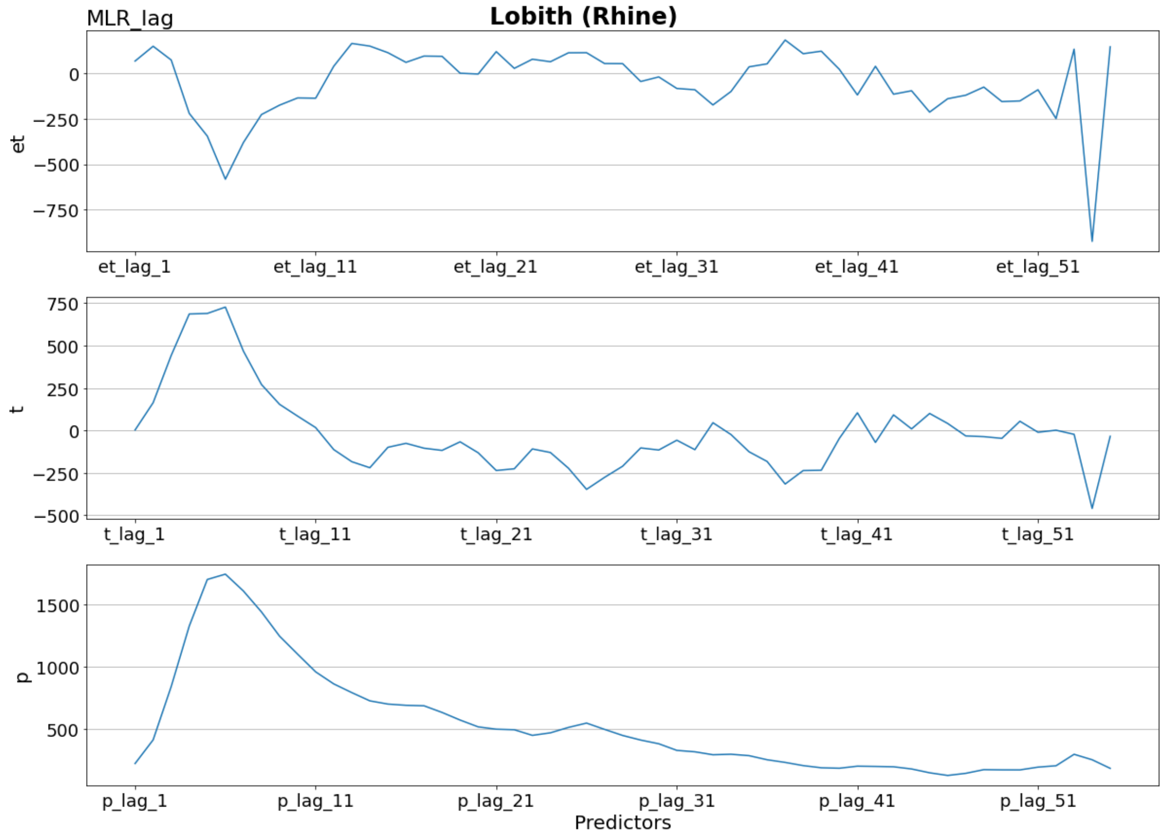

Figure 6.7: coefficients of the MLR models in Lobith station

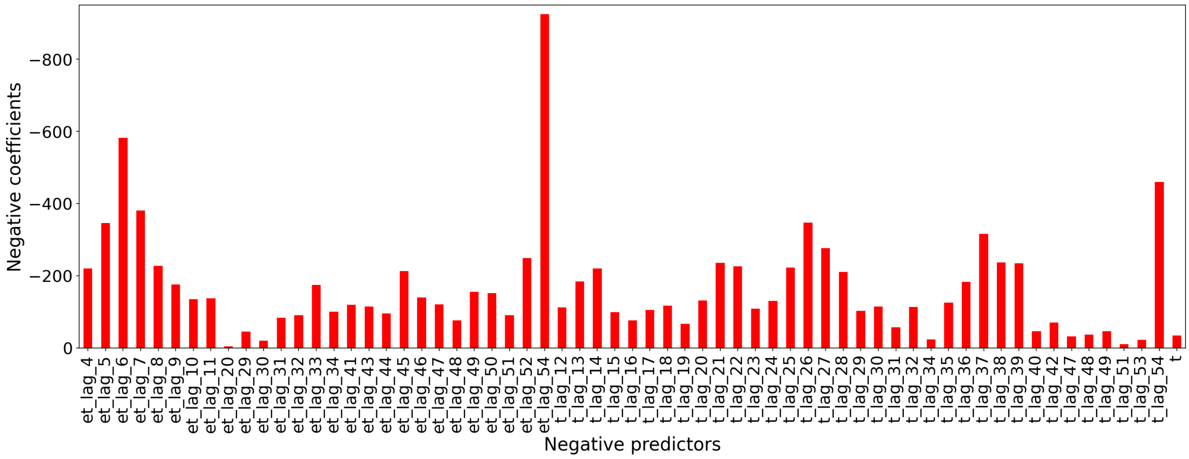

Figure 6.8: negative coefficients of the MLR_lag model in Lobith station

Bibliography

[1] D. N. Moriasi, J. G. Arnold, M. W. Van Liew, R. L. Bingner, R. D. Harmel, and T. L. Veith, “Model evaluation guidelines for systematic quantification of accuracy in watershed simulations,” *Transactions of the ASABE*, vol. 50, no. 3, pp. 885–900, 2007.

[2] C. A. Guimarães Santos and G. B. L. d. Silva, “Daily streamflow forecasting using a wavelet transform and artificial neural network hybrid models,” *Hydrological Sciences Journal*, vol. 59, no. 2, pp. 312–324, 2014.

[3] C. Shen, “A transdisciplinary review of deep learning research and its relevance for water resources scientists,” *Water Resources Research*, vol. 54, no. 11, pp. 8558–8593, 2018.

[4] F. Kratzert, D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter, and G. Nearing, “Benchmarking a catchment-aware long short-term memory network (lstm) for large-scale hydrological modeling,” *Hydrology and Earth System Sciences Discussions*, pp. 1–32, 2019.

[5] Y. B. Dibike and D. P. Solomatine, “River flow forecasting using artificial neural networks,” *Physics and Chemistry of the Earth, Part B: Hydrology, Oceans and Atmosphere*, vol. 26, no. 1, pp. 1–7, 2001.

[6] M. J. Best, G. Abramowitz, H. Johnson, A. Pitman, G. Balsamo, A. Boone, M. Cuntz, B. Decharme, P. Dirmeyer, J. Dong *et al.*, “The plumbing of land surface models: benchmarking model performance,” *Journal of Hydrometeorology*, vol. 16, no. 3, pp. 1425–1442, 2015.

[7] M. Gauch, J. Mai, S. Gharari, and J. Lin, “Data-driven vs. physically-based streamflow prediction models,” in *Proceedings of 9th International Workshop on Climate Informatics*, 2019.

[8] Z. M. Yaseen, A. El-Shafie, O. Jaafar, H. A. Afan, and K. N. Sayl, “Artificial intelligence based models for stream-flow forecasting: 2000–2015,” *Journal of Hydrology*, vol. 530, pp. 829–844, 2015.

[9] S. Duan, P. Ullrich, and L. Shu, “Using convolutional neural networks for streamflow projection in california,” *Frontiers in Water*, vol. 2, p. 28, 2020.

[10] Y. Shen, J. Ruijsch, M. Lu, E. H. Sutanudjaja, and D. Karssenberg, “Random forests-based error-correction of streamflow from a large-scale hydrological model: Using model state variables to estimate error terms,” *Computers & Geosciences*, vol. 159, p. 105019, 2022.

[11] A. Mosavi, P. Ozturk, and K.-w. Chau, “Flood prediction using machine learning models: Literature review,” *Water*, vol. 10, no. 11, p. 1536, 2018.

[12] L. Faik. (2021) Deep learning for time series forecasting: Is it worth it? (part i). [Online]. Available: https://medium.com/data-from-the-trenches/deep-learning-for-time-series-forecasting-is-it-worth-it-bfc95a1c9de4

[13] L. Yan, J. Feng, and T. Hang, “Small watershed stream-flow forecasting based on lstm,” in *International Conference on Ubiquitous Information Management and Communication.* Springer, 2019, pp. 1006–1014.

[14] X.-H. Le, H. V. Ho, G. Lee, and S. Jung, “Application of long short-term memory (lstm) neural network for flood forecasting,” *Water*, vol. 11, no. 7, p. 1387, 2019.

[15] M. Yueling, E. Matta, M. Dennis, S. Hanno, H. Reinhard *et al.*, “Can machine learning improve the accuracy of water level forecasts for inland navigation? case study: Rhine river basin, germany,” in *38th IAHR World Congress Panama City 2019, Water-Connecting the world.* Lucas Calvo, 2019, pp. 1979–1989.

[16] D. Feng, K. Fang, and C. Shen, “Enhancing streamflow forecast and extracting insights using long-short term memory networks with data integration at continental scales,” *Water Resources Research*, vol. 56, no. 9, p. e2019WR026793, 2020.

[17] M. Gauch, J. Mai, and J. Lin, “The proper care and feeding of camels: How limited training data affects streamflow prediction,” *Environmental Modelling & Software*, vol. 135, p. 104926, 2021.

[18] M. M. Ghiasi and S. Zendehboudi, “Decision tree-based methodology to select a proper approach for wart treatment,” *Computers in biology and medicine*, vol. 108, pp. 400–409, 2019.

[19] M. Gauch, J. Mai, S. Gharari, and J. Lin, “Streamflow prediction with limited spatially-distributed input data,” in *Proceedings of the NeurIPS 2019 Workshop on Tackling Climate Change with Machine Learning*, 2019.

[20] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[21] L. Ni, D. Wang, J. Wu, Y. Wang, Y. Tao, J. Zhang, and J. Liu, “Streamflow forecasting using extreme gradient boosting model coupled with gaussian mixture model,” *Journal of Hydrology*, vol. 586, p. 124901, 2020.

[22] C. Wade, *Hands-On Gradient Boosting with XGBoost and scikit-learn: Perform accessible machine learning and extreme gradient boosting with Python.* Packt Publishing Ltd, 2020.

[23] B. Sjardin, L. Massaron, and A. Boschetti, *Large scale machine learning with Python.* Packt Publishing Ltd, 2016.

[24] R. M. Adnan, Z. Liang, S. Heddam, M. Zounemat-Kermani, O. Kisi, and B. Li, "Least square support vector machine and multivariate adaptive regression splines for streamflow prediction in mountainous basin using hydrometeorological data as inputs," *Journal of Hydrology*, vol. 586, p. 124371, 2020.

[25] Z. Luo, J. Huang, K. Hu, X. Li, and P. Zhang, "Accuair: Winning solution to air quality prediction for kdd cup 2018," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1842–1850.

[26] (2015) Rossmann store sales, winner's interview: 1st place, gert jacobusse. [Online]. Available: https://medium.com/kaggle-blog/rossmann-store-sales-winners-interview-1st-place-gert-jacobusse-a14b271659b

[27] (2016) Grupo bimbo inventory demand, winners' interview: Clustifier alex andrey. [Online]. Available: https://medium.com/kaggle-blog/grupo-bimbo-inventory-demand-winners-interview-clustifier-alex-andrey-1e3b6cec8a20

[28] (2016) Rossmann store sales, winner's interview: 2nd place, nima shahbazi. [Online]. Available: https://medium.com/kaggle-blog/rossmann-store-sales-winners-interview-2nd-place-nima-shahbazi-ad7a4eb65629

[29] E. Sutanudjaja, L. Van Beek, S. De Jong, F. Van Geer, and M. Bierkens, "Calibrating a large-extent high-resolution coupled groundwater-land surface model using soil moisture and discharge data," *Water Resources Research*, vol. 50, no. 1, pp. 687–705, 2014.

[30] H. Tongal and M. J. Booij, "Simulation and forecasting of streamflows using machine learning models coupled with base flow separation," *Journal of hydrology*, vol. 564, pp. 266–282, 2018.

[31] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using long short-term memory (lstm) networks," *Hydrology and Earth System Sciences*, vol. 22, no. 11, pp. 6005–6022, 2018.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] M. Phankokkruad and S. Wacharawichanant, "A comparison of extreme gradient boosting and convolutional neural network-long short-term memory for service demand forecasting," in *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery.* Springer, 2019, pp. 547–556.

[34] Y. Wang, K. Lin, Y. Qi, Q. Lian, S. Feng, Z. Wu, and G. Pan, "Estimating brain connectivity with varying-length time lags using a recurrent neural network," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 9, pp. 1953–1963, 2018.

[35] Y. B. Lim, I. Aliyu, and C. G. Lim, "Air pollution matter prediction using recurrent neural networks with sequential data," in *Proceedings of the 2019 3rd International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, 2019, pp. 40–44.

[36] P. L. Fung, M. A. Zaidan, O. Surakhi, S. Tarkoma, T. Petäjä, and T. Hussein, "Data imputation in in situ-measured particle size distributions by means of neural networks," *Atmospheric Measurement Techniques*, vol. 14, no. 8, pp. 5535–5554, 2021.

[37] O. Surakhi, M. A. Zaidan, P. L. Fung, N. Hossein Motlagh, S. Serhan, M. AlKhanafseh, R. M. Ghoniem, and T. Hussein, "Time-lag selection for time-series forecasting using neural network and heuristic algorithm," *Electronics*, vol. 10, no. 20, p. 2518, 2021.

[38] Z. M. Yaseen, O. Jaafar, R. C. Deo, O. Kisi, J. Adamowski, J. Quilty, and A. El-Shafie, "Stream-flow forecasting using extreme learning machines: a case study in a semi-arid region in iraq," *Journal of Hydrology*, vol. 542, pp. 603–614, 2016.

[39] C. Gao, M. Gemmer, X. Zeng, B. Liu, B. Su, and Y. Wen, "Projected streamflow in the huaihe river basin (2010–2100) using artificial neural network," *Stochastic Environmental Research and Risk Assessment*, vol. 24, no. 5, pp. 685–697, 2010.

[40] Sklearn preprocessing library. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[41] F. G. Toro and A. Tsourdos, *UAV or drones for remote sensing applications.* MDPI-Multidisciplinary Digital Publishing Institute, 2018.

[42] Introduction to boosted trees. [Online]. Available: https://xgboost.readthedocs.io/en/latest/tutorials/model.html#the-structure-score

[43] E. Sutanudjaja, R. Van Beek, N. Wanders, Y. Wada, J. Bosmans, N. Drost, R. Van Der Ent, I. De Graaf, J. Hoch, K. De Jong *et al.*, "Pcr-globwb 2: a 5 arcmin global hydrological and water resources model, geosci. model dev., 11, 2429–2453," 2018.

[44] L. Van Beek, Y. Wada, and M. F. Bierkens, "Global monthly water stress: 1. water balance and water availability," *Water Resources Research*, vol. 47, no. 7, 2011.

[45] L. Van Beek and M. Bierkens, "The global hydrological model pcr-globwb: conceptualization, parameterization and verification," *Utrecht University, Utrecht, The Netherlands*, vol. 1, pp. 25–26, 2009.

[46] Scipy library. [Online]. Available: https://scipy.org/

[47] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[48] Keras tuner. [Online]. Available: https://keras.io/keras_tuner/

[49] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[50] Hyperband method in keras tuner. [Online]. Available: https://www.tensorflow.org/tutorials/keras/keras_tuner#:~:text=Hyperband%20determines%20the%20number%20of,value%20for%20the%20validation%20loss.&text=Run%20the%20hyperparameter%20search.

[51] F. Galton, "Regression towards mediocrity in hereditary stature." *The Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, pp. 246–263, 1886.

[52] K. Pearson, "Vii. mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia," *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, no. 187, pp. 253–318, 1896.

[53] Gridsearchcv in scikit learn library. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[54] Xgboost parameters. [Online]. Available: https://xgboost.readthedocs.io/en/latest/parameter.html